ISSN: 2313 - 4887

ANNA UNIVERSITY
PROGRESS THROUGH KNOWLEDGE

உத்தமம்
INFITT®
International Forum for Information Technology in Tamil

தமிழ் இணையக் கல்விக்கழகம்
TAMIL VIRTUAL ACADEMY

18–ஆவது தமிழ் இணைய மாநாடு

மாநாட்டுக் கட்டுரைகள்

செப்டம்பர் 20–22, 2019

அண்ணா பல்கலைக் கழகம், சென்னை

தொகுப்பு

முனைவர் வாசு அரங்கநாதன்
பென்சில்வேனியாப் பல்கலைக்கழகம்
முனைவர் எல். சோபா நாயர்
அண்ணா பல்கலைக்கழகம்

உலகத் தமிழ்த் தகவல் தொழில்நுட்ப மன்றம் (உத்தமம்)

தமிழ்நாடு அரசு
வாய்மையே வெல்லும்

सत्यमेव जयते
CIIL

KUMARAGURU
college of technology
character is life

18 வது தமிழ் இணைய மாநாடு

செப்டெம்பர் 20-22, 2019,

அண்ணா பல்கலைக்கழகம்,

சென்னை, தமழ்நாடு, இந்தியா

மாநாட்டு கட்டுரைகள்

தொகுப்பு

முனைவர் வாசு அரங்கநாதன்

முனைவர் சோபா ல

பதிப்பு

உலகத் தமிழ்த் தகவல் தொழில் நுட்ப மன்றம் (உத்தமம்)
அமெரிக்காவின் கலிபோர்னியா மாநிலத்தில் இலாப நோக்கமற்ற
நிறுவனமாகப் பதிவுசெய்யப்பட்ட நிறுவனம்

ISSN: 2313 - 4887

18th Tamil Internet Conference

Anna University,
Chennai, Tamilnadu, India
September 20-22, 2019,

Conference Papers

Compiled by

Dr. Vasu Ranganathan

Dr. Sobha L

Published by:

International Forum for Information Technology in Tamil (INFITT)

A non-profit Organization registered in California, USA

Patrons

Professor M.K. Surappa
Vice-Chancellor, Anna University, Chennai

Professor M. Anandakrishnan
Former Vice- Chancellor of Anna University, Chennai

Professor M. Ponnavaiko
Provost, Vinayaka Mission's Research Foundation, Tamil Nadu

Professor K. Ramasamy
Director, Academic and Faculty Development, SRM University, Chennai

Professor V. Murugesan
Vice-Chancellor, Annamalai University, Annamalai Nagar.

Prof. D. G. Rao
Central Institute of Indian Languages, Mysore

Thiru Shankar Vanavrayar, Joint Correspondent,
President, Kumaraguru College of Technology, Coimbatore.

International Organizing Committee

Dr. A. Murugaiyan, School of Advanced Studies – EPH, Paris (Chair)

Dr. T.V. Geetha, Anna University (CoChair)

Dr. K. Kalyanasundaram, EPFL, Lausanne, Switzerland

Dr. Arul Veerappan, NY, USA (Member)

Dr. Sornam Sankar, NJ, USA (Member)

Mr. S. Maniam, Singapore (Member)

Dr. Seethalakshmi, Singapore (Member)

Dr. C.M. Elantamil, Malaysia (Member)

Mr. Natkeeran, Toronto, Canada (Member)

Mr. T. Thavaruban, Srilanka (Member)

Conference Programme Committee

Dr. Vasu Renganathan, University of Pennsylvania, USA (Chair)

Dr. Sobha L, AU-KBC Research Center, Anna University, Chennai (CoChair)

Dr. Anand Kumar, M, NIT, Surathkal, India

Dr. Arulmozi S, HCU, Hyderabad, India

Dr. Aravindan Chandrabose, SSN College of Engg, Chennai, India

Dr. Balaji Jagan, Bangalore, India

Dr. Baskaran Sankaran, DD Science Solutions India, Selam, India

Dr. Dhanalakshmi Giri, V, Govt. College, Krishnagiri, India

Dr. Ganesan M, Annamalai University, India

Dr. Ganesh Ramakrishnan, IIT-Bombay, Mumbai, India

Dr. Geetha T.V., Anna University, Chennai, India

Dr. Hema Murthy, IIT-Madras, Chennai, India

Dr. James Anthonysamy, Tamil Virtual Academy, Chennai, India

Mr. Krishna Doss Mohan, Microsoft India, Hyderabad, India

Dr. Lakshmana Pandian, Pondicherry Engineering College, Pondicherry, India

Dr. Nagarajan T., SSN College of Engg, Chennai, India

Dr. Narayan Choudary, Central Institute of Indian Languages, Mysore, India

Dr. Pattabhi RK Rao., AU-KBC Research Centre, Anna University, Chennai, India

Dr. Rajendran S, Amrita University, Coimbatore, India

Dr. Rajeswari Sridhar, NIT, Tiruchi, India

Dr. Ramamoorthy, L., Central Institute of Indian Languages, Mysore, India

Dr. Ranjani Parthasarathi, Anna University, Chennai

Mr. Sarveswaran K., University of Moratuwa, Sri Lanka

Dr. Sindhuja G., Uniphore, Chenna, India

Mr. Shrinivasan T., Chennai, India

Dr. Uma Maheswari, NUS, Singapore

Dr. Vijay Sundar Ram R, AU-KBC Research Centre, Anna University, Chennai, India

Local Organizing Committee

Dr. Ranjani Parthasarathi, Anna University, Chennai (Chair)

Mr. S.Maniam, Singapore, (CoChair)

Dr. Abirami, M, Anna University, Chennai

Dr. ArulChelvan S, Anna University, Chennai

Dr. Bama Srinivasan, Anna University, Chennai

Dr. V. Dhanalakshmi Giri, Govt. College, Krishnagiri

Dr. M. Ganesan, Annamalai University

Mr. J. Ganesh, Bosch Bangalore

Dr.T.V. Geetha, Anna University, Chennai

Mr. Kodaikkavirinaadan U, Anna University, Chennai

Dr. G.S. Mahalakshmi, Anna University, Chennai

Dr. T. Mala, Anna University, Chennai

Dr. D. Manjula, Anna University, Chennai

Dr. Pattabhi RK Rao, AU-KBC, MIT campus, Chennai

Dr. N. Rajendran, Anna University, Chennai

Dr. Sakthivel Ramaswamy, IIT Madras

Mr. Selva Murali, Dharmapuri, India

Dr. S. Senthilkumar, Anna University

Mr. T. Shrinivasan, Chennai

Dr. Sobha L, Anna University, Chennai

Dr. C.N. Subalalitha, SRM University

Dr. Vijay Sundar Ram, AU-KBC MIT Campus, Chennai

Dr. M. Vijayalakshmi, Anna University, Chennai

Dr. Chitrakala S, Anna University, Chennai

Dr. Thangaraj N, Anna University, Chennai

Dr. Pandiayaraju, Anna University, Chennai

T. Munirathinam, Anna University, Chennai

Mr.G .Manikandan, Anna University, Chennai

Dr. Bhuvaneswaran R S, Anna University, Chennai

Dr. B.L.Velammal, Anna University, Chennai

Dr. Subalalitha, SRM, Chennai

Dr. R. Geetha Ramani, Anna University, Chennai

Dr. S. Swaminathan, Anna University, Chennai

Dr. S. Renugadevi , Anna University, Chennai

Exhibition Hub & Workshop

Pixibit Pte Ltd, Singapore

InfoMedia

Kumaraguru College of Technology, Coimbatore

Dr. Senthil Kumar

Mr. Mainium

Community Hub Speakers

Dr. Badri Seshadri, Chennai (Chair)

Mr. TNC Ventkatarangan, Chennai

Mr. Maalan Narayanan, Chennai

Mr. Venkatesh Radhakrishnan, Chennai

Mr. Gunasegaran, Singapore

Mr. Senthilnathan, Chennai

Mr. Elantamil, Malaysia

AU Steering Committee

Patrons

Professor M.K. Surappa,
Vice-Chancellor, Anna University, Chennai

Conveners

Dr. L.Karunamoorthy, Registrar, Anna University

Dr. Sobha L., AU-KBC MIT Campus, Chennai

Dr. T.V. Geetha, Dean, CEG, Anna University

Dr. Ranjani Parthasarathy, IST, Anna University

Members

Dr. Abirami, M, Anna University, Chennai

Dr. S. Senthilkumar, Anna University, Chennai

Dr. D. Manjula, Anna University, Chennai

Dr. Pattabhi RK Rao. AU-KBC, MIT campus, Chennai

Dr. Vijay Sundar Ram, AU-KBC MIT Campus, Chennai

Dr. M. Vijayalakshmi, Anna University, Chennai

Dr. G.S. Mahalakshmi, Anna University, Chennai

Dr. T. Mala, Anna University, Chennai

Dr. Bama Srinivasan, Anna University, Chennai

Dr. ArulChelvan S, Anna University, Chennai

Dr. Rhymend Uthariaraj V, RCC, Anna University, Chennai

Dr. Bhuneswaran, R.S., RCC, Anna University, Chennai

Dr. Murugan, RCC, Anna University, Chennai

ACKNOWLEDGEMENTS

INFITT would like to thank the following sponsors for their generous support

Platinum Sponsor:

Tamil Virtual Academy, Chennai

Host Sponsor:

Anna University, Chennai (Co-host)

Gold Sponsor:

Tamil Development Department, Tamilnadu Govt.

Silver Sponsor

Central Institute of Indian Languages (CIIL), Mysore

Bronze Sponsor:

Kumaraguru College of Technology, Coimbatore

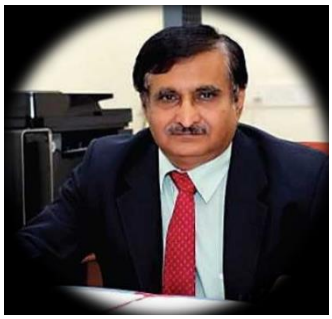
Felicitation Messages

வாழ்த்துச் செய்திகள்

ANNA UNIVERSITY , CHENNAI
PROGRESS THROUGH KNOWLEDGE

Prof. M.K. Surappa

Vice-Chancellor



It gives me immense pleasure in writing this foreword for the Eighteenth edition of the Tamil Internet Conference, TIC 2019, being held at Anna University, Chennai, from Sep 20-22, 2019. Unprecedented technological developments that improve the lives of common man are taking place in the area of AI, and Machine Learning. Bringing this to the doorsteps of the actual users requires the interfaces to be in the language of the people, thus moving Natural Language Processing (NLP) to the forefront. In our country, India, which is multi-lingual, NLP becomes interesting and challenging to handle the various complexities of each language. While research in NLP in Indian languages was started more than three decades ago, not all languages have received the same amount of attention. However, NLP work in Tamil has been progressing steadily for the last two decades, and the Tamil Internet Conference series has been a flagship event showcasing the developments in Tamil Computing. Supported by the Tamil Diaspora world-wide through the organization INFITT, this Conference helps to consolidate the work relating Tamil and technology in all its forms, ranging from Computational linguistics, and tools for Tamil processing to mobile, and virtual reality applications.

The theme of this year's conference is "Tamil Robotics and Language Processing" which highlights the timely progress in the world of Tamil Computing. Anna University has been working and contributing in the area of Tamil Computing for the past twenty years, and we are happy to host this event this year, while we are celebrating the 225th year of our primary institution, College of Engineering, Guindy (CEG).

We would like to encourage students and researchers across Tamil Nadu to participate in this important event.

I congratulate all the delegates who have contributed technical papers for this conference proceedings, and wish this event a grand success.

திரு. தி. சுப்பிரமணியம்

வாழ்த்துச்செய்தி

முதல் தமிழ் இணைய மாநாடு 1997 ஆம் ஆண்டு வெற்றிகரமாகவும் பயனுள்ள வகையிலும் நடைபெற்ற பின்னர், பல தமிழ் இணைய மாநாடுகள் உலகில் தமிழர் அதிகமாக வாழும் நாடுகளில் நடத்தப்பட்டு, இணையத்தில் தமிழ் மொழியின் பயன்பாடு சிறப்பான ஓரிடத்தைப் பெற்றது. தற்போது 18 ஆம் தமிழ் இணைய மாநாடு பல ஆண்டுகளுக்குப் பின் சென்னையில் நடைபெறுகிறது.

இந்த ஆண்டு, சென்னையில் நடைபெறும் இந்த இணைய மாநாட்டிற்குப் பல அரசு சார்ந்த மற்றும் தனியார் நிறுவனங்களின் ஆதரவு நமக்குக் கிடைத்துள்ளது என்பது மிக்க மகிழ்ச்சிக்குரியதாகும். புகழ்பெற்ற சென்னை அண்ணா பல்கலைக்கழகம், தமிழ் இணையக் கல்விக் கழகம் மற்றும் இந்திய மொழிகளின் நடுவண் நிறுவனம் ஆகியன இணைந்து செப்டம்பர் திங்கள் 20 முதல் 22 வரை, இணைய மாநாடு 2019 னை சென்னை அண்ணா பல்கலைக் கழக வளாகத்தில் நடத்துகின்றன. **"தானியங்கிக் கருவிகளில் தமிழ்மொழிப் பயன்பாடு"** என்னும் மையக் கருத்தில் இந்த இணைய மாநாடு ஆய்வு விவாதங்களை மேற்கொள்ள இருக்கிறது.

மிக வேகமாக மாறிவரும் இவ்வுலக இயக்கத்தில், நிரலாக்க மொழிகள், சார்ந்த சாதனங்கள் மற்றும் குறிப்பிட்டுச் சொல்லக்கூடிய வாழ்க்கைத் தொழில் சார்ந்த திறமைகள் ஆகிய அனைத்தின் பயன்காலமும் குறுகியதாகவே உள்ளது. அதே சமயத்தில், திறனாய்வு சிந்தனை, தீர்வுகாண் திறன், முடிவெடுக்கும் திறன் மற்றும் குழுவாக இயங்கும் பணி அமைப்பு ஆகியவை வெற்றியடைந்தே ஆகவேண்டிய நிலையை, தவிர்க்க

இயலாததாகக் கொண்டுள்ளன. வாழ்வியல் திறன்கள் என மொத்தமாக அழைக்கப்படும் இந்தத் திறன்கள் அனைத்தையும், கலையறிவு சார்ந்தன வற்றை முதன்மையாகக் கொண்டுள்ள பாரம்பரிய வகுப்பறைச் சூழ்நிலைகளில் மட்டும் வளர்க்க முடியாது.

"இயந்திரவியல் கல்வி" ஒரு குழந்தையின் முழுமையான வளர்ச்சிக்கு உறுதுணையாய் உள்ளதால், கல்விப் புலத்தில் அது ஒரு புதியதொரு பரிமாணத்தைப் பெற்று வளர்ந்து வருகிறது. இக்கல்விப் பிரிவானது, வாழ்வியலுக்குத் தேவையான அடிப்படை திறன்கள் வளர்ச்சி மற்றும் பயனுறுத்தலை ஒருமுகப்படுத்தி, வாழ்க்கையில் எதிர்நோக்கும் பிரச்னைகளுக்குத் தீர்வு காணும் வகையில், அறிவியல், தொழில் நுட்பம், பொறியியல் மற்றும் கணிதவியல் கற்பதற்கு ஏற்ற வகையில் மாணவர்களுக்கு முழுமையான திறன் வழங்கும் திட்டத்தினை அமைத்துத் தருகிறது. மொத்தத்தில், இதன் முக்கிய தீர்வு இது தரும் நெறிமுறைகள் மற்றும் இயற்படுத்துபவர்களிடம் உள்ளது.

இயந்திரவியலுடன் இணைந்து செயல்படும் வன்பொருட் சாதனங்கள் மற்றும் சார்ந்த மென்பொருள் தொகுப்புகள், மாணவர்கள் எண்ணற்ற தீர்வுகள் மற்றும் ஆற்றல் வாய்ந்த முடிவுகளை மேற்கொண்டு எத்தகைய பிரச்னையையும் எதிர்கொள்கின்ற வகையில், ஆக்கப்படைப்பு திறனுக்கான சுதந்திரத்தையும் பகுப்பாய்வுக்கான திறமைகளையும் வழங்குகின்றன.

அப்படிப்பட்ட தொழில்நுட்பப் பயன்பாடு மாணவர்களிடம் தன்னம்பிக்கையை உருவாக்குகிறது. அது எதனையும் சாதிக்கின்ற மனப்பாங்கினை அவர்களுக்குத் தருகிறது. இந்த மனப்பாங்கு ஒரு குழந்தையின் அடிப்படை உத்வேகத்தினை அதிகரிக்கச் செய்கிறது. இந்த இடத்தில், எந்த ஒரு இயந்திரவியல் தீர்வும் வெற்றிகரமாக அமையவேண்டும் எனில், அது செறிந்த பொருளைக் கொண்டு அதனுடன் தொடர்புடையதாக இருக்க வேண்டும் என்பது கட்டாயமாகும் என்பதை நினைவில் கொள்ள வேண்டும்.

எதிர்கால வாழ்வில் 'இயந்திரவியல்' பெரிய அளவில் இடம் பெற இருப்பதால், நாம் இன்றைய நம் மாணவர்களை இந்த மாற்றத்திற்குத் தயார் செய்திட வேண்டிய கட்டாயத்தில் இருக்கிறோம். தற்போதைய இந்தியக் கல்வி முறையில் இயந்திரவியல்

பயன்பாடு குறித்த கல்வியானது கோட்பாடுகளும் செயல்முறைப் பயிற்சியும் கொண்ட ஒரு கலவையாக உள்ளது. பள்ளிகளில் இதனைச் சரியாகப் பயன்படுத்தினால், பலவகை பாடத்திட்டங்களின் செயல்பாடுகளுக்கான அடித்தளமாய் அமையும். அத்துடன் கணிதவியல், அறிவியல் கோட்பாடுகள், கட்டமைப்பு தொழில் நுட்பம் மற்றும் கணினி நிரலாக்கம் ஆகியவற்றைக் கற்றுத் தருவதற்கான மிகச் சிறந்த வளமான களமாக உருவாகும்.

இயந்திரவியல் தொழில் நுட்பப் பயன்பாடு, பாரம்பரிய கற்றல் வழிகளுக்குச் செயலற்ற இலக்குகளாக மாணவர்கள் இயங்கும் நிலையை மாற்றி, மிகக் குறுகிய காலத்தில் எதனையும் கற்றுக் கொள்ளும் கற்றல் முனைப்புடன் கூடியவர்களாக அவர்களை மாற்றும். வகுப்பறைகளில் இந்த எந்திரன்கள் பயன்படுத்தப்படுகையில், இரண்டு அல்லது நான்கு மாணவர்கள் அடங்கிய குழு ஒவ்வொன்றுக்கும் ஒரு இயந்திரன் தரப்படும். இந்த தொழில்நுட்பம் மாணவர்களிடம் அடிப்படை தகவல் தொடர்பு திறனையும் தனிமனித திறன்களையும் வளர்த்துக் கொள்வதை ஊக்குவிக்கும். இன்றைக்கு பணி அளிப்பவர்கள், மாணவர்கள் தங்கள் சக தோழர்களிடம் அல்லது உடன் பணியாற்றுபவர்களிடம் இணைந்து செயல்படுவது மற்றும் கூட்டமைவான கோட்பாடுகளைத் தெரியப்படுத்திப் பகிர்ந்து கொள்வதனை மிக முக்கிய திறமைகளாக எண்ணுகின்றனர்.

இயந்திரன்களை வகுப்பறைகளில் பயன்படுத்துவதனால் மாணவர்கள் இதுவரை தாங்கள் அறிந்து கொள்ளாத பல பணி வாய்ப்புகளுக்கு அறிமுகப்படுத்தப் படுகிறார்கள். தகவல் தொழில் நுட்பமும் பொறியியலும் எளிமையும் வேடிக்கையும் நிறைந்தது என அறிவதற்கு இயந்திரவியல் ஒரு மிகச் சிறந்த கல்வியாகும். இயந்திரவியல் சார்ந்த ஒரு பொறியியல் திட்டத்தினை வெற்றிகரமாக முடித்திட ஒருவர் மின்னணுவியல், இயந்திரவியல் மற்றும் இரசாயனவியல் கோட்பாடுகளை நன்கு அறிந்திருக்க வேண்டும். ஆகவே, மாணவர்களை இயந்திரவியல் துறைக்கு அறிமுகப்படுத்த வேண்டியது இன்றைய காலத்தின் கட்டாயமாகிறது.

நம் கல்வி முறையில் இயந்திரவியல் இரு வகைகளில் இடம் பெறுகிறது. முதலாவதாக, இயந்திரவியலை வகுப்பறைகளில் பயன்படுத்துவது. இயந்திரன்களை வகுப்பறையில் பயன்படுத்துவதனால் ஏற்படும் நன்மைகளையும் சிறப்புகளையும் அறிந்த ஆசிரியர்கள்

அதனை இன்றைய கல்வி திட்டத்தில் பயன்படுத்தும் வகைகளை உருவாக்குவதாகும். இரண்டாவதாக, பள்ளிகளில் இயந்திரவியல் மன்றங்களை ஏற்படுத்தி அதில் ஆசிரியர்களின் மேற்பார்வையில் மாணவர்களை இயந்திரன்களைப் பயன்படுத்துவது குறித்து அறியச் செய்வது.

மேலே சுருக்கமாகக் கூறப்பட்ட அனைத்துப் பிரிவுகளிலும் ஆய்வுக் கட்டுரைகளை இந்த தமிழ் இணைய மாநாட்டில் உலகெங்கிலும் இருந்து வந்துள்ள பொறியியல் வல்லுநர்கள் படைக்கின்றனர் என்பது மிகவும் மகிழ்ச்சிக்குரிய செய்தியாகும், வங்காள தேசம், அமெரிக்கா, ஐரோப்பிய நாடுகள், ஸ்ரீலங்கா, சிங்கப்பூர், மலேசியா மற்றும் இந்தியா ஆகிய நாடுகளிலிருந்து வல்லுநர்களும் ஆய்வாளர்களும் கலந்து கொள்கின்றனர். தமிழ்நாட்டில் இயங்கும் பல கல்வி மற்றும் ஆய்வு நிறுவனங்கள் நம் அமைப்புடன் இணைந்து செயலாற்றுவது பெருமைக்குரிய ஒன்றாகும். வருங்காலத்தில் இது போன்ற பல கல்வி மற்றும் ஆய்வு நிறுவனங்கள் இணையும் வாய்ப்பினை இந்த மாநாடு தரும் என்று எதிர்பார்க்கிறேன்.

இந்த இணைய மாநாட்டின் முன்னோட்டமாக அண்ணா பல்கலைக் கழகத்தால் 'கொத்துநிரல் அரங்கம்' (ஹேக்கத்தான்) மூன்று நாள் நடத்தப்பட்டது. இதில் மாணவர்கள், பேராசிரியர்களின் வழிகாட்டலில் நிரல்களை உருவாக்கினர். தங்கள் பட்டப்படிப்பை முடிக்க இருக்கும் பொறியியல் மாணவர்கள் இதில் கலந்து கொண்டது இந்த மாநாட்டின் நோக்கம் உரியவர்களைச் சென்றடைந்தது என்றே சொல்ல வேண்டும்.

தமிழ் இணைய மாநாட்டினை என்னோடு இணைந்து அனைத்து தமிழ் இணைய ஆர்வலர்களும் எதிர்நோக்கி உள்ளனர். மாநாட்டிற்கு அனைவரையும் அன்புடன் வரவேற்கும் இந்த வேளையில், இதற்கென உதவி நல்கிய அனைத்து பிரிவினருக்கும் உலகத் தமிழ் தகவல் தொழில் நுட்ப மன்றத்தின் சார்பில் நன்றி தெரிவித்து மகிழ்கிறேன்.

மிக்க அன்புடன்

சுப்பிரமணியம்
தலைவர் - உத்தமம்
சிங்கப்பூர்.

பேராசிரியர் மு. பொன்னவைக்கோ

வாழ்த்துச்செய்தி

முதல் தமிழ் இணைய மாநாடு, 1997-ஆம்ஆண்டு, மே மாதம் சிங்கப்பூரில் 'தமிழ்இணையம்97' என்ற பெயரில் நடைபெற்றது. இரண்டாவது தமிழ் இணைய மாநாடு, 'தமிழ்இணையம்99' என்ற பெயரில் சென்னையில் 1999-ஆம் ஆண்டு பிப்ரவரி மாதம் நடைபெற்றது இம்மாநாட்டை அடுத்து 'தமிழ்இணையம் 2000' மாநாடு இலங்கையில் நடத்தத் திட்டமிடப்பட்டு இருந்தது. அந்த மாநாட்டின் முன்னேற்பாட்டுக் கூட்டம் 2000-ஆம் ஆண்டு நவம்பர் மாதம் இலங்கையில் நடைபெற்றது. அந்தக் கூட்டத்தில் சுவிட்சர்லாந்து நாட்டிலிலுள்ள முனைவர் கல்யாணசுந்தரம் அவர்கள் , உலக அளவில் ஒரு இணையத் தமிழ் ஆய்வுக்குழு அமைக்க வேண்டுமென வரைந்து அனுப்பியிருந்த திட்டத்தை திரு.அருண்மகிழ்நன் முன்மொழிந்தார். அந்தக் கலந்துரையாடலில் பிறந்ததுதான் 'உத்தமம்' என்னும் உலகத் தமிழ்த் தகவல் தொழில்நுட்ப மன்றம் . இம்மன்றத்திற்குப் பெயரிடும் பெருமை எனக்குக் கிட்டியது நான் பெற்ற பேறு. அந்த மாநாட்டை அடுத்து சிங்கப்பூரில் 2000-ஆம் ஆண்டு ஜூலை மாதம் 22-24-ஆம் நாட்களில் நடைபெற்ற தமிழ் இணைய மாநாட்டில் 'உத்தமம்' (INFITT) தொடங்கி வைக்கப்பட்டது. இணையத் தமிழின் ஆய்விற்காக உத்தமத்தில் தமிழ்க் கலைச்சொல் ஆக்கல், யூனிகோடுதமிழ் (UNICODE Tamil) ஆய்வு, இணையதள தமிழ்முகவரி வடிவமைத்தல், தமிழ் வரிவடிவக் குறியீட்டுத் தரப்பாடு, ஆங்கில வரிவடிவத் தமிழ் எழுத்துத் தரப்பாடு, தமிழ் எழுத்துரு அறிதல் (Tamil OCR), லினக்ஸில் தமிழ் (Tamil in Linux), தமிழ் அனைத்து எழுத்துரு தரம் அமைத்தல், ஆகிய பணிகளுக்காக எட்டு ஆய்வுப் பணிக்குழுக்கள் (Working Groups) நிறுவப்பட்டன. பணிக்குழுக்களின் செயற்பாடுகள் பற்றியும் கணித்தமிழ் மற்றும் இணையத் தமிழ்வளர்ச்சி பற்றியும் ஒவ்வொரு ஆண்டும் உத்தமம் நடத்திய தமிழ் இணைய மாநாடுகளில் கலந்தாய்வு செய்யப்பட்டுள்ளன. இதுவரை (1997-லிருந்து 2018 வரை) சிங்கப்பூர், தமிழ்நாடு,

மலேசியா, அமெரிக்கா, ஜெர்மனி, புதுச்சேரி, கனடா, ஆகிய நாடுகளில் 17 மாநாடுகள் நடைபெற்றுள்ளன. தமிழ் இணையம் 2001, 2002, மற்றும் 2003 மாநாடுகளில் நிகழ்ந்த சில நிகழ்ச்சிகளின் விளைவாக உத்தமம் அமைப்பின் மேலாண்மையில் ஏற்பட்ட சில பின்னடைவுகளால், 2005, 2006, 2007, 2008 ஆகிய நான்கு ஆண்டுகளில் தமிழ் இணைய மாநாடுகள் நடைபெறவில்லை. எனவே 21 ஆண்டுகளில் 21 மாநாடுகளுக்குப் பதிலாக 17 மாநாடுகள் மட்டுமே நடைபெற்றுள்ளன. இப்பொழுது 18-வது மாநாடு தமிழ்நாட்டில், சென்னையில் அண்ணா பல்கலைக்கழகத்தில் நிகழவுள்ளது என்பதை அறிய மிகுந்த மகிழ்ச்சி அளிக்கின்றது. இது தமிழ்நாட்டில் நிகழும் எட்டாவது மாநாடு என்பதும் மகிழ்ச்சிக்குரிய செய்தி.

இந்த 21 ஆண்டுகளில் நாம் சாதித்தவை என்ன என்று எண்ணிப் பார்க்கவேண்டிய நேரம் இது. ஏராளமான தமிழ் இணைய தளங்கள் பிறந்துள்ளன. பல்வேறு எழுத்துரு தரப்பாடுகளிலிருந்து இரண்டு எழுத்துருத் தரப்பாடுகள் - ஒருங்குறித்தமிழ் (Unicode Tamil), அனைத்து எழுத்துருத் தரப்பாடு (TACE-16), ஆகிய இரண்டு தரப்பாடுகள் தமிழக அரசால் அரசின் தரப்பாடுகளாக ஏற்கப் பெற்றுள்ளோம். சொற்செயலிகள், தமிழ் எழுத்துரு அறி மென்மம் (Tamil OCR), பேச்சுத் தமிழை எழுத்துத் தமிழாக்கும் மென்மம், எழுத்துத் தமிழைப் பேச்சுத் தமிழாக்கும் மென்மம், தமிழ் இயல்மொழிச் செயலாக்கம் போன்ற பல்வேறு மென்பொருள் உருவாக்கப் பெற்றுள்ளோம். உலகு தழுவி வாழும் தமிழ்மக்களும், தமிழில் ஈடுபாடு உள்ள மற்றையோரும், தமிழ் மொழியைக் கற்கவும், தமிழர் வரவாறு, கலை, இலக்கியம், பண்பாடு பற்றி அறிந்து கொள்ளவும், மழலைக் கல்வி முதல் பட்டப் படிப்பிற்கான பாடப் பொருள்களையும், மிகப் பெரிய தமிழ் மின்நூலகத்தையும் தன்னுட்கொண்டு தமிழ்ப்பணி ஆற்றிவரும் சிறந்ததொரு தமிழ்இணையக் கல்விக் கழகம் கிடைக்கப் பெற்றுள்ளோம.

ஆனால் இப்படைப்புகளெல்லாம் மக்களைப் போய்ச் சேர்ந்துள்ளனவா? தமிழை ஆட்சி மொழியாகவும் வழக்கு மொழியாகவும் கொண்டுள்ள நாடுகளின் அரசுகள் ஏற்றுச் செயல்படுத்துகின்றனவா? இல்லையெனில் அதற்கு உத்தமம் என்ன செய்ய வேண்டும்? எப்படிச் செயல்பட வேண்டும் ? என்பவற்றை இம்மாநாட்டில் கலந்தாய்ந்து முடிவு செய்ய

வேண்டிய நிலையில் உள்ளோம். மேலும் இணையதளத் தமிழ் முகவரி வடிவமைக்கும் பணி இன்னும் நிறைவு பெறாமல் உள்ளது. இப்பணி நிறைவு பெறச் செய்ய வேண்டியவை பற்றி முடிவெடுக்க வேண்டிய நிலையில் உள்ளோம்.

இணையப் பயன்பாட்டிற்கு ஒருங்குறி தமிழ் எழுத்துரு (Unicode Tamil) தரப்பாடு என்றும், பிற பயன்பாடுகளுக்கு அனைத்து எழுத்துரு (TACE-16) தரப்பாடு என்றும் தமிழக அரசு அறிவித்துள்ளது. எல்லாப் பயன்பாடுகளுக்கும் அனைத்து எழுத்துரு (TACE-16) தரப்பாடே சிறந்தது எனபதை பல்வேறு ஆய்வுகள் வெளிப்படுத்தியுள்ளன. இந்த அனைத்து எழுத்துரு (TACE-16) தரப்பாட்டின் பயன்பாடு கூடினால், ஒருங்குறி சேர்த்தியம் (Unicode consortium) ஒருங்குறிதளத்தில் 32-பிட்டு அமைப்பில் இந்த அனைத்து எழுத்துரு (TACE-16) தரப்பாட்டினைச் சேர்க்க இசைவளித்துள்ளது. எனவே, அனைத்து எழுத்துரு (TACE-16) தரப்பாட்டினைப் பல்வேறு பயன்பாடுகளில் செயல்படுத்தி அதன் பயன்பாட்டினைப் பெருக்குமாறு கணித்தமிழ் அன்பர்கள் அனைவரையும் அன்புடன் கேட்டுக் கொள்கின்றேன்.

இன்று ஜப்பான், கொரியா போன்ற நாடுகளில் செயல்படும் கணிப்பொறிகளுக்கு ஆங்கிலம் தெரியாது. ஜப்பான், கொரிய மொழிகளில் கொடுக்கப்படும் கட்டளைகளை மட்டுமே புரிந்நது கொண்டு செயல்படுகின்றன. உலக மொழிகளின் தாய்மொழியாகிய தமிழ்மொழிக் கட்டளைகளால் இயங்கும் கணிப்பொறியை வடிவமைக்க இயலாதா என்ன? அனைத்து எழுத்துரு (TACE-16) தரப்பாடு உருவாகும்வரை இயலாநிலை இருந்தது. ஆனால் இப்பொழுது அத்தடை இல்லை. இனி அப்படியொரு கணிப்பொறியைக் காண்பது எப்போழ்து? Assembler போன்ற அமைப்புச் செயல்மொழி (System Software) windows போன்ற இயக்க மென்பொருள் (Operating System) ஆகியவற்றை அனைத்து எழுத்துரு (TACE-16) தரப்பாட்டில் வடிவமைத்து ஒரு முழுமையான தமிழ்க்கணினியை படைக்கமுடியும். இப்படியொரு முழுமையான தமிழ்க்கணினியைப் படைத்து வழங்குபவருக்கு உருபா ஒரு இலக்கம் பரிசாக வழங்கப்படும் என்று மலேசியாவில் நடைபெற்ற 12-வது தமிழ் இணைய மாநாட்டின்

போழ்தே அறிவித்திருந்தேன். ஆனால் இன்றுவரை அது நிகழவில்லை. விரைவில் அப்படியொரு முழுமையான தமிழ்க்கணினி பயன்பாட்டிற்கு வரவேண்டும் என்பது எனது பேரவா? என் கனவு நிறைவேறுமா? இந்த மாநாட்டில் இது பற்றி முடிவெடுக்க கணிப்பொறி வல்லுநர்கள் முன்வர வண்டும் என்பது என் வேண்டுகோள்.

சென்னையில், அண்ணா பல்கலைக்கழகத்தின் ஆதரவோடு நிகழவுள்ள இந்த பதினெட்டாவது தமிழ் இணைய மாநாடு சிறப்பாக நடைபெற எனது உளங்கனிந்த வாழ்த்துக்களைத் தெரிவித்துக் கொள்கின்றேன்.

வாழ்க தமிழ்! வளர்க கணித்தமிழ்!

அன்புடன்,

மு.பொன்னவைக்கோ

REPUBLIC OF MAURITIUS

OFFICE OF THE VICE-PRESIDENT

MESSAGE

I am happy to note that INFITT is conducting its 19th Tamil Internet Conference in collaboration with Anna University, Chennai, India. I have been fully apprised of the tremendous research and achievements of the INFITT team. I am also very pleased that some of our scholars, teachers and students have participated in similar conferences conducted in Singapore, Malaysia, Germany, Philadelphia and other parts of the world. Through such fora, many young researchers and academics have been able to connect with one another and build mutual understanding of different cultures, especially through the learning of mother-tongue languages.

As technology continues to evolve, we must remain open to new possibilities and experiment with innovative practices. The Republic of Mauritius is no exception. In December 2018, the Prime Minister, Hon. Pravind Jugnauth, launched the Digital Mauritius 2030 Strategic Plan, and outlined his vision of Mauritius as a Digital Island.

True, Mauritius has emerged as a leader on the African Continent in ICTs, as shown by the favourable international rankings of the country. However, we are conscious of the need to double efforts for the digital economy to continue its expansion and be a stronger provider of jobs to our youth. Our students and teachers are equipped with state-of-the-art technologies in the classrooms, and the government is providing all the resources they need.

Our teachers have also adopted a wide range of innovative methodologies to engage students and foster a deeper appreciation of the Tamil Language and culture. This intersection of digital technologies and mother tongue languages makes language learning accessible while simultaneously providing a window into our roots and heritage.

At the international level, INFITT is a key platform for learning and sharing among participants who share a common goal of ensuring that our language remains relevant to future generations in the Tamil diaspora. Now into its 19th year, INFITT continues its fine tradition of holding the Tamil Internet Conference in cities around the world where Tamil-speaking communities reside.

I extend my warm thanks to INFITT for organising this important event, and I wish the organisers, participants and everyone a fruitful Tamil Internet Conference 2019.

Paramasivum Pillay Vyapoory, G.O.S.K.
Ag. President of the Republic of Mauritius

மாநாட்டு நிகழ்ச்சிகள் குழு அறிக்கை

முனைவர் வாசு அரங்கநாதன், பென்சில்வேனியாப் பல்கலைக்கழகம்

முனைவர் சோபா ல, AU-KBC, அண்ணா பல்கலைக்கழகம்

உத்தமம் நிறுவனத்தின் 18வது மாநாட்டு நிகழ்ச்சிகள் குழுவுக்குத் தலைமை ஏற்க வாய்ப்பளித்த உத்தமம் நிறுவனத்துக்கு நாங்கள் எங்களது நன்றியைத் தெரிவித்துக்கொள்கிறோம். எங்களது குழுவினர் பலர் மாநாட்டு அறிக்கை வெளியிட்ட நாள் தொடங்கி இம்மலரை உருவாக்கி முடிக்கும்வரை இம்மலரில் கட்டுரைகளும் மற்ற வாழ்த்துச் செய்திகளும் செவ்வனே வரவேண்டும் என்பதில் மிகவும் கவனமாக இருந்தோம். மாநாட்டுக்கு வந்த நாற்பத்தைந்து கட்டுரைகளில் தரமானவற்றையும் தமிழ்க்கணினிக்காக மிகவும் கடினமாக உழைத்து உருவாக்கிய கட்டுரைகளையும் தேர்ந்தெடுக்கப் பல முயற்சிகளை எடுத்தோம். மற்ற மாநாடுகள் போன்றல்லாது இம்மாநாட்டில் easychair.org என்னும் நிறுவனத்தின் இணைய நிரலியைக் கட்டுரைகளைப் பெறுவதிலிருந்து அவற்றைக் கணித்துத் தேர்ந்தெடுப்பது வரை பயன்படுத்தினோம். இது இம்மாநாட்டில் செயல்படுத்திய புது முயற்சியாகும். இந்நிரலி வழி ஆய்வாளர்களையும் நிகழ்ச்சிக் குழு உறுப்பினர்களையும் எளிதாகத் தொடர்புகொண்டு எங்களால் கட்டுரைகளைத் தேர்ந்தெடுக்க இயன்றது.

இம்மாநாட்டின் மையக்கருத்தான “தானியங்கிக் கருவிகளில் தமிழ்மொழிப் பயன்பாடு” என்னும் தலைப்பில் கட்டுரைகளை வரவேற்றோம். கணினி எங்ஙனம் மொழியை அறிந்து செயல்படுகிறது என்னும் ஆய்வை வளர்க்கும் முகத்தான் தொடர்ந்த இம்முயற்சியில் பல மாணவர்களும் ஆய்வாளர்களும் பங்குபெற்றனர். தமிழ் எழுத்துருவே கவனத்தில் இருந்த காலக்கட்டம் போய் இன்று கணினியும் மனிதர்களைப் போல தமிழ் மொழியைப் பயன்படுத்தும்

விதமறியும் ஆய்வில் ஆய்வாளர்கள் ஈடுபட்டிருப்பது கண்டு பெருமைப்படவேண்டும். தமிழின் வாக்கியங்களைக் கண்டறியும் முறை; தமிழின் பழமொழிகளைக் கண்டறியும் முறை; தமிழ்ச் சொற்களின் சொற்திருத்தி, ஒற்றுப்பிழை சரிபார்த்தல்; தமிழில் பயன்படுத்தும் கிரந்தப் பயன்பாட்டை நீக்கித் தமிழ்ச்சொற்களை மட்டுமே பயன்படுத்தும் முறை எனப் பல கட்டுரைகள் இம்மாநாட்டு மலரில் இடம்பெற்றுள்ளன. இம்மலரில் தொகுக்கப்பட்டுள்ள கட்டுரைகளில் பெரும்பாலான கட்டுரைகள் மாநாட்டு மையக்கருத்தான 'தானியங்கும் அடிப்படையில் கணினி வழித் தமிழ் மொழியைப் பயன்படுத்துவது' என்னும் கருத்தினடிப்படையிலேயே வந்துள்ளன என்பதை அறியும்போது தமிழ்க்கணினியின் பதினெட்டு ஆண்டுகால வளர்ச்சியின் உச்சத்தைக் காணமுடிகிறது.

உத்தமம் நிறுவனம் தொடர்ந்து தனது தமிழ்க்கணினி ஆய்வுப் பணியில் ஈடுபட்டு மேன்மேலும் பல தமிழிணைய மாநாடுகள் பலவற்றைக் காண நிகழ்ச்சிக்குழுவினர் சார்பாக எங்களது வாழ்த்துகளையும் பாராட்டுகளையும் தெரிவித்துக்கொள்கிறோம்.

Dr. Santhosh Babu, IAS

Principal Secretary, Information Technology
Chairman and Director
Tamil Virtual Academy, Chennai

கணினித் தமிழை மேலும், மேம்படுத்துவதற்கு ஏதுவாக தமிழ் இணையக் கல்விக் கழகத்தில் கணினித் தமிழுக்கென தனிப்பிரிவு மற்றும் கல்விக் காணொலிகளைத் தமிழாக்கம் செய்தல் போன்ற திட்டங்கள் வகுக்கப்பட்டுள்ளன. இக்கணினித் தமிழ்ச் சேவையைத் தொடருவதற்காகத் தமிழ் இணையக் கல்விக்கழகமும், அண்ணா பல்கலைக் கழகமும் மற்றும் உத்தமம் நிறுவனமும் இணைத்து TIC 2019 என்ற மாநாட்டினை நடத்துவதில் பெருமகிழ்ச்சி அடைகிறேன்.

தமிழ் இணையக் கல்விக் கழகம் மூலம், உலகளாவிய தமிழ்ச் சமுதாயத்தினர்க்கும், தமிழியலில் ஈடுபாடுள்ள மற்றையோர்க்கும் தமிழ்மொழி, இலக்கியம், பண்பாடு பற்றிய கல்விச் சாதனங்கள் மற்றும் பாடத் திட்டங்களை உருவாக்கி இணையம் வழியாக அளிக்கப்பட்டு வருகிறது.

மேலும், தமிழ் இணையக் கல்விக்கழகத்தால் "Linguistically Annotated Corpus for Tamil Literature" Syntactically and Semantically Annotated Corpus for Tamil Literature, An Electronic Dictionary with Pronunciation, Visual Thesaurus for Tamil. TANII திட்டத்தின் கீழ் Tamil Speech Computing Tools, Sentence Patterns Structures and Rules in Tamil, போன்ற திட்டங்களுடன் 15 தமிழ் மென்பொருள்கள் கணினித் தமிழுக்காக உருவாக்கப்பட்டுள்ளன.

மேற்படி பயனுள்ள பல தகவல்கள், மேலும் பல தமிழ் ஆர்வலர்களுக்கு சென்று சேருவதற்கு இந்த மாநாடு உறுதுணையாக இருக்கும் என நம்புகிறேன்.

Table of Contents

Semantics

Application

Syntax

Morphology

Corpus

Language Teaching

Machine Translation

Optical Character Recognition

Others

Tamil Paraphrase Detection using Long-Short Term Memory Networks

B. Senthil Kumar, D. Thenmozhi, Chandrabose Aravindan, S. Kayalvizhi
SSN College of Engineering
Chennai

theni_d@ssn.edu.in

Abstract. Paraphrase detection, one of challenging task in NLP is to detect whether the given pair of sentences which are rephrased or with word reordering are preserving the meaning semantically. Paraphrase detection for Indian languages especially for Tamil, one of the Dravidian language which is agglutinative is a challenging task. In this paper, we present the Paraphrase detection for Tamil language using Long-Short Term Memory (LSTM) neural networks. Eventhough the state-of-art systems used traditional learning techniques, it suffers from carefull hand-crafted feature engineering which require pure lexical, syntactic or lexico-syntactic features of the language. To alleviate this problem, deep LSTM is used to train our system which considers the pair of sentences and predicts it as a Paraphrase (P) or Non-paraphrase (NP). Our system performs 15.2% better than the existing system using deep neural networks.

1. Introduction

The ability to detect similar sentences written in natural language is crucial for several applications, such as text mining, text summarization, plagiarism detection, authorship authentication, query ranking and question answering. Paraphrase can be identified, generated or extracted. Paraphrase identification can be used in generation system to choose the best from the list of candidates generated by the paraphrase generation system. It also plays vital role in validating the paraphrase extraction system and machine translation systems. Detecting redundancy is a very important issue for a multi-document summarization system because two sentences from different documents may convey the same semantic content and to make summary more informative, the redundant sentences should not be selected in the summary.

In this paper, the focus is to identify the paraphrase sentences from the DPIL corpus for Tamil language. The shared task on Detecting Paraphrases in Indian Languages (DPIL)@FIRE 2016 [1][2] was a good effort towards creating benchmark data for paraphrases in Indian Languages. Identifying the paraphrases in Indian languages especially for Tamil is a difficult task because evaluating the semantic similarity of the underlying content and understanding the morphological variations of the language are more critical.

Our main contributions in this work are:

(i) We propose a deep neural architecture for Tamil paraphrase detection using LSTM enhanced with attention mechanisms.

(ii) We evaluate this model in a monolingual setting, performing paraphrase detection on standard DPIL datasets, to assess the suitability of this model type for the paraphrasing task.

(iii) We compare the performance of our model to the state-of-the-art Tamil paraphrase detection model using deep learning, including the detailed analysis.

2. Related Work

Out of ten teams who submitted results in the DPIL shared task, five teams had submitted their results for Tamil paraphrase detection. Kong et al., [8] submitted results for all the four Indian languages – namely

Tamil, Malayalam, Hindi and Punjabi. They have used Cosine Distance, Jaccard Coefficient, Dice Distance and METEOR features and classification is done based on Gradient Tree Boosting. They achieved the overall best score across all the four languages. The Tanik et al., [9] used similarity based features, word overlapping features and scores from the machine translation evaluation metrics to find out the similarity scores between pair of sentences. They tried with three different classifiers namely Naïve Bayes, SVM and SMO.

The Tamil Shallow parser was used by Thangarajan et al., [10] to extract the morphological features of language and applied Support Vector Machine (SVM) and Maximum Entropy to classify Tamil paraphrases. They submitted results only for Tamil language with 82% accuracy. Kamal Sarkar [11] had submitted results for all the four languages. He used different lexical and semantic level (Word embeddings) similarity measures for computing features and used multinomial logistic regression model as a classifier. His model performed 78% accuracy for the Tamil language. Sarkar et al., [12] used the features based on Jaccard Similarity, length normalized Edit Distance and Cosine similarity. These feature-set are trained using Probabilistic Neural Network (PNN) to detect the paraphrases. For Tamil language, the system achieved 83.33% accuracy in subtask1.

All the above models applied the traditional learning techniques by using the lexical, syntactic or semantic similarity feature sets. Mahalakshmi et al., [3] used recursive auto-encoders (RAE) to represent the feature vectors for Tamil paraphrase detection. Initially the sentence pairs are parsed using the shallow parser and its word, phrase vectors are computed by RAE for learning feature vectors for phrases in syntactic trees in an unsupervised way. The model then used the Euclidean distances to measure the similarity and then softmax classifier is used to detect the paraphrases.

3. Dataset

We have used the Tamil paraphrase pair of sentences from the DPIL@FIRE2016 shared task. The shared task required participants to identify sentential paraphrases in four Indian languages – Hindi, Punjabi, Malayalam and Tamil. In this shared task, there were two sub-tasks: task1 is to classify a given pair of sentences as paraphrases (P) or not paraphrases (NP) and task2 is to identify whether a given pair of sentences are completely paraphrases (P) or semi-paraphrases (SP) or not paraphrases (NP). The evaluation dataset is mainly obtained from the newspaper. The details of this corpus can be found in http://nlp.amrita.edu/dpil_cen/. The corpora are divided into two different subsets. We used Task-1 subset for Tamil language, which categorizes 2500 paraphrase Tamil sentence pairs into one of binary class and with 900 test pairs.

The corpus statistics showed that the vocabulary sizes for Hindi & Punjabi languages are less than Tamil and Malayalam. This is because the Dravidian languages, Tamil and Malayalam are agglutinative in nature. Due to this phenomenon, Dravidian languages end up by having more unique words and hence larger vocabulary. The size of vocabulary for Tamil is around 17K for 2500 pairs of train sentences. This agglutinative phenomenon of Tamil language increases the complexity in detecting the paraphrases.

4. Proposed Methodology

To detect the paraphrases, we adopted the NMT architecture [4]. The model consists of embedding layer, encoders, decoders which uses Bi-LSTM layers and attention mechanism on top of encoders, projection layer on top of decoders to predict the class label. We treated this model as classification system to predict the class label as Paraphrase (P) or Non-Paraphrase (NP) for the given pair of input sentences. The embedding layer creates the vocabulary for both the input and the output. Here the input is the pair of sentences and the output is the binary class label P and NP. The dataset annotated the paraphrases using XML format for each language. The following is the sample paraphrase in Tamil:

<Paraphrase pID="TAM0001">

<Sentence1> சங்கராபுரம் தொகுதியில் போட்டியிடும் ஸ்டாலின் நடைபயணமாக சென்று பிரசாரம் செய்தார். </Sentence1>

<Sentence2> தி.மு.க., வேட்பாளர் ஸ்டாலின் போட்டியிடும் சங்கராபுரம் தொகுதியில் சின்ன சேலம் பகுதியில் நடைபயணமாக சென்று ஓட்டு சேகரித்தார். </Sentence2>

<Class> P </Class>

</Paraphrase>

Each paraphrase was assigned a unique ID followed by the two sentences marked by sentence number tags and the gold class label tag. This input pair of sentences are extracted from the dataset, preprocessed and presented as input sequences to encoder in the model as below:

<s> சங்கராபுரம் தொகுதியில் போட்டியிடும் ஸ்டாலின் நடைபயணமாக சென்று பிரசாரம் செய்தார். <eol> தி.மு.க., வேட்பாளர் ஸ்டாலின் போட்டியிடும் சங்கராபுரம் தொகுதியில் சின்ன சேலம் பகுதியில் நடைபயணமாக சென்று ஓட்டு சேகரித்தார் .</s>

The input pair of sentences are delimited with <eol> which acts as a boundary marker between the sentences pair and given as input to the Bi-LSTM units. The time-based Bi-LSTM units which act as an encoder generates a context vector for the given input pair of sentences which is then mapped to the corresponding class in output during the training. The attention mechanism on top of encoder Bi-LSTM units calculates the weights which select the set of inputs that contribute in predicting the output class label. During the testing, the unseen paraphrase sentences are given as input to the model. The encoding Bi-LSTM units that generate the context vector is given as input to decoder and act as an initializer to the decoding Bi-LSTM units. The context vector, the previous output and current input are given as input to the decoder Bi-LSTM units to predict the class label. From our earlier experiments, we resorted to use one Bi-LSTM layer as encoder and decoder. The performance of the system varies with the number of layers of Bi-LSTM units, number of epochs, type of attention mechanisms, the data size which generated the vocabulary and other hyper parameters to the model.

5. Result

To evaluate the system, we have considered 5-fold cross validation on the training data. We tried with two types of attention mechanisms – Normed Bahdanau (NB) and Scaled Luong (SL) on top of encoders. For the given 2500 pair of training sentences, the model accuracy is measured by dividing the training dataset into 5 folds. For each fold 500 pairs are considered for testing and the remaining 2000 pairs are considered as training data. The overall performance of the system for 5 folds is 65.2% accuracy and is shown in Table 1.

Table 1. Performance of Models

Sl No	n-Fold	Model Accuracy(%)	
		NB	SL
1	1-Fold	60.8	67.0
2	2-Fold	64.4	62.0

3	3-Fold	68.6	68.4
4	4-Fold	63.0	63.6
5	5-Fold	68.2	65.0
	Overall	65.0	**65.2**

It is observed from the results that both the two models – NB and SL – performed almost similar. We have compared the results of Mahalakshmi et al., [3] who have reported on deep learning approach for paraphrase detection in Tamil. They have also considered 2000 pairs for training and 500 pairs for evaluating the performance. The result comparison of our approach with the existing approach related to deep learning method for Tamil paraphrase is shown in Table 2.

Table 2. Performance Comparison

Methodology	**Accuracy (%)**
	Overall
Mahalakshmi et al., [3]	50
Our method - NB	65
Our method - SL	**65.2**

The major focus of work by Mahalakshmi et al., is to detect the Tamil Paraphrases correctly and they reported 65.17% of accuracy in detecting paraphrases and 34.83% for detecting non-paraphrases. On overall the accuracy of the system [3] is 50%. It is observed from Table 2 that our two approaches have improved the overall accuracy by 15.2%.

5.1. Detailed Analysis

Here is the deep analysis of our model. The confusion matrix for both of our approaches NB and SL are given in Table 3 and Table 4 respectively.

Table 3. Confusion matrix (NB)

Folds / Evaluation	**Fold 1**		**Fold 2**		**Fold 3**		**Fold 4**		**Fold 5**	
	P	**NP**	**P**	**NP**	**P**	**NP**	**P**	**NP**	**P**	**NP**
P	119	131	129	121	**150**	100	130	120	149	101
NP	65	185	57	193	57	**193**	65	185	58	192

Table 4. Confusion matrix (SL)

Folds / Evaluation	**Fold 1**		**Fold 2**		**Fold 3**		**Fold 4**		**Fold 5**	
	P	**NP**	**P**	**NP**	**P**	**NP**	**P**	**NP**	**P**	**NP**

P	127	123	132	118	**141**	109	145	105	148	102
NP	42	208	72	178	49	**201**	77	173	73	177

From Table 3 and Table 4, it is obvious that the accuracy of fold 3 is maximum in NB and SL systems. The average accuracy is slightly better in SL system because of an improvement in precision by 0.25 and recall by 1.28 when compared with NB system as shown in Table 5 and Table 6.

Table 5. Performance of NB

Folds/Metrics	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Avg**
Precision (%)	64.67	69.35	72.46	66.67	71.9	69.01
Recall (%)	47.6	51.6	60	52	59.6	54.16
F1 (%)	54.84	59.17	65.65	58.43	65.2	60.66
Accuracy (%)	60.8	64.4	68.6	63	68.2	65

Table 6. Performance of SL

Metrics/Folds	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Avg**
Precision (%)	75.15	64.71	74.21	65.32	66.9	69.26
Recall (%)	50.8	52.8	56.4	58	59.2	55.44
F1 (%)	60.62	58.15	64.09	61.4	62.8	61.41
Accuracy (%)	67	62	68.4	63.6	65	65.2

The system with scaled-luong attention mechanism (SL) performed slightly better than normed-bahdanau attention system (NB).

Several research works have reported on paraphrase detection for Indian languages including Tamil. However they have used traditional learning techniques with lexical, syntactic and lexico-syntactic and word embedding features to develop their systems on DPIL@FIRE2016 dataset. The shared task reported that the traditional learning techniques yield 83.33% of accuracy for Tamil language, whereas our deep learning approach gives 56.4% & 49% of accuracies on test data for NB and SL models respectively. This is due to the limitations in the size of the corpus in Tamil.

5.2. Error Analysis

The models suffer from the data sparseness problems because specific paraphrase instances occur only a handful of times in the training set. Consider the following instance from the test data:

Sentence number: 11

இந்திய விடுதலைப்போராட்டம் 1857ல் வேலூர்ப்புரட்சியின் போதே ஆரம்பித்துவிட்டது eol 1857ல் நடைபெற்ற வேலூர்ப்புரட்சியிலிருந்தே இந்திய விடுதலைப்போராட்டமானது உயிர்கொள்ள ஆரம்பமானது

The above is classified as P (Paraphrase) in gold target, whereas our systems (both NB and SL) predicted it as NP (Non paraphrase). Because NMT learns word representations in continuous space, it tends to translate (map) the words that are frequent in context [13]. Most of the words like விடுதலைப்போராட்டம், வேலூர்ப்புரட்சி, ஆரம்பித்துவிட்டது, உயிர்கொள்ள, ஆரம்பமானது, etc., in the test instance 11 are of type UNK where there is not even a single occurrence in the training data. Whereas the word இந்திய has occurred 52 times in P, 125 times in NP sentences in training data. Similarly another word நடைபெற்ற has its appearance of 28 times in P and 44 in NP sentences. Both these words appeared more frequently in the context of NP sentences rather than P sentences of the training data.

6. Conclusion

We used the LSTM-based deep neural network model to detect the Tamil paraphrase from the DPIL corpus. We evaluated our system for 5 folds on the given training data of 2500 instances. We developed two variations with respect to attention techniques on the deep neural network – system using scaled-luong (SL), normed bahdanau (NB) as attention mechanisms. Among these, SL showed the overall accuracy of 65.2% on the training data of the DPIL corpus for Tamil paraphrase detection. Eventhough the state-of-the-art systems for Tamil paraphrase detection have used the traditional learning techniques, it suffers from heavy hand-crafted feature engineering. Whereas deep neural network systems alleviate this need of lexico-syntactic features. The performance of this system can be improved further with more number of training instances. Since the deep neural network systems require more data to be trained, the DPIL training instances for Tamil paraphrase – 2500 – was not enough for the deep neural network model to capture and learn the syntactic feature of the language automatically from the given instances.

References

1. Anand Kumar, M., Singh, S., Kavirajan, B., Soman, K.P., DPIL@FIRE 2016: Overview of shared task on detecting paraphrases in Indian Languages (DPIL), CEUR Workshop Proceedings, 1737, pp. 233-238, 2016.
2. Anand Kumar M., Singh S., Kavirajan B., Soman K.P., Shared Task on Detecting Paraphrases in Indian Languages (DPIL): An Overview. In: Majumder P., Mitra M., Mehta P., Sankhavara J. (eds) Text Processing. FIRE 2016. Lecture Notes in Computer Science, vol 10478. Springer, 2016.
3. Mahalakshmi, S., M. Anand Kumar, and K. P. Soman., Paraphrase detection for Tamil language using deep learning algorithm, International Journal of Applied Engineeing Reseseach, Volume 10, no. 17, pp: 13929-13934, 2015.
4. Minh-Thang Luong, Eugene Brevdo, Rui Zhao, Neural Machine Translation (seq2seq) Tutorial, https://github.com/tensorflow/nmt, 2017.
5. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, Neural machine translation by jointly learning to align and translate, ICLR, 2015.
6. Minh-Thang Luong, Hieu Pham, and Christopher D Manning, Effective approaches to attention-based neural machine translation. EMNLP, 2015.
7. Ilya Sutskever, Oriol Vinyals, and Quoc V. Le., Sequence to sequence learning with neural networks, NIPS, 2014.
8. Kong, Leilei, Kaisheng Chen, Liuyang Tian, Zhenyuan Hao, Zhongyuan Han, and Haoliang Qi. HIT2016@ DPIL-FIRE2016: Detecting Paraphrases in Indian Languages based on Gradient Tree Boosting, In FIRE (Working Notes), pp. 260-265, 2016.

9. Saikh, Tanik, Sudip Kumar Naskar, and Sivaji Bandyopadhyay, JU_NLP@ DPIL-FIRE2016: Paraphrase Detection in Indian Languages-A Machine Learning Approach, In FIRE (Working Notes), pp. 275-278, 2016.
10. Thangarajan, R., S. V. Kogilavani, A. Karthic, and S. Jawahar, KEC@ DPIL-FIRE2016: detection of paraphrases on Indian languages, In FIRE (Working Notes), pp. 282-288, 2016.
11. Sarkar, Kamal. KS_JU@ DPIL-FIRE2016: detecting paraphrases in Indian languages using multinomial logistic regression model, In FIRE (Working Notes), pp. 250-255, 2016.
12. Sarkar, Sandip, Saurav Saha, Jereemi Bentham, Partha Pakray, Dipankar Das, and Alexander F. Gelbukh, NLP-NITMZ@ DPIL-FIRE2016: Language Independent Paraphrases Detection, In FIRE (Working Notes), pp. 256-259, 2016.
13. Nguyen, Toan and Chiang, David, Improving Lexical Choice in Neural Machine Translation, In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1, ACL, pp. 334-343, 2018.

Automatic Generation of Description for Tamil Proverbs

R.Anita, C.N. Subalalitha
Department of Computer Science and Engineering,
SRM Institute of Science and Technology, Kattankulathur-603 203
anitaramalingam17, subalalitha{@gmail.com}

Abstract. The Tamil proverbs are treasures of Tamil language and they have been used over the years. The proverbs are crisp and easy way to remember advices, awareness and knowledge. The current Tamil generation mostly are unaware of the cultural values of proverbs. Also, many Tamil proverbs are misunderstood by the global society and the explanation is not known for many Tamil proverbs. The proposed research takes an attempt to automatic generation of meaning for the Tamil proverbs using sentence scoring approach. This work can be useful for many Natural Language Processing applications, such as Summary Generation System, Question Answering System, and Information Retrieval System. The proposed work is evaluated using the metrics namely, precision, recall and F-measures and has achieved 91% of precision, 80% of recall and 85% of F-Score.

Keywords: Sentence Scoring, Meaning Generator, Tamil proverb.

1 Introduction

The Tamil language has many valuable proverbs, which are passed on from one generation to other. The current Tamil generation is unaware of many of these proverbs and even if aware of few of them, they are misinterpreted and are used in the wrong contexts. Explanation of many proverbs are not available on the World Wide Web (WWW) but many have been used in stories and essays. The correct description and the meaning of these proverbs need to be automatically extracted from the contexts in which the proverbs are uttered. This will help making the global Tamil society aware of the proverbs. One of the ways to reach the current Tamil generation is to make them available on the WWW. This work lays such a foundation for automatic generation of meaning of the proverbs using sentence scoring approach. The contributions of this research is twofold:

1. Data set creation
2. Retrieving description from the context of the text using sentence scoring method.

The structure of the paper is organized as follows: Section 2 presents the literature survey. Section 3 explains the proposed work. The result and discussions are given in Section 4. Section 5 explains about the conclusion.

2. Related Work

The literature survey has been done on the sentence scoring approach, summarization and selecting sentences from the context, since the proposed work relies on these approaches.
Deoras et. al. (2011) proposed re-scoring strategy for capturing longer distance dependencies on English texts. Recurrent Neural Network language model was used. English Broadcast News (BN) corpus was used as a dataset.

Li et. al. (2012) proposed graph based sentence ranking method for summarization on English texts. They used TAC 2008 and 2009 benchmark data sets. They evaluated their work with Recall-Oriented Understudy for Gisting Evaluation (ROUGE).

Ferreira et. al. (2013) used sentence scoring method for extractive text summarization on English texts. They performed assessment of 15 algorithms. They used three datasets, namely, CNN, Blog summarization and SUMMAC. They used ROUGE for evaluation.

Mikolov et. al. (2013) proposed efficient estimation of word representations in vector space on English texts. They used Feedforward Neural Net language model and Recurrent Neural Net language model for their work. They used Google News corpus for training. They evaluated their work with accuracy. Vural et. al. (2013) proposed unsupervised sentimental analysis of movie reviews in Turkish language. They obtained data from a movie website, Beyazperde. They evaluated their work with accuracy.

Ferreira et. al. (2014) proposed multi-document summarization using sentence clustering algorithm on English texts. They used DUC 2002 as a dataset for testing performance of their system. They calculated F-measure as evaluation metric.

Mesnil et. al. (2014) proposed sentimental analysis of movie reviews in English. They compared with several machine learning algorithms. IMDB movie reviews was used as a dataset. Accuracy was used as evaluation metric.

Van et. al. (2014) used data merging techniques for multi-document summarization on English texts. They used DUC 2002 and DUC 2005 as datasets. They evaluated their work with precision, recall and F-measure.

Babar et.al. (2015) proposed extractive text summarization using Fuzzy logic Extraction approach and Latent Semantic Analysis. They used ten different datasets. Precision, recall and F-measure were used as evaluation metrics.

Ren et.al. (2017) proposed neural network model called contextual relation based summarization for extractive text summarization on English texts. They used six datasets, namely, DUC 2001, DUC 2002, DUC 2004, DUC 2005, DUC 2006, and DUC 2007. They used ROUGE for evaluation.

It can be observed that the sentence scoring was used in expository (essay type) English and Turkish texts. The English text follows SVO (Subject-Verb-Object) pattern. The proposed work processes Tamil texts along with the proverbs. Tamil expository texts follow both SVO and SOV (Subject-Object-Verb) pattern. Tamil proverbs on the other side neither follow SVO nor follow SOV pattern. Tamil is a free word order language and also Tamil texts are relatively rich in morphological variants. These challenges make processing of Tamil proverbs more difficult than processing the expository texts. The proposed work generate the description of Tamil proverbs automatically from the contexts in the text.

3. Proposed Work

The architecture diagram for the proposed work is shown in Figure 1. The Tamil proverb is given as the input to the Proverb Meaning Generator. If the description of the proverb is available then it is given to the output module directly. When the proverb is used in the middle of the texts, then the context analysis needs to be done to generate its description. The preceding five sentences and the succeeding five sentences of the proverb are taken as the context. Each sentence in the context is given a score and the closer sentences are chosen for generating description. Finally, the description is generated using templates.

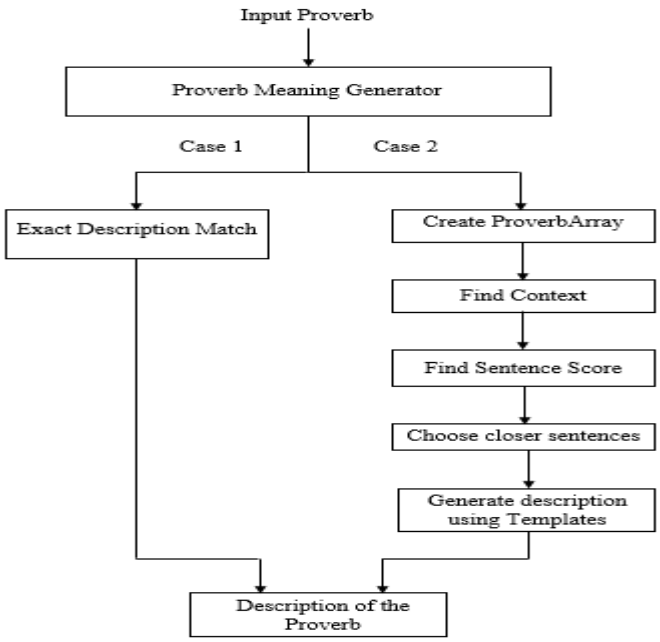


Figure 1. Architecture Diagram

The meaning of all the words of the input proverb is found using the Tamil Agarathi (Dictionary). All words of the proverb along with their meaning are stored in an array called, Proverb Array.

3.1. Finding Context

The proverb may appear anywhere in a text. The meaning of the proverb are assumed to be present in the preceding and succeeding sentences of the proverb. Five sentences preceding the proverb and five sentences succeeding the proverb are taken as the context window which is used to generate the meaning for the proverbs.

3.2. Sentence Score

A sentence score is calculated for all the sentences selected in the context window for a proverb. The words of the selected sentences are assigned with a weight. The weight is assigned to each word in the sentences based on its occurrence in the Proverb Array. The sentence score is the summation of all the weights of the words.

3.3. Choosing the high weighted Sentences

The sentences whose scores cross the threshold are chosen for meaning description generation. The threshold is fixed as 3 currently by analyzing 200 proverbs and their contexts manually.

3.4. Generate Description for the Proverb

The description of the proverb is generated, using fixed templates (Subalalitha et. al. 2011). The template is a collection of sentences comprising, proverb followed by the sentences sorted as per their scores.

Example 1 is given in Figure 2. The proverb in example 1 is exactly matching with the description. It is retrieved as the description of the proverb.

Example:1

Proverb in Tamil
உப்பிட்டவரை உள்ளளவும் நினை.

Transliteration
Uppittavarai ullalavum ninai.

Tamil Meaning
உனக்கு உதவி செய்தவரை என்றும் மறவாதே.

Figure 2. Example1

Example 2 is given in Figure 3. The proverb P1 in example 2 is not having Tamil explanation directly. It is the subpart of description of another proverb P2. The explanation for P1 is retrieved from the description of P2. Since the sentence "ஒவ்வொரு எலியும் தனக்குத் தேவையானதை விட ஐந்து மடங்கு தானியத்தை அள்ளிக் கொண்டு செல்லுமாம்" has the high score, the Proverb Meaning Generator generates this sentence as the description of P1.

Example 2: P1

Proverb in Tamil
அறுப்புக் காலத்தில் எலிக்கு ஐந்து பெண்சாதி.

Transliteration

Aṟuppuk kālattil elikku aintu peṇcāti.

P2:

Proverb in Tamil

ஆனைக்கு ஒரு காலம் வந்தால் பூனைக்கும் ஒரு காலம் வரும்

Transliteration

Āṉaikku oru kālam vantāl pūṉaikkum oru kālam varum

Tamil Meaning

யானைகளைக் கொண்டு கதிரடித்த பின்னர், பிரிந்த நெல்மணிகளைத் தனியாக எடுத்துத் தானியக் கிடங்குகளில் சேமித்து வைத்திருப்பர். இப்போது தான் சிக்கல் துவங்குகிறது. சேமித்து வைக்கப்பட்டுள்ள இந்த நெல்மணிகளைக் குறிவைத்து எலிகளின் பட்டாளம் படையெடுக்கத் துவங்குகின்றது. ஒவ்வொரு எலியும் தனக்குத் தேவையானதை விட ஐந்து

மடங்கு தானியத்தை அள்ளிக் கொண்டு செல்லுமாம். இதனடிப்படையில் தான்,

"அறுப்புக் காலத்தில் எலிக்கு ஐந்து பெண்சாதி"

என்ற பழமொழியே எழுந்தது எனலாம். கூட்டம் கூட்டமாக இவை புகுந்து நெல்மணிகளைத் திருடிக் கொண்டு செல்வதை அப்படியே விட்டுவிட்டால், கிடங்கையே காலி செய்து விடும். எனவே இந்த எலிகளைக் கூண்டோடு ஒழிக்க, பூனைகளைக் கொண்டுவந்து தானியக் கிடங்குக்குள் வைப்பர். பூனைகளும் கிடங்கிற்குள் வருகின்ற எலிகளை வேட்டையாடித் தின்றுவிடும்.

இவ்வாறாக கதிரடிக்கும் காலத்தில் யானையின் உதவியும் சேமிக்கும் காலத்தில் பூனையின் உதவியும் மனிதருக்குத் தேவைப்பட்டது.

Figure 3: Example2

Algorithm for the proposed work is given below.

Input: Tamil proverbs

Output: Explanation of proverbs in Tamil

1. Get proverb from user, P1
2. Match with the dataset
3. If P1 is exactly matched

   retrieve the explanation as the description of P1

4. Else if P1 is the subpart of another text
   - 4.1. Find meaning of words in P1 from Tamil Agarathi (Dictionary) and form ProverbArray
   - 4.2. Let P as sentence i, S_i
   - 4.3. Select sentences from contexts, S_{i-5}, S_{i-4}, S_{i-3},S_{i-2},S_{i-1} and S_{i+1},S_{i+2}, S_{i+3}, S_{i+4}, S_{i+5}
   - 4.4. Find the sentence score
   - 4.5. Choose the sentences crossing the threshold
   - 4.6. Generate description using the template

4. Result and Discussion

The proposed method is tested with 1000 Tamil proverbs. The explanation for 600 Tamil proverbs are given by the Proverb Meaning Generator directly. 400 Tamil proverbs are used in the description of other texts. The Proverb Meaning Generator generates description using sentence scoring method. The precision, recall and F-measure for those proverbs are calculated as shown in Table 1.

Proverbs (N)	**Correctly Retrieved (C)**	**Retrieved (M)**	**Precision P=C/M**	**Recall R=C/N**	**F-Measure** $F=\frac{2PR}{P+R}$
400	320	350	91.4	80	85.3

The reasons for the drop in precision are, the explanation for Tamil proverbs are very long. Also, the context window size chosen is five. The relevant explanation for the current proverb may have appeared outside the context window. The correct description can never be matched if there are no closer sentences found.

5. Conclusion

The Tamil proverbs were spoken by Tamil ancestors based on their experiences. They are transferred from one generation to another generation orally. The proverbs may guide the current Tamil generation. The Tamil proverbs have so many nuggets hidden which need to be explored for the goodness of the society. One of the ways to explore it to make them online. This research lays such a foundation for obtaining automatic description of the unknown Tamil proverbs.
This work can be further used by many Natural Language Processing applications, such as, Information Retrieval System, Summary Generation System, and Question Answering System.

References

1. Babar, S. A., and Pallavi D. Patil. "Improving performance of text summarization." Procedia Computer Science 46 (2015): 354-363.
2. Deoras, Anoop, Tomáš Mikolov, and Kenneth Church. "A fast re-scoring strategy to capture long-distance dependencies." In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1116-1127. Association for Computational Linguistics, 2011.
3. Ferreira, Rafael, Luciano de Souza Cabral, Frederico Freitas, Rafael Dueire Lins, Gabriel de França Silva, Steven J. Simske, and Luciano Favaro. "A multi-document summarization system based on statistics and linguistic treatment." Expert Systems with Applications 41, no. 13 (2014): 5780-5787.
4. Ferreira, Rafael, Luciano de Souza Cabral, Rafael Dueire Lins, Gabriel Pereira e Silva, Fred Freitas, George DC Cavalcanti, Rinaldo Lima, Steven J. Simske, and Luciano Favaro. "Assessing sentence scoring techniques for extractive text summarization." Expert systems with applications 40, no. 14 (2013): 5755-5764.
5. Li, Xuan, Liang Du, and Yi-Dong Shen. "Update summarization via graph-based sentence ranking." IEEE transactions on Knowledge and Data Engineering 25, no. 5 (2012): 1162-1174.

6. Mesnil, Grégoire, Tomas Mikolov, Marc'Aurelio Ranzato, and Yoshua Bengio. "Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews." arXiv preprint arXiv:1412.5335 (2014).
7. Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).
8. Ren, Pengjie, Zhumin Chen, Zhaochun Ren, Furu Wei, Jun Ma, and Maarten de Rijke. "Leveraging contextual sentence relations for extractive summarization using a neural attention model." In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 95-104. ACM, 2017.
9. Subalalitha CN, Umamaheswari E, Geetha TV, Parthasarathi R, Karky M. Template based multilingual summary generation.
Tamil Computing Lab (TaCoLa) College of Engineering Guindy Anna University, Chennai. 2011.
10. Van Britsom, Daan, Antoon Bronselaer, and Guy De Tre. "Using data merging techniques for generating multidocument
summarizations." IEEE Transactions on Fuzzy Systems 23, no. 3 (2014): 576-592.
11. Vural, A. Gural, B. Barla Cambazoglu, Pinar Senkul, and Z. Ozge Tokgoz. "A framework for sentiment analysis in turkish: Application to polarity detection of movie reviews in turkish." In Computer and Information Sciences III, pp. 437-445. Springer, London, 2013.

UNL DECONVERSION FOR SENTENCE REALIZATION IN TAMIL

D. Josephine Sylvia[1], S. Lakshmana Pandian[2]
[1,2]Dept. of Computer Science and Engineering
Pondicherry Engineering College
Puducherry, India
Josephinesylvia07@gmail.com, lpandian72@pec.edu

Abstract. — Natural language processing (NLP) is an area of computer science and artificial intelligence concerned with the interactions between computers and human (natural) languages. Universal Networking Language (UNL) is an intermediate representation for Interlingua based machine translation. The Universal Networking Language (UNL) is an electronic language in the form of a semantic network for computers to express and exchange every kind of information. The UNL approach is a hybrid approach of the rule and knowledge-based approaches to machine translation. The Deconverter is a Language Independent Generator (LIG), which provides a Framework for Syntactic and Morphological generation of Native Language. In this proposed work, Deconvertor can convert UNL expressions into native language (Tamil), using a language specific set of Word Dictionary, Grammatical Rules and Co occurrence Dictionary by ensuring that the meaning of the input sentence is preserved.

Keywords: Natural language processing (NLP), Universal Networking Language (UNL), Morphological generation, Deconvertor

1 Introduction

Natural Language Processing (NLP) is a field of computer science, artificial intelligence and linguistics and is an area of research and application that analyze how computers are used for understanding and manipulating natural language text or speech to achieve the desired tasks. Natural Language Processing (NLP) is a sub-field of Artificial Intelligence that is focused on enabling computers to understand and process human languages, to get computers closer to a human-level understanding of language. Natural language processing (NLP) aims to design and build software that will analyze, understand and generate languages that humans use naturally [1]. Machine translation is a very important application in natural language processing.

In order to overcome the language barrier, many attempts have been made in the past. Professional translators have been bridging such a communication gap. The quantity of translation by human however is rather small as compared to the required communication needed for the different languages. The main reason for such a limitation is high cost involved in the translation work. In addition, the number of translators for minor language is rather small[2]. Translation made by human being, thus has its limitation in terms of cost and human resources. Computer translation systems have made significant progress. Some of them are now being incorporated in network browsers.

The two demands for these systems indicate how large the language problem is among Internet users. Computer translation systems are useful under limited conditions. For instance, the user can evaluate and modify a translated document in his own language, but seldom in the other language. However, after translating with a computer, the user has to work to edit the output document. In addition it would require language knowledge to edit the translation of the document in the other language. In sending information throughout the world, the sender normally does not know the language of the recipient. In this case, the sender is bound to use a computer translation system blindly, because the user cannot check whether the translated results are correct or not. This is a serious limitation in current computer translation system.

The Universal Networking Language (UNL) is an electronic language for describing, summarizing, refining, storing information in a machine and natural-language-independent form. UNL, as a language for expressing information and knowledge described in natural languages, has all the components corresponding to that of a natural language [5]. UNL represents sentences in the form of logical expressions, without ambiguity. These expressions are not for humans to read, but for computers. Thus, UNL is an intermediate language to be used through the Internet, which allows communication among people of different languages using their mother tongue.

2. Related Work

Rajeswari Sridhar et al.[1] proposed English to Tamil machine translation system using universal networking language. The system aims at translating a given English sentence to a Tamil sentence, which conveys the meaning of the input, and ensures a grammatically correct sentence as the output. This system was evaluated using bilingual evaluation understudy (BLEU) score. The system is simple and efficient of using UNL for translation and as the result the system is easily scalable. Phan Thi Le Thuyen et al.[2] proposed Automatic translation for Vietnamese based on unl language. A tool is introduced based on UNL application and reused for the process of encoding a Vietnamese sentence into UNL expression and decoding an UNL expression into Vietnamese. Tools used for EnConverter are IAN tool (Interactive Analyzer)and EnCo tool. Tools used for DeConverter are EUGENE tool and DeCo tool. The converter tools from natural language to UNL are effective and the quality is pretty good and acceptable.

Imane Taghablout et al.[3] designed Amazigh verb in the Universal Networking Language. The system focus on presenting inflectional paradigms, and the lexical mapping stage needed for building an "Amazigh dictionary" for the verbal category according to the UNL specifications. It aims at eliminating linguistic barriers and, furthermore, promoting the access to information in the autochthone languages. Aloke Kumar Saha et al.[4] designed Design and Implementation of an Efficient DeConverter for generating Bangla Sentences from UNL Expression. The system focuses on the Linguistic Analysis of Bangla Language for the DeConversion process. A set of DeConversion rules have been burgeoned for converting UNL expression to Bangla. The system is tested with more than 2000 UNL Expressions. The System achieved accuracy as 89%, which can be marked outstanding in this Field of Study.

Biji Nair et al.[5] proposed Language Dependent Featues for UNL-Malayalam Deconversion. The deconverting generator for Malayalam language is done using Universal Networking Language (UNL) for Machine Translation. The deconversion is tested against 100 Malayalam Sentences that has achieved an appreciable F-measure score of 0.978. The system is efficient in generating syntactically unambiguous semantically equivalent target sentence for the UNL source sentences. Ananthi Sheshasaaye et al.[6] tackled The Role of Morphological Analyzer and generator for Tamil language in Machine Translation Systems. Statistical machine translation plays a predominant role in machine translation of larger vocabulary tasks. Achieving this goal is not an easy task especially when it comes to languages like Tamil which are agglutinative in nature. Deep analysis is needed at the word level to confine the correct meaning of the word from its morphemes and categories. The computational implementation of analysing natural language is done by Morphological analyzer. This formed a ground work for better understanding of various approaches that are used to develop morphological analyzer and generator of Tamil languages.

S. Lushanthan et al.[7] proposed Morphological Analyzer and Generator for Tamil Language. The system illustrates how the lexicon and the orthographic rules of the Tamil language have been written as regular expression using finite state operations. This model is built using Xerox toolkit which uses “two level morphology”. The model may produce several results or forms in the look down process for nouns. This model uses a common fully automated transliteration scheme and implemented. Jisha P.Jayan et al.[8] proposed Morphological Analyser and Morphological Generator for Malayalam. Morphological Analyzer is a program for analyzing the morphology of an input word and the analyzer reads the inflected surface form of each word in a text and provides its lexical form while Generation is the inverse process. Both Analysis and Generation make use of lexicon. The suffix stripping method has been used for developing the morphological analyser and for developing the morphological generator the suffix joining method has been used.

Velliangiri Dhanalakshmi et al.[9] proposed Grammar Teaching Tools for Tamil language. The tools like Parts of speech Tagger, Chunker and Dependency parser for the sentence level analysis and Morphological Analyzer and Generator for the word level analysis were developed using machine learning based technology. The tool is developed for Malayalam, Kannada and Telugu languages. It also improves vocabulary, improve writing and reading skills and also speed up the learning of second language. T.Dhanabalan et al.[10] proposed UNL deconverter for tamil. The DeConverter from UNL to Tamil language is done using Universal Networking Language (UNL) as the intermediate representation. The information needed to generate the Tamil sentence is available at different linguistic levels. UNL greatly reduces the cost of developing knowledge or contents necessary for knowledge processing, by sharing knowledge and content.

3. Universal Networking Language

The Universal Networking Language is a computer language that enables computers to process information and knowledge. It is designed to replicate the functions of natural languages. Using UNL, people can describe all information and knowledge conveyed by natural languages for computers. As a result, computers can intercommunicate through UNL and process information and knowledge using UNL, thus providing people with a Linguistic Infrastructure (LI) in computers and the Internet for distributing, receiving and understanding multi-lingual information. Such multilingual information can be accessed by natural languages through the UNL System [7]. UNL, as a language for expressing information and knowledge described in natural languages, has all the components corresponding to that of a natural language.

UNL project is aimed at elimination of the language barrier. Main approach of this system is to represent information in the form of knowledge, using language independent Interlingua to represent knowledge. With this characteristic in mind, Universal Networking Language (UNL) is developed. UNL intermediates understanding among different natural languages. UNL represent sentences in the form of logical expression without ambiguity. It is an intermediate language, which allows communication among people of different language using their mother tongue. The UNL is a language specification for the exchange of information over the Internet [8]. The motivation behind UNL is to develop an Interlingua representation such that semantically equivalent sentences of all language have the same Interlingua representation. UNL plays the role of an interface between different languages to exchange information. UNL represent each sentence in the given text as set of relation.

4. Proposed Work

The Deconverter is a language independent generator, which provides a framework for syntactic and morphological generation synchronously. It can convert UNL Expressions into a variety of natural languages, by using respective word dictionaries and sets of grammar rules of deconversion of the target languages [10]. A word dictionary contains the information of words that correspond to UWs included in the input of UNL Expressions and grammatical attributes (features) that describe the behaviors of the words. Deconversion rules (grammar rules of deconversion) describe how to construct a sentence using the information from the input of UNL Expressions and defined in a word dictionary. The Deconverter converts UNL Expressions into sentences of a target language following the descriptions of Deconversion rules.

A "Deconverter" which generates natural language from UNL, plays a core role in the UNL system. The UNL Representation is given as input text [11]. Using the UNL Representation, a UNL Graph is generated using UNL Relations such as agt, mod, man etc. The Graph Analyzer analyzes the UNL graph to get tiling of text using Tamil words equivalent to the universal words are collected from the word dictionary. The Text Tiling is to arrange the sequence of terms to realize the sentence. Tamil is a morphologically rich language and hence a large amount of information can be generated in the morphological generation phase with the help of analyzing the UNL words and binary relations. Morphologically formed words with the relation and syntactic rules are used for the sentence formation process. This sentence formation process generates the Tamil language sentence. The Proposed Work Architecture is shown in figure 1.

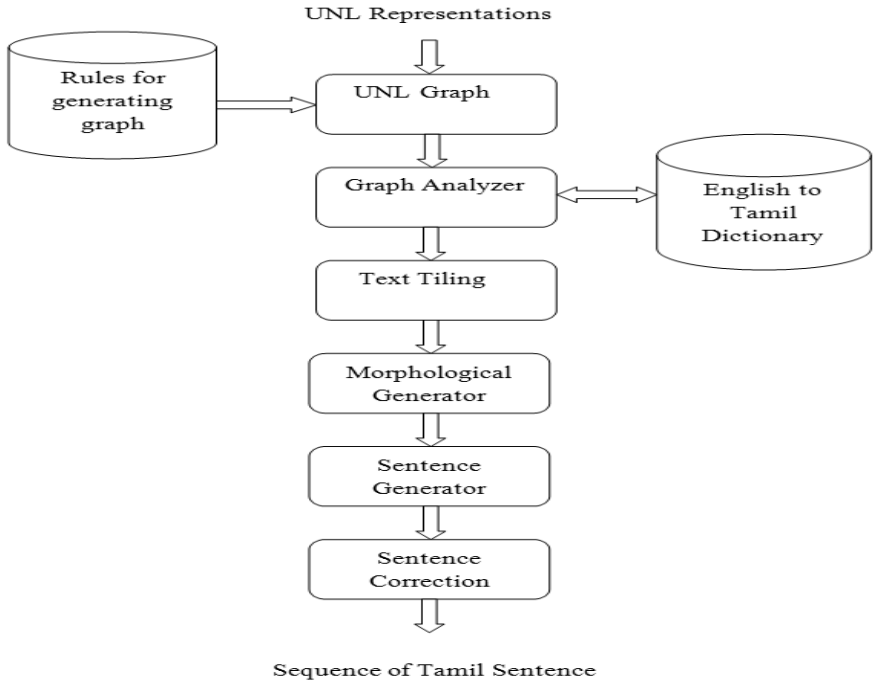


Figure 1. Proposed Work Architecture

4.1 UNL Representation

Universal Networking Language (UNL) is a computer language that enables computers to process information and knowledge across the language barriers. It is composed of UNL Expressions, Universal Words (UWs), Relations, Attribute.

•UNL Expression: UNL expresses information and knowledge in the form of semantic network. The semantic network of the UNL is a directed graph [12]. Such a semantic network of the UNL is called a "UNL Expression" or "UNL Graph". A UNL expression therefore consists of a UW or a set of binary relations. In UNL documents, a UNL expression for a sentence is enclosed by the tags {unl} and {/unl}.

•Universal Words (UWs): UWs are words of the UNL, constitute the UNL vocabulary. A combination of a set of UWs linked with each other through relations and modified by attributes expresses the meaning of a sentence.

•Relations: There are 46 relations in the UNL, such as 'agt', 'gol', 'obj', etc. They are used to connect every two UWs or scopes to construct the semantic networks of UNL Expressions. The relations are edges in the UNL graphs that constitute UNL Expressions.

•Attributes: Attributes are mainly for the purpose to describe subjectivity information. It includes time, aspect, emphasis, focus, topic, attitude, feeling and judgment.

4.2 UNL Graph

UNL expresses information and knowledge in the form of semantic network.The semantic network of the UNL is a directed graph.Such a semantic network of the UNL is called a "UNL Expression" or "UNL Graph".

4.3 Graph Analyzer

The Graph Analyzer analyzes the UNL graph to get tiling of text using Tamil words equivalent to the universal words are collected from the word dictionary. The Text Tiling is to arrange the sequence of terms to realize the sentence. Tamil words equivalent for the universal words are collected from the word dictionary [13]. For example, 'அவன்', 'மிக', 'ஓடு', 'வேகம்' are the Tamil word equivalents for the universal words

'he', 'very', 'run', and 'fast' respectively. From the binary relation in UNL, 'அவன்' is agent of the action 'ஓடு' and the verb 'ஓடு' is mentioned as past tense in UNL format. If the agent is third person singular means, the gender marker 'ஆன்' is added to the verb 'ஓடு' and the past tense marker 'n' is also added. 'வேகம்' defines the way to carry out an event 'ஓடு' and it is adverb. So 'ஆக' is added to the adverb 'வேகம்'. The word 'மிக' is intensifier and it modifies the adverb 'வேகம்' and it is simply added before adverb 'வேகம்'. So the generated sentence from the above information is 'அவன் மிக வேகமாக ஓடினான்'.

4.4 English to Tamil Dictionary

The word dictionary consists of about 1 Lakh English words and its equivalent Tamil words. The first step is to retrieve the Tamil word corresponding to every head word (HW) in the UNL [14] . The dictionary is used as a Universal Word dictionary. The query to retrieve the Tamil word matches the Headword. The primary task is to retrieve the relevant dictionary entries from the Tamil language word dictionary corresponding to the words in the word part of the UNL structures. The word entries of each language are stored in the Word Dictionary. Each entry of the Word Dictionary is composed of three kinds of elements: Headword, Universal Word (UW) and Grammatical Attributes. The headword is a notation/surface of word of a natural language. UW expresses the meaning of the word, which is to be used as a trigger or link for obtaining equivalent words or expressions. Grammatical Attributes are the information about the behaviour of the word in a sentence, which is to be used in deconversion rules.

4.5 Morphological Generator

A morphological generator is a program that performs the task of morphological generation. Morphological generation may be considered an opposite task of morphological analysis [15]. A morphological generator needs to be designed to tackle the different syntactic categories such as nouns, verbs, adjectives, adverbs. The general format of the morphological generator is Stem/root + suffixes · Word. The morphological generator generates morphological forms for nouns and verbs when the root word is given. The morphological generation mainly deals with the concatenation of corresponding suffixes with the root word to form a word of specific grammatical category.

The Morphological Generator takes lemma and a Morpho-lexical description as input and gives a word-form as output. The aim in morphological generation is to produce the inflected form of a word according to the features and values in the Feature Structure. It is also necessary to reuse the linguistic resources created for analysis purpose. From practical point of view, morphological generation is the inverse process of analysis, namely the process of converting the internal representation of a word to its surface form [16]. The same rule definitions can be used to generate the desired word form as used for analysis. The only difference will be the direction of execution order of the elements in the rule definition. The morphological generation mainly deals with the concatenation of corresponding suffixes with the root word to form a word of specific grammatical category.

The input of the morphological generator would be the root word which then inflects this word to the morphology of the respective language and gives as the output the target forms of the word [17]. The Morphological structure of Tamil verb is quite complex since it caters to person, gender, and number markings and also combines with auxiliaries that indicate aspect, mood etc. While morphologically generating the verb, the gender, number and person of the subject is necessary in order to select the appropriate suffix catering to the selected tense.

Tamil is a morphologically rich language and hence a large amount of information can be generated in the morphological generation phase with the help of analyzing the UNL words and binary relations. The word level Morphological generator for Tamil generates the derivative word for the root word and the various features conveyed by the suffixes.

4.6 Sentence Generator

Morphologically formed words with the relation and syntactic rules are used for the sentence formation process. This sentence formation process generates the Tamil language sentence. After adding the suffixes to the noun and the verb forms of the root words, the words need to be framed into a sentence. Tamil grammar is used for the creation of Tamil sentences [18]. The verbs in the input sentence are considered as central nodes.

The subject that lies just before the verb is added to the verb. Finally, the sentence is formed just by concatenation, i.e. the subject comes first, the phrase in the reversed order, then the verb, and this might be followed by any number of similar patterns. In the case of compound/ complex sentences, Tamil words corresponding to a conjunction are inserted.

4.7 Sentence Correction

The sequence of generated Tamil sentences is compared with the existing sentences in corpus, if both the suffix get matches then the output of the generated Tamil sentence is correct. Otherwise the more relevant sentences of the corpus information are used to correct the generated sentence.

5. Experimental Result

The UNL represents information sentence by sentence. Each sentence is converted into a directed hyper graph having concepts as nodes and relations as arcs [19]. The knowledge within document is expressed in three dimensions: Word knowledge is expressed by Universal Words (Uws). Concept Knowledge is captured by relating UWs through a set of UNL relations. The UNL Representation is given as input text. Using the UNL Representation, a UNL Graph is generated using UNL Relations such as agt, mod, man etc.

He ran very fast
[W]
He (icl>person) @generic:0
very (icl>concept) @generic:1
run (icl>do) @past .@entry:2
fast (aoj>thing) @generic:3
[/W]
[R]
2 agt 0
3 man 2
1 mod 3
[/R]

Here ‘agt’, ‘man’ and ‘mod’ are the relation labels.‘He(icl>person)’,‘very(icl>concept)’,‘run(icl>do)’ and ‘fast(icl>thing)’ are the Uws. ‘@entry’ is used to indicate entry or main UW of a sentence. ‘@generic’ is used to indicate generic concept. ‘@past’ is used to indicate the time with respect to the speaker.

The figure 2 shows the UNL Graph. UNL expresses information and knowledge in the form of semantic network. The semantic network of the UNL is a directed graph. Such a semantic network of the UNL is called a “UNL Expression” or “UNL Graph”.

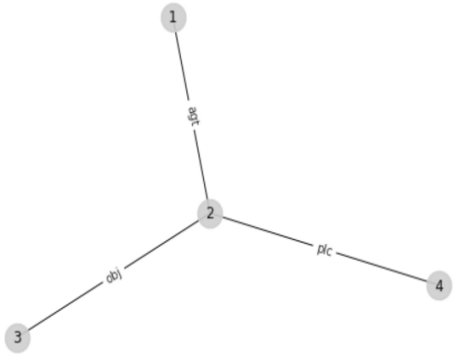


Figure 2. UNL Graph

The Figure 3 shows the Word Dictionary. The Word Dictionary contains English words and its equivalent Tamil words. The dictionary is used as a Universal Word dictionary. The query to retrieve the Tamil word matches the Headword.

Figure 3. Word Dictionary

The Figure 4 shows the Translation of English Words. The word or text is translated from English to Tamil using Word Dictionary.

Figure 4. Translation

The process of converting any word from one language to another without changing its pronunciation and phonetics is known as Transliteration. The Figure 5 shows the Trigram based reward value. Transliteration is the process of transferring a word from the alphabet of one language to another [20]. Transliteration helps people pronounce words and names in foreign languages. In translation transliteration is used for named entities. The Figure 6 shows the Transliteration.

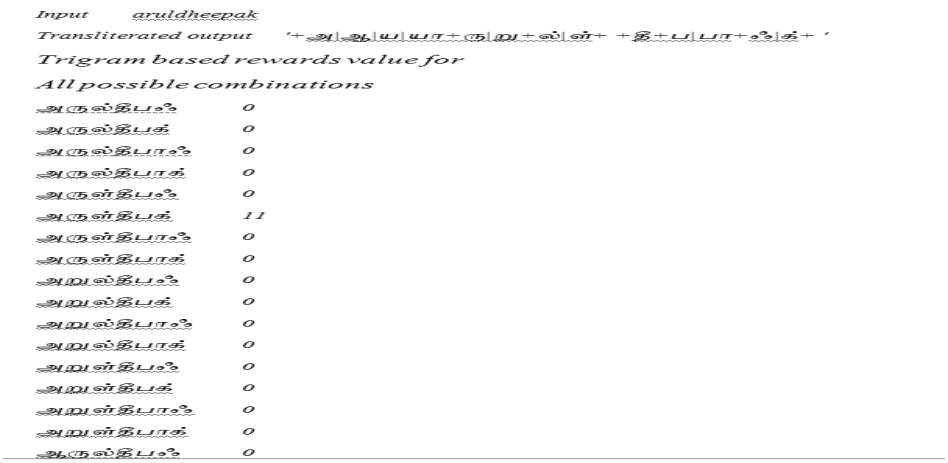


Figure 5. Trigram based reward value

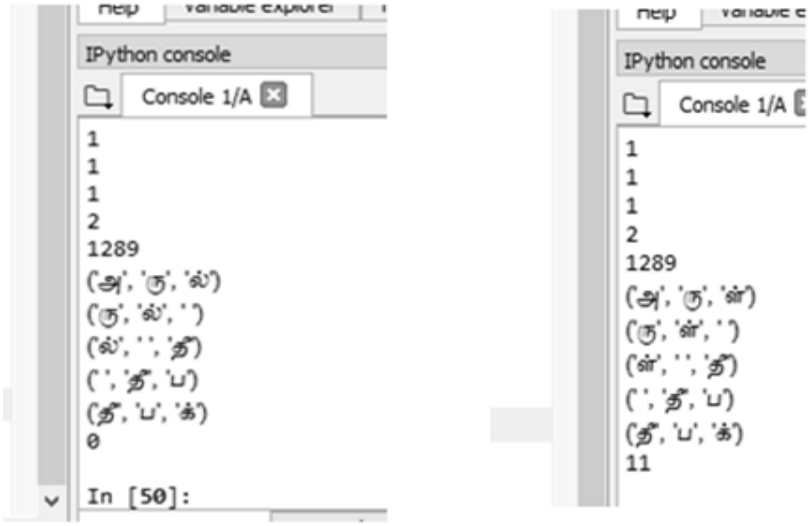


Figure 6. Transliteration

The morphological generator generates morphological forms for nouns and verbs when the root word is given. The input text is given in type of a box. After the necessary Tamil noun rules are applied, the possible combinations of noun words are shown in Figure 7.

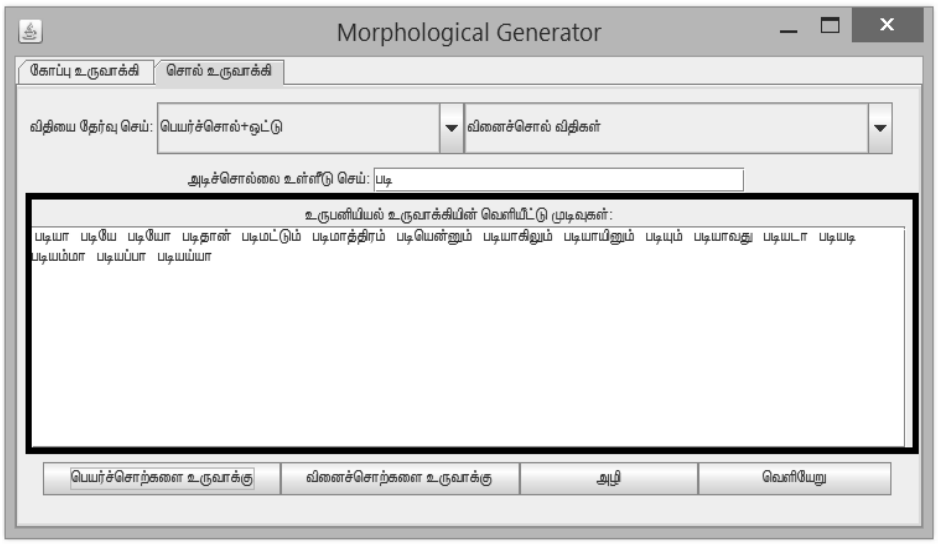


Figure 7. Output for Noun Generator

After the necessary Tamil verb rules is applied, the possible combinations of verb words are shown in figure 8.

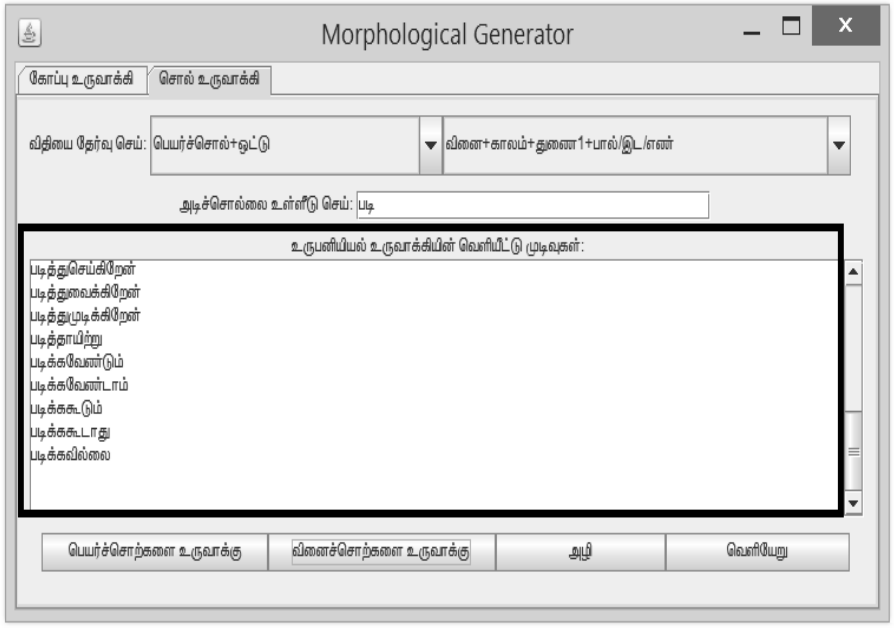


Figure 8. Output for Verb Generator

The figure 9 shows Sentence Generator. Morphologically formed words with the relation and syntactic rules are used for the sentence formation process. After adding the suffixes to the noun and the verb forms of the root words, the words need to be framed into a sentence.

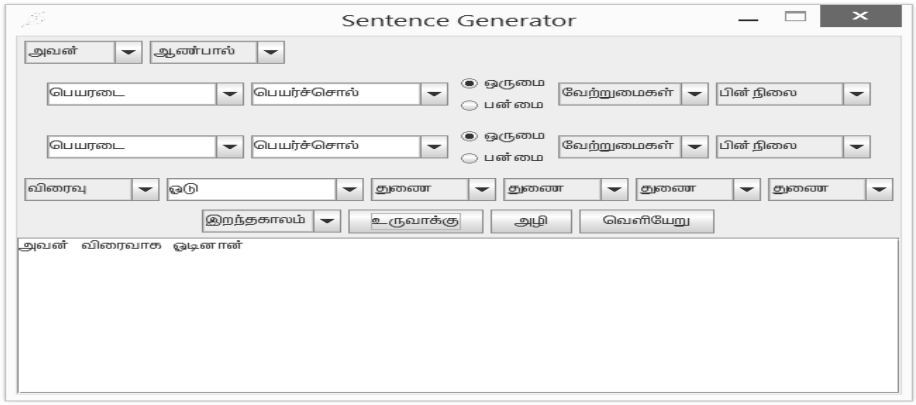


Figure 9. Sentence Generator

The figure 10 shows the Corpus for Sentence Correction. This corpus is compared with the sequence of generated Tamil sentence in the Sentence Generator in Figure 9. If both the suffix gets match then the output of generated Tamil sentence is correct.

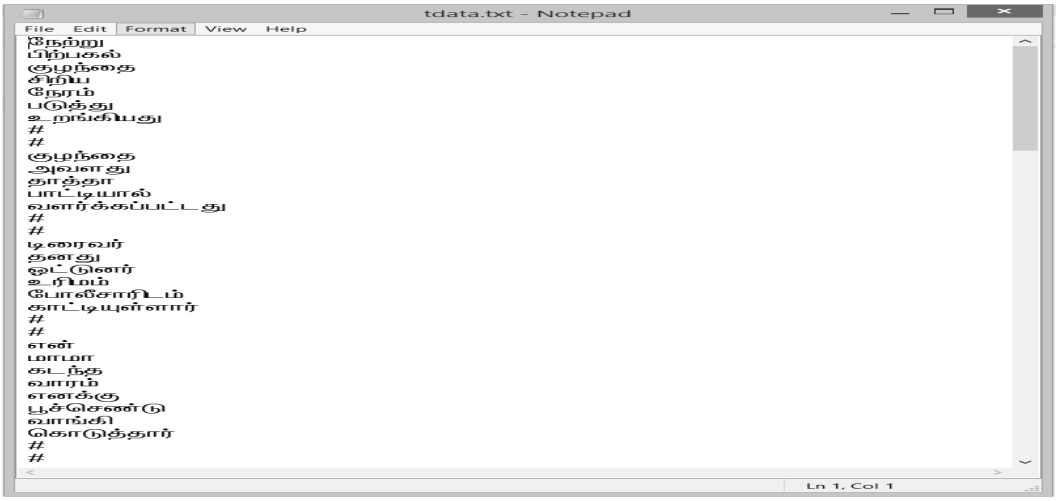


Figure 10. Corpus

6. Performance Evaluation

The performance of the entire system is evaluated using the parameter is described below.

Word error rate (WER): WER is a common metric to evaluate the performance of an MT system. It is determined by calculating the Levenshtein distance between those words in the candidate translation and the reference translation, which have a common prefix of at least three. As discussed already, the Levenshtein distance essentially gives the number of additions, deletions and modifications of words between the candidate and reference translations. WER is calculated for every word in the sentences of the enconversion. The results are then averaged over all the sentences in the corpus.

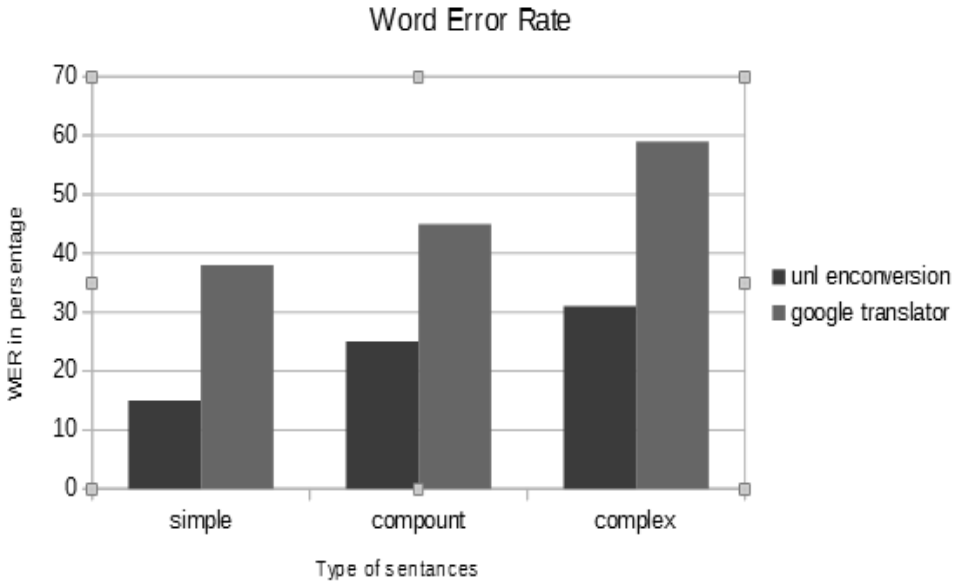


Figure 11. Bar chart showing the percentage of WER for different types of sentences

WER for our system using the UNL enconversion was calculated to be 25%, whereas for the Google translation it was 59%. Figure 5 gives the comparison of WER for simple, compound and complex sentences in the dataset, for our translation system as against the Google translation. The data in figure 11 clearly shows the advantages of incorporating UNL graph and a morphological generator in the system. The Google translation just uses a corpus based approach, and hence, complete sentences are not formed that convey the tense and gender of the subject. These results show that our newly created UNL graph and morphological generator are quite competent, in that it generates words much closer to the ones in the reference translation. However, the morphological generator is prone to a few sandhi errors and also some conjunction errors. Errors in the POS tagger also caused wrong suffixes to be added to the root word. The efficiency of our translation system could be improved to reduce the WER, by further enhancing the rules in the morphological generator.

7. Conclusion and Future Enhancements

In this work, a UNL deconversion for sentence realization in Tamil from UNL representation is developed. It deals with many issues and the required solutions are introduced. The UNL eliminates the need to study both target and source language by the programmer. Once a document is converted into UNL it can be converted into native language using the corresponding deconverter for that language. Thus it eliminates the need for individual machine translation development. In Tamil most information for generating sentence from UNL structure is tackled in morphological and syntactical level. First, the UNL representation is given as input and the UNL Graph is generated and analyzed. The Tamil words equivalent for the universal words are collected from the word dictionary and arranged in a sequence order. The use of UNL as intermediate representation makes translation of Tamil language available worldwide uses a standardized format.

In the future, we can improve the efficiency of the deconverter by introducing the number of heuristics that are currently determining the syntax plan and making them more precise by testing there results on large corpus. To extend the syntax planning, lexical knowledge of the universal words are to be used. In the current implementation, the Syntax Plan is generated purely on the basis of relation labels. But the belief is that in large complicated sentences the lexical information of the UW will have a significant effect on the syntax plan and to make the system more linguistically more strong. This system can be further

improved by removing disamibuigty of words by employing word net and extending the corpus dictionary to include more number of words in it.

REFERENCES

1. Rajeswari Sridhar, Pavithra Sethuraman and Kashyap Krishnakumar, "English to Tamil machine translation system using universal networking language",Vol.No.4, Issue No.6, pp no.607-620, March 2016.
2. Le Thuyen, Phan Thi, and Vo Trung Hung, "Automatic translation for Vietnamese based on UNL language", International Conference on Electronics, Information, and Communications (ICEIC), pp no.1-5, Jan.2016.
3. Baljeet Kaur Dhindsa and Dharam Veer Sharma,"Translation Challenges and Universal Networking Language", International Journal of Computer Applications(IJCA),Vol. No. 133, Issue No.15,pp no. 36-40, January 2016.
4. Imane Taghablout, FadouaAtaa Allah and Mohamed marraki, "Amazigh verb in theUniversal Networking Language", IEEE Conferences of Computer Systems and Applications (AICCSA),Vol. No.4, Issue No.6,pp no. 1-4, November 2015.
5. M. F. Mridha, Aloke Kumar Saha, Md. Akhtaruzzaman Adnan, MollaRashied Hussein and Jugal Krishna Das, "Design and Implementation of an Efficient Enconverter for Bangla Language", ARPN Journal of Engineering and Applied Sciences(ARPN),Vol. No. 10, Issue No. 15, pp no. 6543-6548, August 2015.
6. Biji Nair, Rajeev R and Elizabeth Sherly, "Language Dependent Features for UNL-Malayalam Deconversion", International Journal of Computer Applications (IJCA),Vol. No.100, Issue No.6,pp no.37-41, August 2014.
7. Ananthi Sheshasaayee and Angela Deepa V.R, "The Role of Morphological Analyzer and generator for Tamil language in Machine Translation Systems", International Journal of Computer Science and Engineering (IJCSE), Vol. No.2, Issue No.5, pp no.107-111, 2014.
8. S.Lushanthan, A. R. Weerasinghe and D. L. Herath, "Morphological Analyzer and Generator for Tamil Language", IEEE International Conference on Advances in ICT for Emerging Regions (ICTer), Vol No.7, Issue No.5, pp no.190-196, December 2014.
9. Kumar, Parteek, and Rajendra Kumar Sharma. "Punjabi Deconverter for generating Punjabi from universal networking language", Journal of Zhejiang University Science C , pp no.179-196, March 2013.
10. Hameed, M. S., Subalalitha and C. N., Geetha T. V, "A deconverter framework for Malayalam", In Proceedings of the International Conference on Advances in Computing, Communications and Informatics, pp no. 847-856, August 2012.
11. Bhattacharyya, "Multilingual Information Processing Using Universal Networking Language", IndoUK Workshop on Language Engineering for South Asian Languages (LESAL), April 2001.
12. Velliangiri Dhanalakshmi, M Anand Kumar and R U Rekha, "Grammar Teaching Tools for Tamil language", International Conference on Technology for Education, pp. 85-88 , July 2010.
13. Md. NawabYousuf Ali, Jugal Krishna Das and S. M. Abdullah Al- Mamun, "Specific Features of a Converter of Web Documents from Bengali to Universal Networking Language", IEEE International Conference on Computer and Communication Engineering,Vol No. 8,Issue No. 2, pp no. 726 - 731,2008.
14. Dhanabalan, T., & Geetha, T. V, "UNL deconverter for Tamil", International Conference on the Convergences of Knowledge, Culture, Language and Information Technologies, December 2003.
15. M.N.Y.Ali, A.M.Nurannabi, G. F. Ahmed, J.K.Das, "Conversion of Bangla Sentence for Universal Networking Language", International Conference on Computer and Information Technology (ICCIT), Dhaka, pp.108-113, 2010.
16. Aloke Kumar Saha, M. F. Mridha, Jahir IbnaRafiq and Jugal Krishna Das, "Data Extraction from Natural Language Using Universal Networking Language",IEEE International Conference on Current Trends in Computer, Electrical, Electronics and Communication" (ICCTCEEC) ,Vol No.2, Issue No. 7,pp no.24-29, September 2017.
17. Hiroshi Uchida, Meiying Zhu, "The Universal Networking Language beyond Machine Translation", UNDL Foundation, September 2009.
18. Dey, K., and Bhattacharyya, P, "Universal Networking Language based analysis and generation of Bengali case structure constructs", Universal Network Language: Advances in Theory and Applications, pp no. 215-229, 2005.
19. For Universal Networking Language: Universal Networking Digital Language Foundation. http://www.undl.org/
20. AjiNugraha , SantosaKasmaji and AyuPurwarianti, "Employing Natural Language Processing to Analyse Grammatical Error in a Simple Japanese Sentence", IEEE International Conference on Electrical Engineering and Informatics(ICEEI), Vol No. 3, Issue No. 7, pp no. 82-86, August 2015.

One-ANAPHORA RESOLUTION IN TAMIL

Vijay Sundar Ram
AU-KBC Research Centre, MIT Campus of Anna University, Chennai
sundar@au-kbc.org

Abstract. Natural language is cohesive. The cohesiveness is brought by various phenomena. Reference is one of the phenomena which brings cohesiveness. Reference includes different anaphoric expressions and *one*-anaphora is one among them. It is less attempted in Indian languages and there is no published work in Tamil. We have described *one*-anaphora in Tamil discourse and presented a rule-based algorithm to identify and resolve the *one*-anaphors. We have evaluated the engine with a set of web News dailies. The results are encouraging.

Keywords: *One*-anaphora, Tamil, Reference, *One*-anaphora resolution

1 Introduction

The cohesiveness in natural language brings elegancy to the text. Reference is one of the major phenomenon which brings cohesiveness between intra and inter sententially. The reference markers are pronominal, reflexives, reciprocal, distributive, *one*-anaphor and noun-noun reference. Resolution of these reference markers plays a vital role in building semantic intensive NLP systems. In the present work, we describe one of the less attempted reference markers, *one*-anaphora in Tamil discourse. We present a rule-based algorithm to identify and resolve*one*-anaphors.Cardinals occur as anaphoric expressions in certain instances, these are defined as *One*-Anaphors.

We have dealt *One*-anaphora resolution in Tamil. Tamil is one of the south Dravidian language. It is morphologically rich and highly agglutinated language. It is a nominative-accusative language and clauses are introduced by non-finite verbs. Though Tamil is a relatively free word-order language, noun phrases and clauses have rigid structures. In Tamil, genitive drop, accusative drop, copula drop, and pro drop are allowed. In the following section, we will explain *one*-anaphora in Tamil.

Consider the following discourse (Ex.1).

maraththil ainthu puRaakkaL uLLana. (1.a)

Tree(N)+Loc five pigeons be(V)+past

(There are five pigeons in the tree.)

iraNtu venniram maRRum muunRu saampalniram. (1.b)

Two white_colour and three grey_colour

(Two are white and three are grey.)

In Ex.1.b, there are two cardinals ‘iraNtu’ (two) and ‘muunRu’ (three). These two cardinals have occurred as anaphoric expressions. Both the cardinals ‘iraNtu’ and ‘muunRu’ refers to ‘ainthu puRaakkaL’ (five pigeons) in Ex.1.a.

The cardinals also occur with person, number and gender as ‘oruvan’ (one person 3 singular masculine), ‘oruththi’ (one person 3 singular feminine), ‘oruvar’ (one person 3singular honorific), ‘iruvar’ (two people plural honorific). Consider the following example.

viruthukkAka muuvar thernthuthetukkappattanar. (2.a)

Award(N)+adv three_people select(V)+past+3ph

(Three people were selected for the award.)

athil oruvar maruththuvar. null (2.b)

In_that one_person doctor (N) (copula)

(In that one is a doctor.)

Here in Ex.2.b, ‘oruvar’ (one person) in the second sentence is anaphoric and it refers to ‘muuvar’ (three people) in the first sentence.

Cardinals such as ‘oru’ (one), ‘iru’ (two) do not occur as anaphoric expressions, where as ‘onru’ (one), ‘iraNtu’ (two) etc can occur as anaphoric. These cardinals such as ‘onru’, ‘iraNtu’ etc also occur in listing of points. Consider the following example.

avan ***oru*** *nalla paiyan.* (3)

He(PN) one good boy.

(He is a good boy.)

Here in Ex.3, the cardinal ‘oru’ occurs as a quantifier preceding the head noun ‘paiyan’ (boy.) And it is not anaphoric.

avarukku ***onru*** *mattum pidiththathu.* (4)

He(PN)+dat one only like(V)+past+3s

(He liked one only.)

In Ex.4., cardinal ‘onru’ occurs as head noun. And it requires a referent.

There are many theoretical studies on one-anaphora such as Halliday & Hasan [1]; Webber [4]. There are very few attempts in computational system development. One of the earliest attempts in Indian languages is Vasisthpublished by Sobha & Patnaik [3]. The authors have classified the *One*-anaphora in Malayalam and Hindi on the basis of countability [+/-C]. The pronoun with [+C] can have features having [+count, +/-animate]. Consider the example pronouns such as ‘one’ is [+C], while ‘little’ is [-C]. The authors have presented a rule to resolve these anaphoric expressions which states that the antecedent NP is the non-subject NP in the immediate clause.

HweeTou et al. [2] has classified use of ‘one’ in English into six classes namely Numeric, Partitive, generic, Anaphoric, Idiomatic and Unclassified. They have a presented a computational system using C4.5 Decision tree, a machine learning technique to classify the ‘one’ and to resolve the anaphoric ‘one’.

2 Our Approach

We attempt to resolve *one*-anaphors with a rule-based approach. Here we follow Sobha & Patnaik [3] approach of classifying the *one*-anaphor. We try to resolve the +C *one*-anaphors, whose antecedents will also have +C characteristics. Consider the following examples Ex.5.

raamanitam ***pala cattaikaL*** *uLLana.* (5.a)

Raman(N)+Loc many shirts be(V)+past

(Raman has many shirts.)

athil ***onru*** *civappu niram.* (5.b)

In_that one red colour

(One is red in colour)

In Ex.5, the second sentence (Ex.5.b) has *one*-anaphor ‘onru’. It is has +C characteristics. It refers to ‘pala cattaikaL’ (many shirts) in the first sentence, which also have +C characteristics.

We have performed it in two steps.

1, Identification of *One*-Anaphors

2, Resolution of *One*-Anaphors with a rule-base approach.

2.1 Identification of *one*-anaphors

We try to identify the Cardinals that have occurred as noun phrase. Cardinals occurring as quantifiers in the noun phrase and Cardinals in the listing are not considers as these cardinals are not anaphoric. The algorithm is given below.

1) If Cardinal such as ‘onru’, ‘iraNtu’, etc occurs in a sentence, then step 2.
2) Check if the consecutive sentences have cardinals such as ‘onru’, ‘iraNtu’ in the starting of the sentences. If consecutive sentences do not have cardinals in the beginning of the sentences, then the cardinal is classified as anaphoric.

2.2 Resolution of *One*-Anaphor

After identifying the anaphoric cardinals, we proceed to resolve it using rule-based approach. We look for non-nominative noun phrases with [+C] characteristics in the immediately preceding clause or sentence. Step-wise process is described below.

1) If a sentence with *one*-anaphor with [+C] occurs, then look for non-nomimative NPs in the preceding immediate clause or sentence.
2) Check if the NP has [+C] characteristics, then it is chosen as antecedent NP.

2.2.1 Identification of NP with [+C] characteristics

Characteristic of NP with [+C] is determined based on the following two conditions.

1) If the NP has a quantifier then YES.
2) If the NP is in plural form then YES.

3 Corpus Description

We have collected 400 News articles from Tamil News dailies online versions, containing cardinals. We first scraped the text from the web pages and processed it with a sentence splitter and tokeniser. The sentence splitted and tokenised text is pre-processed with syntactic processing tools namely morphanalyser, POS tagger, chunker, pruner clause boundary identifier. The text enriched with shallow parsed information is fed to Named entity recogniser and thenamed entities are identified. The News articles are from Sports, Disaster and General News. The distribution of the cardinals is given in table 2.

Table 2: Distribution of Cardinals in the Corpus

S.No	Type	Number of Occurrence
1	Quantifier	186
2	Used in Listing	93
3	Anaphoric NPs	42

4. Experiments and Result

The text enriched with shallow parsing and Named Entity information is fed to rule based engine to identify the anaphoric cardinals and then processed with the *one*-anaphora resolution system. The performance measures namely precision, recall and f-measure are presented in the table 3.

Table 3: Performance Measures

S.No	**Precision (%)**	**Recall (%)**	**F-Measure (%)**
1	77.23	64.70	70.31

The output of the rule-based engine is analysed. The observations are as follows.

The accusative drop in Tamil introduces error in identifying the antecedents, as we look for non-nominative NP in the immediately preceding clause or sentence. Consider the following example.

raamu coomuvukku muunRu cattaikaL kotuththaan. (6.a)

Ramu(N) Soomu(N)+DAT three shirt(N)+pl give(V)+past+3sm

(Ramu gave three shirts to Soomu.)

athil onRu mangcaL iraNtu patcai niramaakum. (6.b)

In_that one yellow two green(N) colour be(V)

(In that one is yellow and two are green.)

In Ex.6.b has two cardinals ‘onRu’ (one) and ‘iraNtu’ (two). These *One*-anaphors refer to ‘muunRu cattaikaL’ (three shirts) in Ex.6.a. The rule to resolve *one*-anaphor looks for a non-nominative noun phrase with (+count) in the immediate preceding clause or sentence. But here in the immediately preceding sentence Ex.6.a, as there is an accusative drop, the noun phrase ‘muunRu cattaikaLai’ has occurred as ‘muunRu cattaikaL’ with the accusative marker ‘ai’ being dropped. As the referent NP occurs as nominative NP, the rule fails to identify the correct antecedent. The errors introduced by preprocessing modules namely POS tagger and chunker leads to wrong identification of anaphoric cardinals.

5. Conclusion

We have presented a description on *One*-anaphora in Tamil discourse, one of the reference marker, which brings cohesiveness to the discourse.We have presented a rule-based methodology to identify and resolve *One*-anaphora. We have tested the methodology with a set of web Tamil News dailies and obtained F-measure of 70.3%. The accusative drop in Tamil discourse poses a challenge in identifying the correct antecedents.

Acknowledgement

Authors thank *IMPRINT India initiative* for the support given to carry out this research.

Reference

1. Halliday, M & Hasan, R 1976, Cohesion in English, Longman, London.
2. HweeTou, N, Zhou, Yu, Dale, Robert, Gardiner& Mary 2005, ‘A Machine Learning Approach to Identification and Resolution of One-Anaphora’, Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), Edinburgh, Scotland, UK, pp. 1105-1110).
3. Sobha, L & Patnaik, BN 2000, ‘Vasisth: An Anaphora Resolution System for Indian Languages’, In Proceedings of International Conference on Artificial and Computational Intelligence for Decision, Control and Automation in Engineering and Industrial Applications, Monastir, Tunisia
4. Webber, B 1979, A Formal Approach to Discourse Anaphora, Garland Publishing Inc., New York &London.

Why People Prefer English words over its Tamil Equivalent for a Closed Set of Words? - A Deeper Analysis

Subalalitha C.N[1], Balaji J.[2]
[1] SRM Institute of Science and Technology, [2] Allstate Solutions
{subalalitha, jagank.balaji}@gmail.com

Abstract. Tamil language is one of the classical languages and is one of the oldest languages in the world. A survival of a language is in the hands of the people who pass it on to their next generation. Tamil has survived this long but most of the current Tamil generation is educated through English language and they tend to use many English words in the place of Tamil words. This may be a hindrance in passing the Tamil language to the next generation. This paper proposes an idea to identify those group of English words which are used in the place of Tamil words and generate a new word which is as easy to pronounce the English equivalent. The intuition is that people prefer English words over Tamil words as the English equivalents are much easier to pronounce. This idea is explored, and the new word is coined in such a way that it is similar to its English equivalent. This might help people to replace the English word with the newly coined word.

1. Introduction

Tamil language has a very long history and its literatures are 2500 years old. It is one of the classical languages in world. Tamil speaking society is spread across the globe and everyone is striving hard to make Tamil to live long till this earth is alive. Many languages failed to cope up with the advancements in the computers. The main reasons may be their scripts were not compatible with the computers and ample websites were not created in that language which prevented many applications to be built for those languages. Tamil has crossed all these barriers very easily and it is in the top position among all Indian languages on World Wide Web as per the survey conducted by Google in the year 2017 (Senthil, Varavanai ,2017). Thanks to the Tamil society for contributing such an extravagant pageant for Tamil in the computational arena. Despite all the bright sides, it is bit disappointing that people still prefer English words over its Tamil equivalent when it comes to a set of words. This paper analyses the root cause behind it and tries to bring a solution to make the Tamil to prefer Tamil words while conversing or writing in Tamil.

This paper focusses on two factors namely, easiness to pronounce and coining a new word as an alternative to replace the English word . The new word coining should have some similarity with its English equivalent word. The easiness score idea is similar to the pleasantness score proposed by Elanchezhiyan et al.(2013) which measures the sweetness of a word when used in a song. The easiness score focusses on the easiness to pronounce. This paper puts forth only the idea and the implementation is under progress.

2. The Proposed work

The proposed idea is divided into two parts namely, the analysis of the problem of why people tend to opt English words over Tamil words and the second part is about the solution that may be framed to overcome this problem.

2.1 The Analysis

A Tamil child's brain is first loaded with Tamil vocabulary since its birth but most of the Tamil population are educated through English language, the English vocabulary and the Tamil vocabulary are equally loaded with Tamils which cannot be avoided. This bilingual knowledge gives preference for English words most of the time despite knowing its Tamil Equivalent. Let us look at this from a slightly different

angle. Before analyzing why people are preferring English words for certain Tamil words while speaking or writing, let us analyze why people prefer Tamil words over English words for a large set. The reasons may be, 1. Those Tamil words are inculcated in the mind long before their English Equivalent 2. Those words are frequently used by the majority population. This gives us the clue of why people prefer English words for a closed set of Tamil words.
The reasons why an English word is prioritized over a Tamil word for the following reasons.

1. The English word was taught long before the Tamil equivalent word was taught.

2. The English word is used by most of the people.

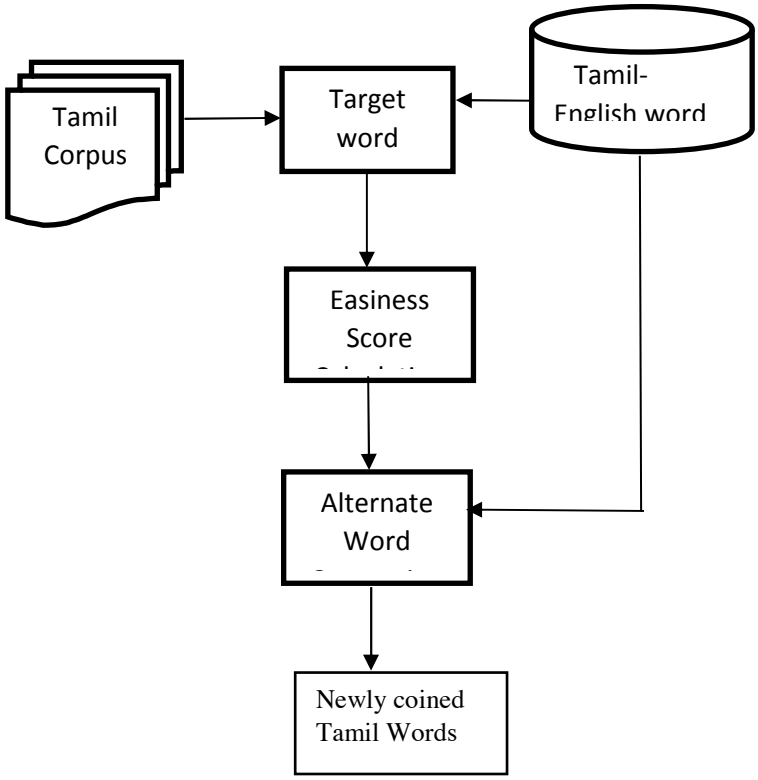
3. The English word may be comparatively easier to pronounce while communicating.

This paper analyses in deep on the third reason listed above and tries to look into what kind of Tamil words loses its place to English words. We coined out the third reason when we analyzed many blogs and websites where colloquial style of Tamil is used.

The next part gives an overview of the solution that may aid to overcome this problem.

2.2 Solution

This paper puts forth an algorithm which involves the idea of framing a metric which measures the easiness of pronouncing a Tamil word. The easiness score measures the easiness in pronouncing a Tamil word. Since the exact algorithm of how an easiness score is calculated is being worked out by the authors, this paper reveals the workflow of the algorithm.



A set of Tamil web documents will be crawled from the web and a context based analysis is done to extract the usage of English words in the place of Tamil words. The web documents targeted are the blogs and tweets. The targeted words are those which are mostly replaced by an English equivalent words and the

easiness score is calculated for the targeted word. A threshold is set and if a word crosses that threshold, it is passed onto the alternate word generation module where an alternate word is generated. The alternate word is generated based on the closedness of the English word.

The idea of coining a new alternative word arose from the way the word, “Catamaran” originated from the Tamil word, “கட்டுமரம்”. The word “catamaran” has much similarity in terms of pronunciation. May be this would have paved way for the most of the Tamils to prefer “Catamaran” over “கட்டுமரம்”.

This paper perceives the above mentioned idea in the reverse way. This proposed algorithm checks the Easiness Score of the words that are rarely used by the Tamils by comparing with its English equivalent and generates a closer Tamil words which has higher Easiness Score. The new word generation algorithm would coin a new Tamil words which has much similarity with it’s English equivalent and hence the transition from using the English to Tamil words would be much easier for the Tamils.

People prefer using the word, “Biscuit” than using the words, மாச்சில்லு (maachillu), ஈரட்டி (eeratti) and விசுக்கோத்து (viskoththu). If a new word which is similar to biscuit like, “ஈரட் (erat)” which sounds similar like “biscuit” is coined, the people would tend to use “ஈரட்” than “biscuit”. The easiness score and the new word generation algorithm would target features that will produce such alternatives.

3. Conclusion

This paper proposes an initial framework to make people use more Tamil words than using English words. This paper has explored the easiness in pronouncing a Tamil word which may be one of the reasons for the people to opt English words which are much easier to pronounce than their Tamil equivalents. The paper also proposes a solution to coin a new Tamil word which is much similar to the English word and is easier to pronounce than its earlier version. This paper makes an initial attempt to prevent Tamils choosing to use more Tamil words than English thereby making this ancient classical language long live for many more centuries.

References

1. Elanchezhiyan K, TamilSelvi E, Karthikeyan S, Rajapandian C & Madhan Karky Scoring Models For Tamil Lyrics ,12th International Tamil Internet Conference, Malaysia, Aug-2013.
2. Senthil, Varavanai (2 May 2017). "இணையத்தில் அதிகம் பயன்படும் மொழி 'தமிழ்'..! - கூகுள் சர்வே முடிவு”

Fine-Grained Named Entity Recognizer for Tamil

Malarkodi C.S. and Sobha Lalitha Devi
AU-KBC Research Centre,
MIT Campus of Anna University, Chennai
sobha@au-kbc.org

Abstract. This work describes the development of Fine-grained Named Entity Recognizer for Tamil using the machine learning technique CRFs. The tagset used for NE identification consists of 106 tags. The main objectives of the proposed work is to develop the named entity system for Tamil which can be used for higher level NLP applications like Information Extraction, Question & Answering Systems etc. The linguistic analysis of Part of Speech (POS) occurring in the context of named entities has been done to find out the patterns of NEs. The error analysis is given for each feature combinations. The different types of problems encountered in the development of NE system are also discussed. The post processing rules are implemented to solve the issue discussed in the error analysis section and the performance of the system is significantly improved.

1. Introduction

Named Entity Recognition (NER) is defined as the automatic identification of proper names and classify it into the respective categories such as names of person, location, organization, products etc. Named entity identification is the fundamental task in IE and higher level NLP applications. There is a need to maintain and extract the useful information from the news articles and web data which grows rapidly in a large extent. NER systems can identify the names of the persons, locations and organizations in the web data and thus categorizing the respective articles which enables the content discovery much easier. In relation extraction, the named entities such as person names and place names need to be identify first, in order to find out the relations exists among these entities. In event extraction, named entities provides the information about the persons involved in an event, where it happened and when the event has happened. In Q&A systems, NER plays a vital role, since the answer strings to the 'WH' questions such as when, where and who are named entities. In sentiment analysis, the named entities need to be identify first, to obtain the opinion about the respective person or a product.

Initially the term NER was defined in Message Understanding Conference (MUC), when the structured information about company and defense related activities need to be extract from the unstructured text. It was noticed that the main information units to be extract are named entities (Grishman et al. 1996). The very first research work in NER was done by Lisa F. Rau, who developed the system to recognize company names using hand-crafted rules and submitted in Seventh IEEE conference on Artificial Intelligence on Applications. In MUC-7, five out of eight systems were generated using rule based method (Chinchor 1998). Nadeau et al. (2007) has reported fifteen years of research carried out in the field of entity recognition. Earlier days, NER systems were developed using the hand-crafted linguistic and heuristic rules.

The first named entity corpus for 6 Indian Languages and English was developed as part of Cross Lingual Information Access (CLIA) Project which was funded by Government of India, Ministry of Communications & Information Technology. The named entity tagset used consists of 106 hierarchical tags, which is the standardized tagset widely used for the Indian Languages. The named entity corpus was developed for English and Indian Languages such as Tamil, Telugu, Hindi, Punjabi, Marathi and Bengali. The dataset was collected from various news articles, blogs related to tourism domain. The Tamil corpus consists of 5k sentences, 87k tokens and 18,654 named entities. The Telugu corpus has 2k sentences, 45k

tokens and 10k NEs. The Hindi, Punjabi, Marathi and Bengali corpus consists of 88k, 90k, 79k and 56k tokens respectively. The Hindi and Marathi corpus consists of 6k named entities. The Punjabi and Bengali corpus has 7k and 3k named entities. The English corpus with 85k tokens consists of 12k named entities. The nested entities are also tagged in the corpus (Malarkodi et al., 2012).

Sobha et al. (2007) developed a multilingual named entity system to identify the place names using Finite State Automaton (FSA). The language identifier was developed using statistical techniques to identify the language of the input sentence. The location identifier depends on the suffix stripping techniques and the input text need not be pre-processed for this task. The place identifier works with the precision of 95% and recall of 71% respectively. Vijayakrishna & Sobha (2008) worked on Domain focused Tamil Named Entity Recognizer for Tourism domain using CRF. It handles nested tagging of named entities with a hierarchical tag set containing 106 tags. They considered root of words, POS, combined word and POS, Dictionary of named entities as features to build the system. The three phase hybrid NE recognizer was constructed for Tamil with six NE tags by Pandian et al. (2008). In order to obtain the morphological components, a dictionary consists of word clues and case markers are used to identify the NE in the first phase. Semantic and syntactic information of subjects and objects are obtained second phase. The HMM algorithm exploited the statistical information obtained from the first two phases. They have claimed average f-score as 72.72% for various entity types. Malarkodi et al. (2012) discussed the various challenges, while developing the NE system in Tamil language. The 22 second level NE tags from the hierarchical NE tagset were used for system implementation. The challenges reported are agglutination, ambiguity, nested entities, spell variation, capitalization and noise in the data. They have done two sets of experiments, first using word, POS, and chunk information and second experiment was conducted to reduce the agglutination and morphological related issues. Hence the root word information was considered instead of actual word, along with POS and chunk information.

In 2013 AU-KBC has organized NER shared task (Pattabhi et al., 2013) as part of Forum for Information Retrieval for Evaluation (FIRE), to create a benchmark data for Indian Languages. The dataset was released for 4 Indian Languages like Bengali, Hindi, Malayalam, and Tamil and also for English. The training and test set used for Bengali consists of 62k and 48k tokens. The Hindi corpus has 105K tokens for training and 55k tokens for test purposes. The training and test data for Malayalam corpus consists of 41k and 28k tokens respectively. The Tamil dataset consist of 41k for training and 28k for test purposes. The English dataset has 219k tokens for training and 35k tokens for test set. The corpus was annotated with 106 hierarchical tags of Indian Language Tagset. Five teams have participated in the task. The various techniques used by the participants are CRF, rule based approach and list based search. The 2nd edition of NER track for IL has organized as part of FIRE 2014 (Pattabhi et al., 2013) for English and 3 IL namely Hindi, Malayalam, and Tamil. The main focus of this track is nested entity identification. The FIRE corpus consists of 140K token for English, 125K tokens for Tamil, 62K tokens for Malayalam and 127K for Hindi respectively. Though 10 teams have registered earlier, only 2 teams can submitted their test runs due to time constraint. The participants have used CRF and SVM for system development.

In 2013, ICON NER tool contest was conducted for six Indian languages namely Tamil, Telugu, Bengali, Marathi, Hindi, Punjabi and English. The dataset used for English consists of 85k tokens. The corpus used for Indian languages consists of 76K tokens for Tamil, 71K tokens for Bengali, 65K tokens for Marathi, 73K tokens for Punjabi, 86K tokens for Hindi respectively. The corpus was annotated with 22 outer level tags of Indian Language Tagset which consists of 106 hierarchical tags (Sobha et al., 2013). Antony et.al. (2014) constructed the NE system for Tamil Biomedical documents using SVM classifier. The features used are unigrams or bigrams, case markers, substring clues and tf-idf to identify the named entities in the bio-medical domain.

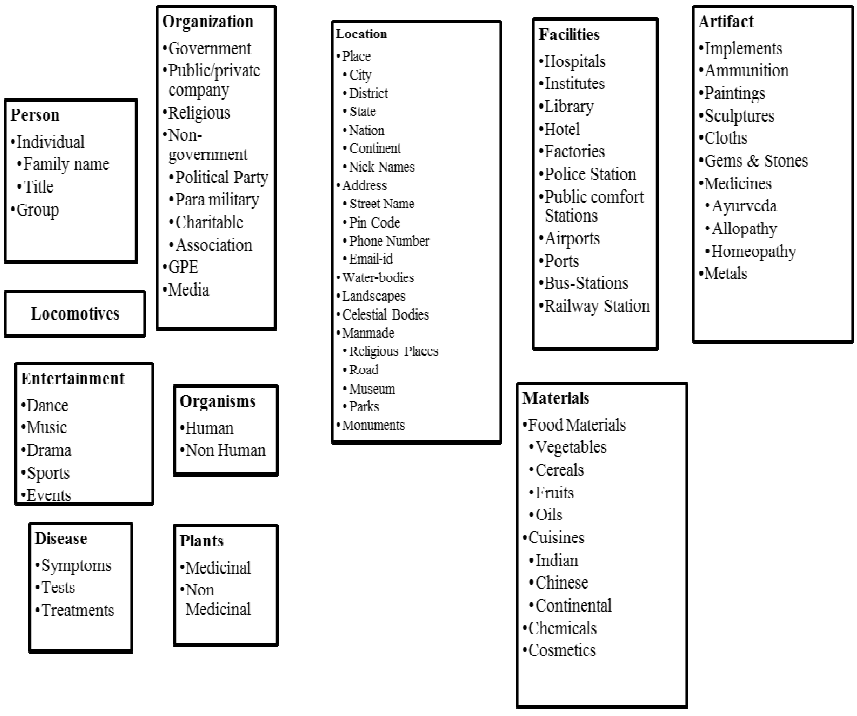
2. Corpus statistics and NE Tagset

The dataset developed as part of FIRE 2013 NER shared task and national level projects such as Cross Lingual Information Access (CLIA) are used for this work. The Tamil NER dataset consists of 200K tokens, 13K sentences and 27,498 named entities. This Hierarchical tag set was developed at AU-KBC Research Centre, and standardized by the Ministry of Communications and Information Technology, Govt. of India. This tag set is being used widely in Cross Lingual Information Access (CLIA). The NER tagset consists of 3 NE types as the major level. They are Entity expressions, Time Expressions and Numeric Expressions. The entity expressions are divided into 11 types namely person, location, organization, facilities, locomotives, artifact, entertainment, cuisines, plants, organisms and disease. The Numeric expressions are classified into 4 types. They are Distance, Count, Money and Quantity.

Table 1: Corpus Statistics

Languages	Tokens	Sentences	NEs
Tamil	2,04,144	13,571	27,498

Figure 1: Named Entity Tagset



3. Corpus Analysis

The part of speech patterns frequently occurred in the window of three (n-1 and n+1 positions) of the context of named entities are analyzed for Tamil and the results are discussed in this section. We analyse the corpus to arrive at the most suitable word level features for identifying the NE which can be used for machine learning purpose. We have taken a window of three words and identified the most frequent grammatical and typographical feature that occurs. The distribution of each feature in each language is given in detail below.

In Tamil corpus, the named entities occurred at the beginning of the sentence is 3,776 instances and in 2,056 instances named entities occurred at the end of the sentence, punctuations preceded the NE in 6,222 times and 4,596 times punctuations succeed the NE, common nouns preceded the NE in 6,274 times and succeeded the NE in 9,038 times, proper nouns occurred before NE in 2,250 instances and after NE in 2,868 instances. The postpositions occurred before NE in 999 instances, adjectives occurred before NE in 1418 instances and conjunction occurred before NE in 384 times. The verbal participle preceding the named entities in 716 instances and the relative participle verbs preceded the NE in 1,007 times. The finite verbs succeed the NE in 998 instances, postpositions, adverb and adjectives occurred at 1131, 980 and 878 instances respectively.

Figure 2: POS patterns of named entities in Tamil

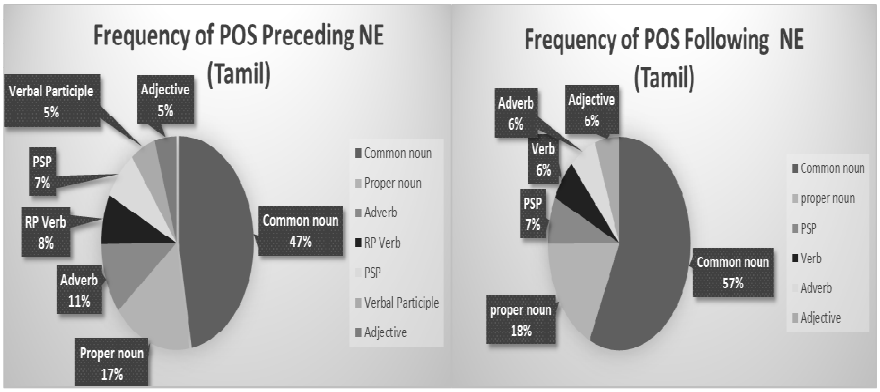


Figure 2.9 Frequency of POS tags in n-1, n+1 position (Tamil)

We have analysed the corpus for the various part of speech which are associated with the named entities. In the window of three the following are the grammatical features occurred. Also the Typographical features are also arrived through the analysis. From the above analysis we arrived at the following points. In Tamil the most commonly occurring pattern for NE are

1. **Grammatical patterns**: RP verbs precedes and follows , Common noun precedes or follows, Occurring after the verb, Postpositions precedes the NEs, Verbs succeeds the NEs AND Postpositions, adjectives or adverbs follows the NEs.

2. **Typological patterns**: .NEs in the beginning of the sentence, NEs in the end of the sentence, Punctuations followed NEs and NEs Occurring after punctuations.

4. Corpus Annotation

The text collected is pre-processed with part-of-speech tagging and chunking using automated tools. The pre-processed text is given as an input to the graphical user tool which was designed by AU-KBC to annotate Named Entities. This tool aids the annotators to tag the named entities without typological errors. The format to annotate named entities are given below.

<LEVEL1_NE_TYPE TYPE=LEVEL2_NE_TYPE SUBTYPE_1=LEVEL3_NE_ TYPE> Named Entity </LEVEL1_NE_TYPE>

Where LEVEL1_NE_TYPE denotes the major NE categories such as ENAMEX, NUMEX, TIMEX. LEVEL2_NE_TYPE indicates the second level tags in our hierarchical tagset namely Person, Location, and Organization etc. LEVEL3_NE_ TYPE refers to Individual, Place etc.

<ENAMEX TYPE=ENTERTAINMENT SUBTYPE_1=DANCE>Kathak</ENAMEX>

In the example, the type of the dance “Kathak” is tagged as “Entertainment” NE type and it comes under the subtype “Dance”

<ENAMEX TYPE=”FACILITIES” SUBTYPE_1=”INSTITUTE”> Madras Institute of Technology </ENAMEX>

In the above example, the name of the college “Madras Institute of Technology” is tagged as “Facilities” NE type and it comes under the subtype “Institute”

Inter-annotator agreement is defined as the degree of agreement among annotators. It is the percentage of judgments on which the two annotators agree when tagging the same content independently. The Kappa statistic k is a better measure of inter-annotator agreement which takes into account the effect of chance agreement (Ng et al., 1999). We calculated the kappa score for each NE text span. We had two annotators for each language. The overall kappa score for Tamil is 0.96, which means there is good agreement between the annotators.

5. Features Selection

The feature selection plays a significant role in machine learning technique. The performance of the machine learning system depends on the feature selection criteria and the pre-processing modules. This section describes about various features used for named entity identification

5.1 Lexical Level Features

The features depends on the word level information and the contextual information such as words occurring adjacent to named entities are considered as lexical level features. The lexical level features helps the system to learn the words and contextual cues surrounding the named entity. The contextual information, first word, suffix and prefix information are the lexical level features used in this work

5.2 Context Word Information

The context information denotes the tokens occurring adjacent to the named entity. The contextual words will be useful to capture the contextual cues surrounding the named entity. Hence, the combinations of words surrounding the current token in the window of five position are considered as features.

First word

We have analysed the training corpus for all the languages, to check the occurrences of named entities which occurred as first word of the sentence. Since subject comes in the beginning of the sentence, the subject might be a named entity in most of the languages. We observed that around 20% of the words occurring at beginning of a sentence in the training corpus are NEs for all the languages. So we consider the beginning word of the sentence as a feature. In Tamil corpus 3,776 times, named entities occurred at the beginning of the sentence.

Prefix and Suffix Information

The named entities can share a common prefix and suffix information. For example, the words ending with '-pettai' most likely specifies the place names such as "chromepettai, thenampettai and nazarethpettai" in Tamil. The suffixes such as '-pet, -pettai, -Ur, -patti, -pur, -puri, -bad' denotes the place names in Indian Languages. Hence we consider trigrams and four grams of prefix and suffix information as features.

5.2 Syntactic Level Features

The features depends on the linguistic or grammatical information are called as syntactic level features. The syntactic level features such as POS and chunk information are used in this work.

POS Information

Part of Speech tags denote the grammatical category of the word such as noun, verb, adverb, adjective etc. It is considered as a crucial factor in NE identification. The POS information of a current word and surrounding words are considered as features.

Chunk Information

The chunk information is used to group the tokens into phrases such as noun phrase, verb phrase, adverbial phrase etc. It helps the system to learn the boundary of named entities.

POS Patterns Preceding and Following NE

The POS tags occurring as context of named entity act as a trigger for NE identification. The part of speech patterns adjacent to named entity provides the grammatical information to learn the patterns of named entities. Hence we have decided to give importance to the POS tags preceding & succeeding named entities. Based on the linguistic analysis discussed in the previous section, the POS patterns of named entities occurring at the following positions are considered as feature.

- Frequent POS tag occurring at the proximity of NE in n-1 position
- Frequent POS tag occurring at the proximity of NE in n+1 position
- Frequent POS tag occurring at the proximity of NE in (n-1, n, n+1)
- Frequent POS tag occurring at the proximity of NE in (n-2, n-1, n, n+1, n+2

6. Experiments and Results

This section describes the experiments, the results and the discussion on the results obtained from the Tamil named entity recognition system. The detail statistics of the dataset used in this work is explained in the previous section. Inorder to find the best results, we have selected various feature combinations and using these features different sets of experiemnts were conducted. The intial section describes the experiemnts for the baseline system. Then the error analysis for each feature combination is discussed in detail. Follwing this section, experiemts and results for the extended system is discussed. The types of erros occurring in named entitiy identification is explained with examples. We have developed post-processing rules which are discussed in detail.

There are experiments conducted using different feature combinations to find the best feature set. The 10-fold cross validation was conducted for both the baseline system and the extended system. The average results obtained from the cross validation are given in results and discussion section. There are six NER models generated using different feature combinations for the baseline system. The results obtained using each feature combinations are analysed to find the error cases. The detail error analysis was done for each combination and from the error details the error-driven features are identified. The extended system was developed using the error –driven features and it improved the system's performance. There are seven NER models generated using different feature combinations for the extended system. The feature combinations used in six NER models of extended system are similar to the NER models discussed in the baseline system. The seventh NER model was generated using the dynamic features. The chunk information is not available for Marathi, Spanish, Dutch, German and Hungarian. So the model 2 and model 4 which use chunk as a feature is not generated for these languages. The other named entity models for these languages are generated without chunk information. The NER models generated using various feature combinations for both baselined and extended system are given in Table 2.

Table 2 Details of NER models & Feature Combinations

NER Models	**Model1**	**Model2**	**Model3**	**Model4**	**Model5**	**Model6**
Features	P	P+C	P+W	P+C+W	P+C+SP	P+C+SP+ W
Baseline System	Yes	Yes	Yes	Yes	Yes	Yes
Extended System	Yes	Yes	Yes	Yes	Yes	Yes

In the table P – POS Information, C – Chunk Information, W – Word related features, SP – statistical suffixes and prefixes.

Table 3: Results for Baseline System

Languages	**Tamil**		
Results	**PRE**	**REC**	**F-M**
P	26.99	37.72	31.46

P+C	27.57	38.14	32.00
P+W	46.07	70.14	55.61
P+C+W	46.33	70.19	55.81
P+C+SP	48.47	74.86	58.84
P+C+SP+W	55.89	77.86	65.07

We have conducted various experiments with different combinations of features and the average f-score obtained by 10 fold cross-validation for the combination of best feature sets are given in Table 3. From the above table it can be observed that with POS information alone, which is the baseline system, obtained the f-measure of 31.46%. The POS and Chunk information increased the recall by 1%. This shows that POS and Chunk feature together can improve the recall. This feature raised the overall performance by 1%. The inclusion of word as feature along with POS has improved the recall substantially by 32%. Whereas the inclusion of statistical suffix and prefix information further increased the precision by 2% and recall by 4% and thus increased the F-measure by 3%. Thus from the experiments we find that the combination of features such as pos, chunk, word, prefix and suffix gives the best results than the baseline system.

Table 4: Error Analysis for Baseline System

Features	Remarks
POS related features	**Errors:** 1) Named entities occurring in adjacent positions are tagged as single entity 2) one named entity with multiple tokens is tagged as two entities 3) beginning of an entity is tagged by intermediate tag, part of an entity is tagged by the system 4) food items are tagged as LOCATION 5) list of location names are tagged as PERSON 6) organization named are not identified correctly
POS, chunk	**Errors reduced by this feature:** 1) Same entity is tagged as two entities in POS related feature. Inclusion of chunk feature, reduced these kind of errors 2) Different entities in adjacent positions are tagged as one in the previous feature combination, that is also handled by this feature combination 3) list of location names tagged as person is identified as LOCATION **Errors:** 1) special days are tagged as location 2) person names are tagged as location in some instances 3) location names are tagged as person in some instances 4) person names occurring as part of place names are tagged as PERSON 5) NE boundary is wrongly tagged in some instances(I-tag is wrongly tagged or failed to tag)
POS, word	**Errors reduced by this feature:** 1) False positives generated by previous feature combinations are reduced nearly 50% for each NE categories by using the word feature. For example, in previous combination 136 instances person names are tagged as location, in this case it is reduced to 26. 2) Organization names are identified **Errors:**

	1) special days are tagged as PERSON 2) person names are partly tagged in some instances 3) one named entity with multiple tokens is tagged as two entities 4) person names occurring as part of place names are tagged as PERSON 5) place names occurring as part of organization names are tagged as LOCATION
POS, word and chunk	**Errors reduced by this feature:** Inclusion of chunk feature with POS and word feature rectified the cases where same entity is considered as different entities by the system. **Errors:** 1) special days are tagged as PERSON 2) person names are partly tagged in some instances 3) person names occurring as part of place names are tagged as PERSON 4) place names occurring as part of organization names are tagged as LOCATION 5) NE boundary is wrongly tagged in some instances(I-tag is wrongly tagged or failed to tag)
POS, chunk statistical suffixes and prefixes	**Errors reduced by this feature:** 1) location names tagged as person in previous feature combinations are rectified using the statistical suffixes and prefixes. For example, in 133 cases location names are tagged as person using previous feature combinations, inclusion of this feature reduced it to 59. **Errors:** 1) In some instances organization names occurring as part of title names are tagged as organization. 2) Place names are partially tagged in some instances 3) place names occurring as part of the organization names are tagged as LOCATION 4) Special days are tagged as PERRSON and LOCATION 5) person names occurring as part of place names are tagged as PERSON
POS, chunk statistical suffixes and prefixes, word	**Errors:** 1) In some instances organization names occurring as part of title names are tagged as organization. 2) place names occurring as part of organization names are tagged as LOCATION 3) Special days are tagged as PERRSON and LOCATION 4) person names occurring as part of place names are tagged as PERSON **Errors reduced by this feature:** 1) Location names occurring as part of organization names are tagged as location in previous feature combination. Similarly in some instances person names occurring as part of organization or location names are tagged as person. Though the same kind of errors existing in few instances, in comparison with previous feature combination the errors are reduced by using this feature combination

During error analysis, the error driven features are identified and the extended models are generated by eliminating those features. In this section the results obtained by each feature combinations in the extended system are discussed. The POS information yielded the precision and recall of 34% and 38% respectively. The chunk information improves the overall f-measure by 1%. The inclusion of word related features with POS and chunk information gives the improvement of 27% in precision score and 32% improvement in the recall value. The statistical suffix and prefix information along with pos, chunk and word combination improves the f-score by 5%. The combination of features used in the model6 performs better than any other feature combinations and improves the f-score by 19%.

Table 4: Results for the Extended System

Language	**Tamil**		
Results	**PRE**	**REC**	**F-M**
P	34.36	38.04	36.2
P+C	34.39	39.97	37.18
P+W	61.34	71.32	66.33
P+C+W	61.56	72.27	66.91
P+C+SP	61.21	77.52	69.36
P+C+SP+W	80.12	80.10	80.10

In comparison with the results obtained from the baseline system, the performance of the system is improved in each sets of experiments conducted in the extended system using different feature combinations. The different feature combinations shown in Table 4 clearly show that all the features used in the present system have the capability to improve the system's performance. By using POS information alone, we have achieved reasonable scores for baseline system. From the results we can clearly understand that the highest improvement is obtained by using the dynamic features. In general, the usage of suffix, prefix information, word, POS and chunk information significantly improves the overall performance of the system for all the languages.

6.1 Types of Errors in NE Identification

We have analysed the results where the system failed to identify the NEs. From our observation, mainly the errors are propagated due to the following reasons.

6.1.1 Ambiguity Due To Nested Entities

Ambiguity arises when one type of named entity occurring as part of another entity (nested entities). If an entity is considered as maximal entity, it belongs to one NE type and the entities occurred as part of it comes under different NE category. In such cases, the system might fail to identify the whole entity and mark only the part of that which comes under another NE category

6.1.1.1 Organization Names Identified As Location

Ta: [wamilYYnAtu]$_{LOC}$ curYrYulA valYarccik kalYYakam

En: Tamilnadu Tourist Development Corporation

Location names occurring as part of organization names are tagged as location in some instances. In the above example “Tamilnadu Tourist Development Corporation” is the name of a government organization name, which comes under NE type “ORGANIZATION”. But instead of identifying the whole entity the system can tagged the part of an entity “Tamilnadu” alone as “LOCATION”.

Ml: [keralYa]$_{LOCATION}$ koNZgrass
En: Kerela Congress

In the above example “Kerela Congress” is the name of a political party name, which comes under NE type “ORGANIZATION”. But instead of identifying the whole entity the system can tagged the part of an entity “Kerela” alone as “LOCATION”.

Hi: [uwwarAMcala] $_{\text{LOCATION}}$ sarakAra
En: Uttaranchal Government

In the example “Uttaranchal Government” is the name of a government organization name, which comes under NE type “ORGANIZATION”. But instead of identifying the whole entity the system can tagged the part of an entity “Uttaranchal” alone as “LOCATION”.

6.1.1.2 Person Names Identified As Other Types

Ta: [SrIrafkam]$_{\text{LOCATION}}$ peVrumAlY
En: Srirangam perumal

Location names occurring as part of person names are tagged as location. In this example, instead of tagging the maximal entity “SrIrafkam peVrumAlY” as PERSON system tagged the entity “SrIrafkam” as LOCATION and “peVrumAlY” as PERSON.

6.1.1.3 Location Names Identified As Other Types

Ta: [nAkarAjar]$_{\text{LOCATION}}$ thirukoyil
En: Nakarajar Temple

Person names occurring as part of "names of schools, hotels, places" are tagged as "person" by the system. In this example, instead of tagging the maximal entity “Nakarajar Temple” as LOCATION, the system tagged the entity “Nakarajar” as PERSON and “Temple” as LOCATION.

Ml: [vijay] $_{\text{PERSON}}$ cOka
En: [Vijay] $_{\text{PERSON}}$ Chowk

In the above example, the “Vijay Chowk“ is the place name. But, instead of tagging the maximal entity “Vijay Chowk” as LOCATION, the system tagged the entity “Vijay” as PERSON**.**

Ta: [cafkaranY] $_{\text{PERSON}}$ kovil
En: [sankaran] $_{\text{PERSON}}$ kovil

In this example, the “Sankaran Kovil“ is the place name. But, instead of tagging the maximal entity “Sankaran Kovil” as LOCATION, the system tagged the entity “Sankaran” as PERSON and “kovil” as LOCATION.

Hi: [seMta jAzna] $_{\text{PERSON}}$ carca

En: St. John Church

In the above example, the “St. John Church” is the name of a church. But, instead of tagging the maximal entity “St. John Church” as LOCATION, the system tagged the entity “St. John” as PERSON and “Church” as LOCATION.

6.1.1.4 Special Day Identified as Month

Ta:[kArwwikE]$_{MONTH}$ wIpam
En: karthikai Deepam

Month names occurring as part of "special days" are tagged as "month" by the system. In the example, the “Karthikai Deepam“ is the name of a featival. But, instead of tagging the maximal entity “ Karthikai Deepam” as SPECIAL DAY, the system tagged the entity “Karthikai” as MONTH.

6.1.2 Different Entities Occuring in Adjacent Positions

Ta: [kaNNamafkE peVrumAlYE]$_{PERSON}$ poojai seiththAL
En: Kannamangai perumal Pooja did

Person names occuring in adjacent positions are tagged as same entity. In the above example “kannamangai” and “Perumal” are two different entities. But instead of identifying them as separate entities, the system has wrongly tagged the both entities as a single entity.

6.1.3 Ambiguity between Common and Proper Nouns

In comparison with European languages, Indian Languages have more ambiguity between proper and common nouns. In our corpus, we observed that some common nouns found in the dictionary are occurring also as person names which create ambiguity issues.

Ta: intha kAranathirkAka *cakwi* pUjE cethAlY
En: For this reason(ADV) sakthi(N) pooja(N) did(V)
(Due to this reason sakthi did pooja)

Ta: pakEvarkalYE veVllum cakwiyEyum
En: Enemies(N)+acc win(V)+future power(N)+acc
Ta: waruvawAl iwwalawwu ammanY mikavum AkrocamAka
En: give(V)+future+cond this place(N)+gen Goddess(N) more(ADV)
Ta: *cakwi* ulYlYavar.
En: aggressive(N) power(N) has(v)+PRP
(The goddess of this place is very aggressive, as she gives power to win the enemies)

In the first example “*cakwi(sakthi)*” is a person who did pooja to God. In the second example *cakwi* is not a person name, it means “power”. Here in first example “*cakwi*” is a named entity, but in second example it is a common noun. When the same word occur as named entity and common noun ambiguity issue will occur.

6.1.4 Genitive Drop in Sequential Nes

The genitive case marker indicates the possessive form. The genitive drop in sequential named entities creates an ambiguity in some instances

Ta: inwiyan vanki ataiyARu kiLaiyil

En: Indian Bank Adaiyar branch+ (loc)

(Adayar branch of Indian Bank)

In the above example, “inwiyan vanki (Indian Bank)” is the name of a bank, it comes under the NE type ORGANIZATION and Adayar is the place name. But due to the genitive drop in “-yin or -utaiya” Indian Bank Adayar is tagged as a single entity.

The errors discussed in the error analysis can be resolve using post-processing rules and incorporating dictionaries for location and popular organization names for the respective languages. For example, if the keyterm such as “Temple” following the proper noun, then the “PERSON” should change to “LOCATION” tag.

Post Processing

To overcome this issue of nested entities, use of post processing heuristic rules is reliable and convenient. This post processing has also helped to improve numeric and time expression detection. We have used linguistic and heuristic rules for post processing. The constraints we applied were based on the Part-of-speech tag, noun phrase chunk and a set of key-terms. We explain below some of the post-processing heuristic rules used in this work.

Rule 1: To attain the Organization tag

-2 NNP|NNPC|NNC|NN+B-NP=1
-1 NNP|NN+I-NP=1
0 thoziRsAlai|sangam|vanki (NN) =1

The term “thoziRsAlai (industries), sangam (club), vanki (bank)” act as a keyterm to indicate the organization NEs. If the current token is “thoziRsAlai|sangam|vanki”, previous POS is a noun category(NNP,NNPC,NNC,NN) followed by beginning noun-phrase chunk(B-NP) and second previous POS from current token is a noun category followed by B-NP then the nested named entity is tagged as Organization.

Rule 2: To solve the ambiguity exists between Person and Location tag.

-2 NNP|NNPC|NNC|NN+B-NP=1
-1 NNP|NN+B-NP=1
0 koyil|kovil|sannathi|temple+NN =1

The term “koyil|kovil|sannithi” denotes the names of the temple. If the current token is “koyil|kovil|sannathi”, previous POS is a noun category followed by beginning noun-phrase chunk and second previous POS from current token is a noun category followed by B-NP then the nested

Rule 3: To get the numeric and time expressions

-1 QC=1 PERIOD
0 Keyterms denotes time and numeric expressions =1

The term “varutam|nAlY|ANtukalY|nAtkalY|mAwanworYum|nUrYrYANtu” indicates year|day|years|days|every month|century in English. The cue phrase “Ki.mi.” denotes Kilo meter in English. If the current token is one among the keyterms which denotes the numeric or time expressions and the previous POS specifies cardinal number then the named entity is tagged as the respective NE tag. The results

obtained after post-processing is given in Table 4. After post-processing Tamil language has obtained the precision of 82% and recall of 85% respectively.

Table 4.13: Results after post-processing

NE Type	Precision %	Recall %	F-score %
Tamil	82.12	85.32	83.68

7. Conclusion

The Tamil Named Entity Recognizer with fine grained tagset was explained in this paper. The POS patterns of named are analysed for Tamil and taken as the feature for CRFs. There are six NER models are generated using different feature combinations. The errors obtained in the baseline system are explained in detail. The error driven features are ignored and the six NE models are generated as part of extended system. The combination of features used in the model6 performs better than any other feature combinations and obtained the precision of 80.12% and recall of 80.10% respectively. The different types of errors occurred during the NE identification are explained with examples. The post-processing rules are implemented to overcome the issues and the overall f-score is further improved by 3%.

Acknowledgement

Authors thank *IMPRINT India initiative* for the support given to carry out this research.

References

1. Antony, J.B, & Mahalakshmi, G.S 2014, 'Named entity recognition for Tamil biomedical documents', Proceedings of the 2014 International Conference on Circuits, Power and Computing Technologies, ICCPCT-2014, pp. 1571-1577.
2. Chinchor, N 1998, 'Overview of MUC-7', Proceedings of the Seventh Message Understanding Conference (MUC-7), Fairfax, Virginia.
3. Grishman, R & Beth, S 1996, 'Message understanding conference-6: A brief history', Proceedings of the16th International Conference on Computational Linguistics, vol. 1.
4. Nadeau, D & Sekine, S 2007, 'A survey of named entity recognition and classification, Linguisticae Investigationes, vol. 30, no. 1, pp. 3–26.
5. Ng, H.T, Lim, C.Y & Foo, S.K. 1999 'A case study on inter-annotator agreement for word sense disambiguation', Proceedings of the SIGLEX99: Standardizing Lexical Resources.
6. Malarkodi C.S, Pattabhi, R.K & Sobha L 2012, 'Tamil NER–Coping with Real Time Challenges', Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages(MTPIL-2012), COLING, pp. 23-38.
7. Pattabhi, R.K, & Sobha L 2013 'NERIL: Named Entity Recognition for Indian Languages @ FIRE 2013–An Overview', FIRE-2013.
8. Pattabhi, R.K, Malarkodi C.S, Ram V.S & Sobha L 2014 'NERIL: Named Entity Recognition for Indian Languages @ FIRE 2014–An Overview', FIRE-2014.
9. Sobha, L & Vijay Sundar Ram R, 2007, 'Multilingual Place Name Tagger for Indian Languages', Proceedings of IJCAI Workshop on Cross Lingual Information Access (CLIA) - Addressing the Information Need of Multilingual Societies, Hyderabad, pp. 34 - 39.
10. Sobha L, Malarkodi, C.S, & Marimuthu, K 2013, 'Named Entity Recognizer for Indian Languages', ICON NLP Tool Contest.
11. Vijayakrishna, R. and Sobha, L. (2008). Domain focused Named Entity for Tamil using Conditional Random Fields. In IJNLP-08 workshop on NER for South and South East Asian Languages, Hyderabad, India. pp. 59-66

மெய்நிகர், மிகை மெய்நிகர் தொழில்நுட்பங்களில் ,புத்தாக்க கணினி தளங்கள், செயலிகள் மென் பொருட்கள் வழி தமிழ் பெரும் எதிர்கால ஆதாய அனுகூலங்கள்

சி.குணசேகரன்
புத்தாக்கப் பயிற்றுவிப்பாளர், பிக்சிபிட் பிரைவெட் லிமிட்டெட்,சிங்கப்பூர்
powerguna@gmail.com

தொடக்கம்

பிரபஞ்சங்களின் பிரதிபலிப்பில் தோன்றிய புவியுலகில்,நம்முள் நிறைந்திருக்கும் இயற்கையைப் பகுத்துணர்ந்து, செயற்கை அறிவியலை உட்புகுத்தி, 20ம் நூற்றாண்டில் தன்னிகர் அற்ற தொழில்புரட்சியால் உயர் வல்லமை பெற புத்தாக்க வாய்ப்புக்கள் பெறத் தொடங்கி இருக்கும் கணினி யுகத்தில் வாழும் உன்னத வல்லமையை மனிதன் பெற்றுள்ளான்.

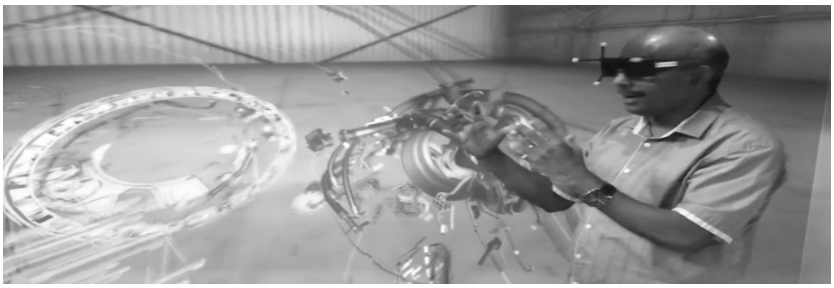
பறவை பறப்பதைக் கண்டு, விமானப் பயணக் கனவை நனவாக்கிட முடிந்தது. அது போல், நாம் இனி நினைத்ததை, கண்டு கேட்டு புரிந்து கொள்ள மெய்நிகர், மிகை மெய்நிகர் தொழில்நுட்பம் உதவும்.அது தமிழ் மொழி வளர்ச்சியில் உண்டாக்கி வரும் தாக்கங்களை இந்தப்படைப்பு விளக்கும்.

மெய்நிகர்(Virtual Reality), மிகை மெய்நிகர் தொழில்நுட்பம்(Augmented Reality -வரலாற்று பின்னோட்டம்

1980 ஆண்டுகளில் மத்தியில் திரு.ஜெரொன் லேனியர் (Mr.Jeron Lanior,CEO ,VPL Research,USA) அறிமுகத்தில் மெய்நிகர் அறிமுகமானது.ஆரம்பத்தில் திரைக் காட்சிகளை முப்பரிமாண படைப்பாக(simulated environments) காட்டவே இந்த முறை பயன்பாடு கண்டது.

எங்கும்,எதிலும் ,எந்த நேரத்திலும்,மனித மூளைக்குள் இல்லாத ஒரு இடம், காட்சி உணர்வுகளை உட் புகுத்தி , கற்றல், கற்பித்தல், கனிணிமூலம் அதிவேக இருவழி உறவாடலுக்கு வாய்ப்பு ஏற்பட்டுள்ளது.ஆப்பிள், கூகல், மைக்ரோசாப்ட், பேஸ்புக் போன்ற உலக நிறுவனங்கள் இன்று மெய்நிகரைப் பயன்படுத்தும் மிகை மெய்நிகர்த் தகவல்களை, படைப்புகளை வழங்க முனைந்துள்ளன.

மிகை மெய்நிகர், ஆரம்பத்தில் மெய்நிகரைச் சார்ந்திருந்தாலும் 1965ல் திரு,ஐவன் சண்டெர்லாண்ட் (Ivan Sunderland) கண்டுபிடிப்பால் புதுமாற்றம் கண்டு, இன்று நமது கற்பனைகளைத் தாண்டிய உலகுக்கும் ஈட்டுச் செல்லும் உன்னத வளர்ச்சியைப் பெற்றுள்ளது.



மெய்நிகர்(Virtual Reality), மிகை மெய்நிகர் தொழில்நுட்பம்(Augmented Reality) வேறுபாடுகள்

இத்தருணத்தில், இந்த இரு தொழில் நுட்பங்களில் முக்கிய வேறுபாடுகள் பற்றி அறிதல் நன்று.

மெய்நிகர்

இது முழுக்க முழுக்க கணினியால் உருவாக்கப்படும் காட்சி சூழலைச் சார்ந்திருக்கும்.360 ஒளி, ஒலி அமைப்பு உள்ள கற்பனைக் காட்சிகளை சிறப்பு கருவிகளை அணிந்து மட்டுமே உணர முடியும். அதி நவீன தொலை பேசிகளின் வரவு, மெய்நிகர் படைப்புகள் உருவாக்கத்திற்கு இன்னும் அதிக படைப்பாற்றல் வாய்ப்புகளைத் தந்துள்ளன.

மிகை மெய்நிகர்

கணினி விளையாட்டுச் சூழல் சார்ந்த செயலிகள் வழி, கைத்தொலைபேசிகள் சார்ந்த நடவடிக்கைகளுக்கு இந்தத் தொழில் நுட்பம் கை கொடுக்கிறது.மின்னியல் புத்தகம்,தொட்டுணரும் இணையப் படைப்புகள் என பல புதிய வெளிப்பாடுகளைக் காட்ட இது பயனளிக்கிறது.

தமிழ் வளர்க்கும் புத்தாக்கம்

உயரிய தொழில்நுட்பங்கள் இன்று பரவலாகக் கிடைப்பதால், நம் தமிழ் மொழி அண்மைய காலத்தில் கணினி உலகில் தனக்கென ஒரு தனியிடத்தை உருவாக்கிக் கொள்வதற்கு, புதுப் பயணம் தொடங்கி உள்ளது.

கூகல் அட்சென்ச்(Adsense),மைக்ரோசோட் தமிழ் , முகநூல்(Facebook AR) போன்ற பொதுத் தள அங்கீகரங்களினால், தமிழ் இப்போது பொருளீட்டும் மொழியாக உயர்ந்திருப்பதால் , புத்தாக்கம் நம் மொழிக்கு புது வாய்ப்புகளை வழங்க உள்ளது கண்கூடு.

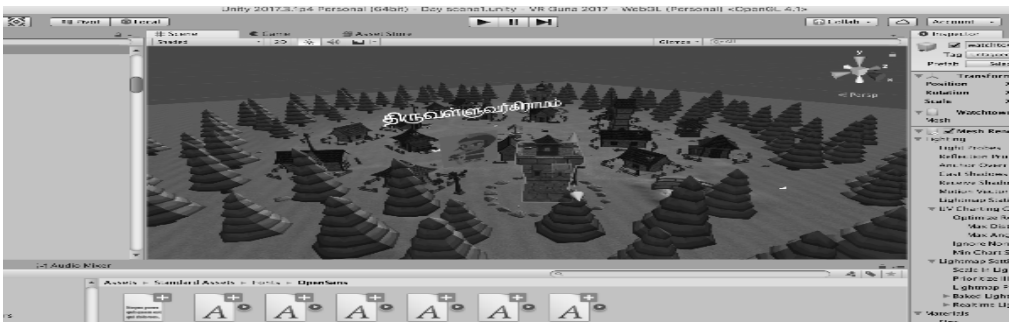
இயந்திர மொழியியலில்(Artificial Interligence) தமிழ் உட்புகுத்தும் ஆய்வுகள் நல்ல முன்னேற்றம் கண்டுள்ளன.சிங்கப்பூரில் AI Academy, A Star, Samsung Singapore,Microsoft, Facebook, Google போன்ற நிறுவனங்கள் தமிழை தமது தொழில்நுட்பங்களில் பயன்படுத்தும் மொழிகளுள் ஒன்றாக அங்கீகரித்துள்ளன.

உதாரணமாக கூகலில், தமிழ் குரல்வழி பயண வழிகாட்டி செயலாக்கம் கண்டுள்ளது. மைக்ரோசப்ட், பிற மொழியில் இருந்து தமிழுக்கு நேரடி மொழி பெயர்ப்பு செய்யும் முறையை அறிமுகம் செய்துள்ளது. முகநூலில் மெய்நிகர் காட்சிகள் அடங்கிய AR Studio, Spaces அறிமுகம் கண்டுள்ளது.

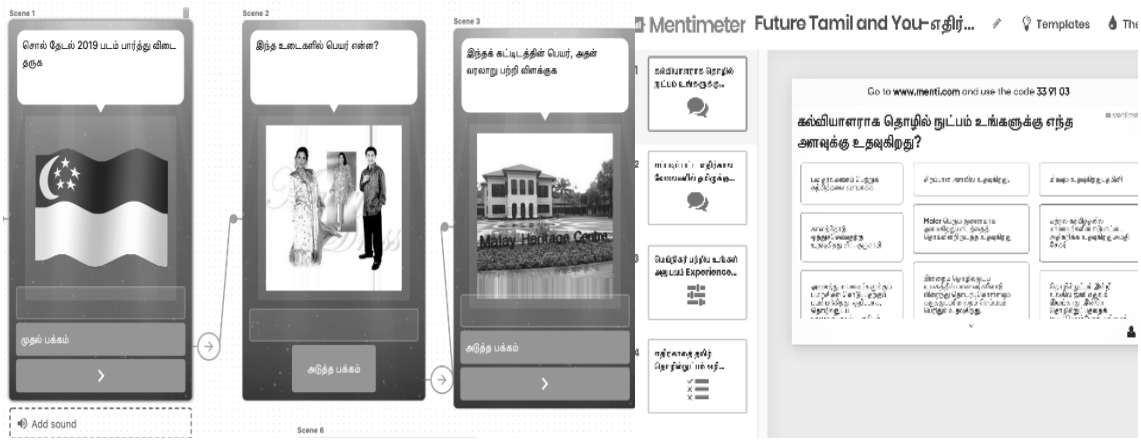
இயந்திர மொழி பேசும் (chat bot) அரட்டைக் கணிப்பொறி, தற்போது பிரபலமாகி வருகிறது. தமிழில் அது பிரபலமாகும் போது, பல வேலைகள் இன்னும் எளிதாக செய்து முடிக்க வாய்ப்புண்டு.

கூகல் அறிமுகம் செய்திருக்கும் ADsense, தமிழில் படைப்பாளர்களுக்கு ஒரு வரப்பிரசாதம்.தரமான சொந்தப் படைப்புகள் மூலம் நமது தமிழ் புதிய தலைமுறை தமிழர்களை விரைவாகச் சென்றடையலாம்.

தமிழில் மெய்நிகர் படங்கள் உருவாக்கம் காண்பதற்கு நல்ல வாய்ப்புகள் உண்டு. Blockchain மூலம் முதலீடுகள் கிடைக்கும் வகையில் முப்பரிமாண படங்கள் தமிழில் உருவாக்கும் வழிமுறைகள் இப்போது நமக்கு கிடந்துள்ளது.vuulr நிறுவனத்துடன் இணைந்து தமிழ் படைப்புகள் உருவாக்கும் கன்னி திட்டம் தொடங்கி உள்ளது.இதன் வழி தரமான தமிழ் காவியங்கள், புதினங்கள், படைப்புகள் ஒளி,ஒலி வடிவம் பெறுவதோடு, முதலீட்டாளர் இன்றி உலகத்தரம் வாய்ந்த கற்பனை, கணினி ஈடுபாடு உள்ளத் தமிழர்களால் படைக்க முடியும்.நமது கலை உலகம் போற்றும் புதுமை வார்ப்புகளாக உருவெடுக்க இது வழி வகுக்கலாம்.



தமிழ் மின்னூல்கள் உருவாக்கத்தில் மிகை மெய்நிகை பயன் படுத்தி வள்ளுவனை, பாரதியை, முப்பெரும் தமிழ் பேரரசர்களை நம் கண்முன் கொண்டுவர முடியும். இன்றைய இளையத் தமிழர்கள் ரசிக்கும் வகையில் கதை, கவிதை, படைப்பிலக்கியங்களை நாம் உருவாக்கலாம். நேரடியாக உரையாடல், ஒலி , ஒளி காட்சி அமைப்புகள் கணினி விளையாட்டுகள் கொண்ட புத்தகங்களை Unity, Vuforia மென்பொருள்கள் கொண்டு உருவாக்குதல் மூலம் தமிழ்ப் புத்தகங்களாகப் படைக்கலாம்.AR kit, AR Core போன்ற இலவச படைப்புத்தளங்களின் வருகை, தமிழில் இன்னும் அதிகமானப் படைப்புகளை உருவாக்கும் வாய்ப்புகளை நம் விரல் நுனிக்குத் தந்துள்ளன.

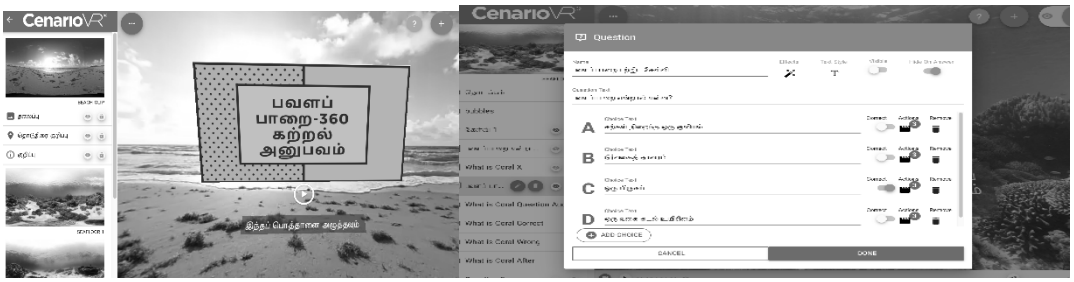


கற்றல் கற்பித்தல்-புத்தாக்க அணுகுமுறைகள்

கூறப்பட்ட இரு தொழில்நுட்பங்களையும் பயன்படுத்தி தமிழ் சார்ந்த பயிற்சித் திட்டங்கள் சிங்கப்பூரில் நடைமுறைப் படுத்தப்பட்டுள்ளன.

பாலர் கல்வி முதல் வாழ்நாள் கற்றல், முதியோர் பயிற்சிகள் வரை செயல்முறைகள் பயன்பாட்டில் உள்ளன.

சிங்கப்பூர் 200ம் ஆண்டு கொண்டாட்டங்களில் ஒரு கூறாக , தமிழ் பேசும் மெய் நிகர் செயலி,(MetaVerse AR) திறன்பேசிக் கருவிகளுக்காக உருவாக்கப்பட்டுள்ளது.அதில் குரல் ஒலிப்பதிவு(Voice Recording), முகப்பதிவு(Image Reconition), செய்தித் தகவல் பரிவர்த்தனை(Chatbot Interaction) ,பயன்பாட்டுத்தகவல் திரட்டு(Data Analytics) தட்டச்சு உள்ளீடு (Unicode Tying)என புத்தாக்க செயல் முறைகள் உட்புகுத்தப்பட்டுள்ளன.பல்லினக் கலாசாரங்களை அனைவரும் எளிதில் அறிந்து கொள்ளும் வகையில் தமிழ்ச் செயலி அமைந்துள்ளது.இதன் முக்கியத்துவம் கருதி சிங்கப்பூர் அரசின் உருவாக்க மானியம் வழி தரமானத் தமிழ்ச் செயலியாக உருவாக்கப்பட்டுள்ளது.



மெய்நிகர் காட்சி உருவாக்கம் வழி கற்பிக்கும் பாடத்திட்டம் வாழ்நாள் கற்றல் பயிற்சியின் ஒரு கூறாக உருவாக்கப்பட்டுள்ளது.

தமிழில் முப்பரிமாண காட்சிகள் மூலம் ஒலி,ஒளியாக்கம் படைக்கும் எளிய முறை செயல் படுத்தப்படுள்ளது.CenerioVR மேக ஊடக உருவாக்கும் தளம்(Cloud Computing) மூலம் தமிழ் தொட்டுணர் கருவி சார்ந்த பயிற்சிகள் உருவாக்கப்பட்டுள்ளன.

தமிழ் சார்ந்த பற்பல தகவல்களை கணினி விளையாட்டு(Unity Gamification), சமூக ஊடக இணையம் வழி கற்பித்தல்(Social media Interaction), இயந்திர மொழியியல் வழி தமிழ் படைப்பாற்றல்,(Robotic Tamil Computing), முப்பரிமாண கலைப் படைப்பு உருவாக்கம்(Artsvive Immersive Digitalisation) என பல திட்டங்கள் தற்போது சிங்கப்பூரில் செயல்படுத்தப் படுகின்றன.

உலகத்தரம் வாய்ந்த நிறுவனங்களோடு அனைவரும் பயனடையும் வகையில் பொருளாதார அடிப்படையில் வெற்றி பெறும் வகையில் நமது செயல்முறைகள் அமைந்துள்ளன. சிங்கப்பூர் அரசில் அளப்பரிய ஆதரவும் உதவிகளும் இதற்குப் பெரிது கைகொடுக்கின்றன.

வாய்ப்புகள் தடங்கல்கள்

தொன்மைமிக்க நம் தமிழ் மொழி, கணினியியலில் கடந்து வந்த பாதை சற்று சுனக்கமானதே. ஆங்கிலம் , சீனம் போல நமக்கென ஒரு இயங்குதளம் உருவாக்கும் தேவை இன்னும் உருவாகவில்லை. எழுத்துருக்கள் பல பரிணாமம் பெற்று, இப்போது யூனிகோட் ஒருங்கிணைப்பால் ஒரு நிலைப் முப்பரிமாணத்தில் தமிழ் பேச்சும் எழுத்துருக்கள் பயன்பாடும் வளர்ந்திருந்தாலும் ஒரு குறுகிய வட்டத்திற்குள்ளேயே பயனீட்டில் உள்ளன.பொருளாதார அடிப்படையில் உலகத்தரம் வாய்ந்த படைப்புக்களைத் தமிழில் உருவாக்கும் படைப்பாளர்கள் கைகோர்க்கும் முயற்சிகள் மேற்கொள்ளப்படலாம்.

இதைத் தாண்டி இயந்திர மொழி மூலம் தமிழ் மேம்படும் போது, பயன் மிக்க மென்பொருள்கள், படைப்புகள் உருவாக்க வாய்ப்புண்டு.

தொழில் ரீதியில், தமிழ் மூலம், உயர்வடையும் வாய்ப்புகள் இனி வரும் காலத்தில் பிரகாசமாகவே உள்ளன.

மருத்துவம், சுற்றுப்பயணத்துறை,அரசு சேவைகள், உயர் பாதுகாப்பு வேலைகள் , விவசாயம், வாழ்நாள்கற்றல், கற்பித்தல், தொலைதூரப் பயிற்சி, முப்பரிமாண படங்கள், புத்தாக்க எழுத்துப்படைப்புகள் என பல உலகலாவிய திட்டங்களில் தமிழை முன்னிலை படுத்த மெய் நிகர், எதிர் மெய் நிகர் நிச்சயம் கைகொடுக்கும்.

புத்தாக்கத் தமிழ், இன்னும் தரப்படுத்தப்படும் போது, அடுத்த நூற்றாண்டில், தமிழ்,செம்மொழியாக மட்டும் இல்லாமல், மின்னாக்கப் பொலிவுடன் எட்டுத்திக்கும் கொடிகட்டிப்பறக்கும் முத்தமிழாக உயரும் எனலாம்.


நிறைவாக

கற்காலம் முதல் ஓலைச்சுவடிகள் வழி, நமது உயிரில் கலந்த தமிழ், எதிர்காலத் தலைமுறைக்கும் பல புதுமைகளை உட்புகுத்தி இயல், இசை நாடகமாக, புத்தாக்கத் தொழில் நுட்பம் கொண்டு சிறந்த படைப்புக்களை உருவாக்க மெய்நிகர், எதிர் மெய்நிகர் செயல்முறைகள், இயந்திர மொழி பேசி, தமிழ் உயரிய மொழியாக உயர ஆராய்ச்சிகள் உலகளாவிய முறையில் தொடரும். உலகமெங்கும் பரவி இருக்கும் தமிழர்கள் அதிகமாகத் தமது மொழியைக் கணினியாக்கம் செய்யும் பழக்கம் இடம்பெறும் போது , நிச்சயம்,தமிழ் , பிற மொழிகள் போல் பொருளீட்டுவதோடு நல்வாழ்வுக்கும் வழி வகுக்கும் என தீர்க்கமாக நம்பலாம்.

Abstractive Summarization of Tamil Texts using Conceptual Graphs

Pattabhi RK Rao.
AU-KBC Research Centre,
MIT Campus of Anna University, Chennai
pattabhi@au-kbc.org

Abstract. The paper proposes an algorithm for obtaining abstractive summary of a given Tamil document based on Conceptual Graph model. Our approach first extracts conceptual graphs for a given document and from the graph the most frequently occurring concepts along with their relationships are taken, then summary of the document is generated. Here we have worked for a highly agglutinative and morphologically rich language, Tamil. We use the Conceptual Graph (CG) formalism as proposed by Sowa (Sowa, 1984) to represent the concepts and their relationships in the documents.

1. Introduction

The huge, enormous and continuous increase of digital data in internet, has made it a necessity for language computing (NLP) community to come up with a highly efficient automated text summarization tools. The research on text summarization was boosted by various shared tasks such as TIPSTER SUMMAC Text Summarization Evaluation task, Document Understanding Conference (DUC 2001 to 2007) and Text Analysis Conferences (TAC 2009). There are several methods for summarizing a text and we find that concept based summarization will be more semantically driven and give cohesion to the summary, which is automatically generated. ``A Summarizer is a system whose goal is to produce a condensed representation of the content of its input for human consumption" (Mani, 2001).In most of the methods used for automated summary generation, the end result is a collection of sentences which do not have connectivity to topic or we can say the cohesion of the text is not present. We are trying to bring in this cohesion to the summary through the conceptual graph based summarization.

Automated summarization is an important area of research in NLP which uses data mining technology. One of the popularly known earliest works on text summarization is by Luhn (Luhn, 1958). He proposed that frequency of a word in articles provide a useful measure of its significance. Significance factor was derived at sentence level and top ranking sentences were selected to form the auto abstract.

A variety of automated summarization schemes have been proposed in the last decade. NeATS (Lin & Hovy, 2002) is a sentence position, term frequency, topic signature and term clustering based approach and MEAD (Radev, 2004) is a centroid based approach. Iterative graph based Ranking algorithms, such as Kleinberg's HITS algorithm (Kleinberg, 1999) and Google's Page- Rank (Brin, 1998) have been traditionally and successfully used in web-link analysis, social networks and more recently in text processing applications. (Mihalcea & Tarau, 2004), (Mihalcea, 2004), (Erkan & Radev, 2004) and (Mihalcea, 2004) have been proposed for single document summary generation. All these methods are based on sentence ranking. All these works are majority for English and few for other European languages such as Spanish, Italian.

In the literature we find there are not many works done for Indian languages and especially for Tamil there are very few. One of the work in Tamil summarization of significance is by Sankar et al (2011). Sankar et al (2011) have worked on text summarization for Tamil documents based on graph based sentence ranking approach. In this they have used Levenshtein Distance (LD) is a measure of the similarity between two strings to calculate the weight of the sentences and further rank the sentences. Each sentence is represented as the node of the graph.

The paper is further organized as follows: The next section briefly introduces and describes about conceptual graphs. Section 3 describes our proposed methodology for summary generation. Section 4 describes the evaluation. Section 5 concludes the paper.

2. Background of Conceptual Graphs

A Conceptual Graph (CG) is a graph representation of logic based on the semantic networks of artificial intelligence and existential graphs of Charles Sanders Peirce. John Sowa states the purpose of conceptual graphs as ``to express meaning in a form that is logically precise, human readable and computationally tractable" (Sowa, 1984). Mathematically, a CG is a bipartite, directed, finite graph; each node in the graph is either a concept node or relation node. Concept node represents entities, attributes, states, and events, and relation node shows how the concepts are interconnected. A node (concept or relation) has two associated values: a type and a referent or marker; a referent can be either the single generic referent, or an individual referent. Thus a CG consists of a set of concept types and a set of relation types. (Sowa, 1976)

A conceptual graph is represented mainly in two forms viz., i) Display form and ii) Linear form. The display form uses the traditional graph form, where concept nodes are represented by rectangular boxes and relation nodes are represented by ovals. In the linear form concepts are represented by square brackets and relation nodes are represented using parenthesis. To represent these graphs internally inside the computer system we use a list data structure consisting of triplet value (c1, c2, r), where c1 is concept one and c2 is concept two and r is the relationship between the concepts c1 and c2. This triplet structure can be again represented using traditional matrix representation which is currently followed by information systems. The example below would give more insight into CGs.

Example 1: English Sentence: ``Marie hit the piggy bank with a hammer."

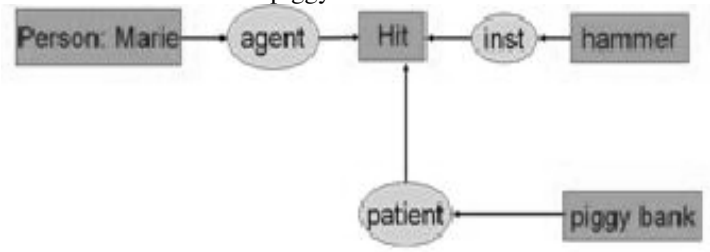


Figure 1: A Conceptual Graph (CG) - Example Sentence

The examples shown above are in display form. At an outer glance these conceptual graphs apparently look similar to a parser output. But actually it is not. The main differentiating aspect of CGs is that they are not restricted by any grammar formalism.

Example 2: Tamil Sentence (in Roman script)

Ta: naNRAka patiththAl nalla mathippeN peRalaam.
En: well read+cond good marks obtain+future
(If we read well, good marks can be obtained.)

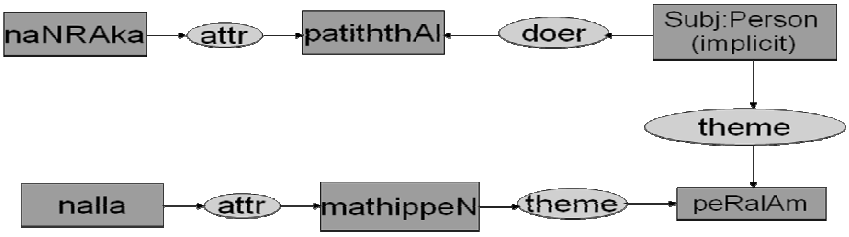


Figure 2: A CG of Tamil Example 2

Example 3: Tamil Sentence

Ta: mazai peythathaal nilam kuLirnthathu
En: Rain raining+cause land cool+past
(The land cooled because of rain.)

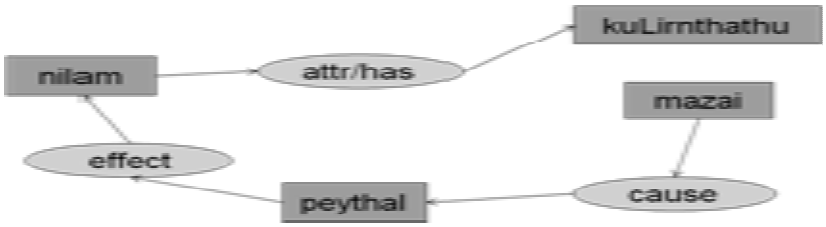


Figure 3: A CG of Tamil Example 3

CGs is not just syntactic analysis. CGs is flexible and not based on any grammar formalism. CGs has concept nodes and relation nodes. CGs does not describe just the governor and dependent relations, relation nodes show the relation between concepts of any nature and manner. These relation nodes capture the semantics. Parsing is a syntactic analysis of a sentence, identifying different components of the sentence and their syntactic roles, following grammar formalism. Parsing generally depicts dependency relations. CGs is a bipartite graph which could be easily translated to first order logic.

3. Proposed Methodology

Our approach involves two primary components. The first component is the extraction of conceptual graphs for the document. And generate a semantic network. The second component involves generation of maximal link chains of the semantic network to generate the summary of the document. The overall architecture of the system is shown in figure 4.

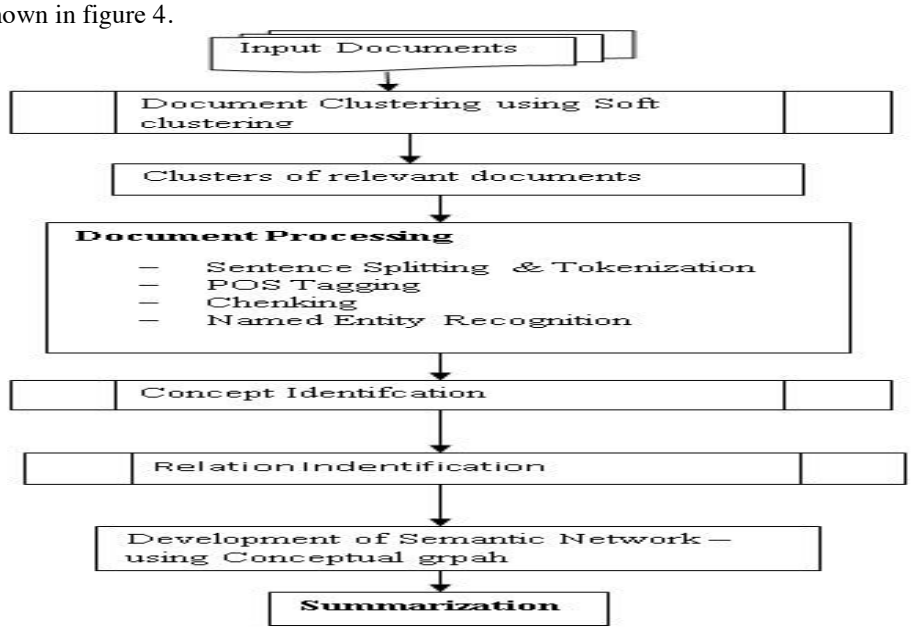


Figure 4: System Architecture of Proposed System

The input documents are collected from web. The aim of our present work is to generate document summary for a given set of documents. We perform soft clustering of the input documents to group the documents into several clusters. And for each cluster of related documents we generate a summary. After the documents are clustered each document is processed to obtain syntactic and semantic information using Natural Language Processing (NLP) tools. The sentence splitter and tokenizer are done using grammar and heuristic rules. We make use of POS tagger developed in house (Sobha et al, 2016) using Conditional Random Fields Approach and Similarly Chunker developed in house (Vijay & Sobha, 2010). We have used a named entity recognizer which was developed in house (Malarkodi & Sobha, 2013). This uses Conditional Random Fields (CRFs), a machine learning technique (Lafferty, 2001). After the NLP processing of the documents, we identify the concepts and their relationships. The next sub sections describe in detail the extraction of concepts and their relationships and formation of conceptual graph.

3.1 Extraction of Conceptual Graphs

3.1.1 Concept Identification

There are two sub-modules in this component; the first one is the concept identification module and second is the relation detection module. In the concept identification module, the concepts as defined in section 2 are automatically identified using deep learning. Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using a deep graph with multiple processing layers, composed of multiple linear and non-linear transformations (Srivastava et al., 2013) (Hinton et al, 2006). Deep learning is defined as a class of machine learning algorithms that use a cascade of many layers of nonlinear processing units for feature extraction and transformation. And learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts. In this work Restricted Boltzmann Machine (RBMs) which is one of the methods in deep learning is considered. In earlier work by Pattabhi and Sobha (Pattabhi and Sobha, 2015), they had described on identification of concepts and their relationships using RBMs. The same implementation is used for identifying the concepts in this work.

3.1.2 Relation Identification

Concepts are always interconnected and do not exist in isolation. Concepts are connected with each by various relationships. We need to identify these various relationships that exist between the concepts to form a conceptual graph which is a semantic network. The figure 5 shows the process flow diagram in the relation identification module.

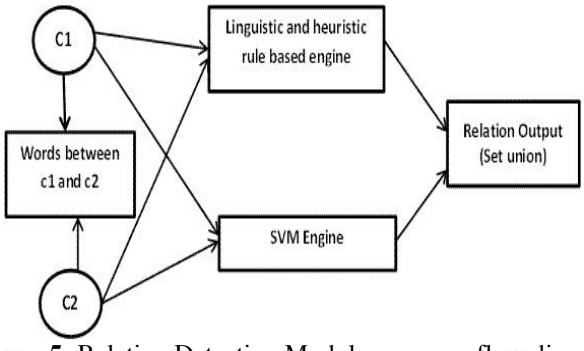


Figure 5: Relation Detection Module - process flow diagram

Relationship identification module uses a hybrid approach. Here we have two sub modules, rule based engine and SVM engine. The output of both engines is merged.

The linguistic rules are used initially to identify well defined relations. The linguistic rules use syntactic structure of the sentence. Some of the linguistic rules are described below:

(i) If concept c1 is a verb/verbal phrase and concept c2 is a noun/noun phrase and there are subordinators such as “after”, “later” before the c2 then these are markers of temporal relations. Using these temporal relationships one can infer senior-junior relationships, if this exists between two person concepts. For example in the sentence “John joined ABC corp after Marie”. This shows John is junior to Marie.

(ii) If concept c1 and c2 are connected by be verbs such as “is”, then there exists “is a” or “sub-type” relationship.

We have developed a preposition relation mapping table which defines different relations for each type of prepositions associated between verb-noun, noun-noun concepts. We also make use of Support Vector Machine (SVM) classifier to identify relations independent of the rule based engine. The output of SVM classifier and the output of the rule based engine are merged to get the set of all relations. In the SVM engine output we only consider those relations which get higher confidence score of more than 0.75 as valid relations. The features used for training SVM engine are the words and POS feature.

3.2 Summary Generation

The <concept-relation-concept> tuple obtained from is actual a bipartite graph consisting of two classes of nodes “concepts” and “relations” and form a CG. For a sentence many such tuples are obtained depending on the number of clauses. These are merged into the sub-graphs of the sentence to form a CG. Sub-graphs are merged by computing clique-sum. In this method two graphs are merged by merging them along the shared clique. A clique in a graph is a subset of vertices in which every two vertices are connected by an edge. Each tuple can be considered as a clique. We identify the shared cliques and merge them to form a unified network of the conceptual graph for the document. This complete CG is the semantic network of the documents. This is a kind of inheritance network, where the lower nodes correspond to more specific regularities and the upper nodes to more general ones. This hierarchy allows multiple inheritances. Thus we form semantic network of the document

From the above semantic network the document summary is generated. The document summary generation has the following two steps.

(i) Identify the clusters of longest chain of nodes in the graph.
(ii) Select the nodes that are in the longest chain to form summaries. Simple Sentences are formed containing those nodes and their relationships. And thus a natural language summary is generated.

4. Evaluation

We have used the ROUGE evaluation toolkit to evaluate the proposed algorithm. ROUGE, an automated summarization evaluation package based on N-gram statistics, is found to be highly correlated with human evaluations (Lin & Hovy, 2002). The evaluations are reported in ROUGE-1 metrics, which seeks unigram matches between the generated and the reference summaries. The ROUGE-1 metric is found to have high correlation with human judgments at a 95% confidence level, so this is used for evaluation. The present approach using CGs works with Rouge score of 0.4789. We have compared our work with the work of Sankar et al (2011). The same corpus as used by Sankar et al (2011) has been used in this work.

The corpus consists of 150 documents for which reference summaries have been manually created. The 150 documents were taken from online news articles. The reference summaries and the summaries obtained by our algorithm are compared using the ROUGE evaluation toolkit, which is presented in Table 1 below and also compares with the result of Sankar et al (2011). For each article, the proposed algorithm generates an abstractive summary consisting on an average of 5-7 sentences, approximately containing 80-100 words summary.

Table 1. Our System results – in comparison with earlier work on Tamil reported In the literature for document Summary generation

System/Algorithm reference	ROUGE 1 Score
Sankar et al (2011)	**0.4723**
Our Approach	**0.4789**

It is observed that the results of the present work are comparable with the state of the art. And this present approach is intuitive.

5. Conclusion

We have presented an algorithm for text mining to abstractive document summarization. Here we have generated document summaries using conceptual graphs. One of the main advantages is that the summary is coherent and also easy scalable. This approach can be adopted for any language, with a very minimal customization, since this uses conceptual graph principles of knowledge representations. We have tested our approach using same corpus or dataset as given by Sankar et al (2011). Though this dataset is very small but that is the one popularly used benchmark dataset for Tamil summarization. We obtained a ROUGE score of 0.4789, which is comparable with state of the art. As we can see from Table 1, this method has significantly improved over the results reported in the literature. The main objective of our work was to ascertain how capturing of structure of a sentence and thereby of the document would help in generating document summary. We found that it was very useful and got good results. The use of CGs helped in the capture of the structure and the semantics and helped in generating abstractive summary.

Acknowledgement

Authors thank *IMPRINT India initiative* for the support given to carry out this research.

References

1. Erkan and Radev D. 2004 Lexpagerank: Prestige in multi document text summarization. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, July.
2. Hinton G. and Salakhutdinov R. 2006 Reducing the dimensionality of data with neural networks. Science, 313(5786):504-507.
3. Kleinberg. 1999. Authoritative sources in the hyperlinked environment. Journal of the ACM, 46(5):604-632.
4. Lafferty J., McCallum A., and Pereira F., 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In Proceedings of the 18th International Conference on Machine Learning (ICML-2001), pp.282-289.
5. Lin and Hovy E. H. 2002. From Single to Multi document Summarization: A Prototype System and its Evaluation. In Proceedings of ACL-2002, 457-464.
6. Lin C. Y. 2004 ROUGE: A package for automatic evaluation of summaries. In Proceedings of the Workshop on Text Summarization Branches Out, Barcelona, Spain.
7. Lin C. Y. and Hovy E. 2003 Automatic evaluation of summaries using n-gram co-occurrence. In Proceedings of 2003 Language Technology Conference (HLT-NAACL 2003), Edmonton, Canada, 71-78.
8. Luhn H. P. 1958. The automatic creation of literature abstracts, IBM Journal of Research and development archive, 2(2):159-165.
9. Malarkodi C. S., and Sobha Lalitha Devi 2013. Automatic Identification of Named Entities in Tamil using CRF, In processings of International Seminar on Current Trends in Dravidian Linguistics, May 27-29
10. Mani I. 2001 Summarization Evaluation: An Overview, In Proceedings of NTCIR,

11. McKeownand K. and Radev D. 1995 Generating sum-maries of multiple news articles, In Proceedings of the 18th Annual International ACM, Seattle, WA, 74-82.
12. Mihalcea., 2004, Graph-based ranking algorithms for sentence extraction, applied to text summarization. In Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions (ACLdemo 2004), Barcelona, Spain.
13. Mihalcea R. and Tarau P. 2004 TextRank - bringing order into texts. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004), Barcelona, Spain.
14. Mihalcea R, Tarau P., and Figa E. 2004 PageRank on semantic networks, with application to word sense disambiguation. In Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004), Geneva, Switzerland.
15. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean 2013 Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR.
16. Pattabhi R. K . Rao., Sobha Lalitha Devi and Paolo Rosso 2013 Automatic Identification of Concepts and Conceptual relations from Patents Using Machine Learning Methods. In Proceedings of 10th International Conference on Natural Language Processing (ICON 2013), Noida, India, .
17. Pattabhi R. K. Rao. and Sobha Lalitha Devi. 2015 Automatic Identification of Conceptual Structures Using Deep Boltzmann Machines. In Proceedings of Forum for Information Reterival and Evaluation, ACM DL, Gandhinagar, India, 16-80.
18. Radev D., Hongyan Jing, Malgorzata Stys and Daniel Tam 2004 Centroid-based summarization of multiple documents, Information Processing and Management, 40:919-938.
19. Sankar K, Vijay Sundar Ram R and Sobha Lalitha Devi, 2011, Text Extraction for an Agglutinative Language, Problems of Parsing in Indian Languages, May 2011 Special Volume 11(5), in Language India.
20. Sowa J. F., 1976 Conceptual Graphs for a Data Base Interface. IBM Journal of Research and Development 20(4):336-357.
21. Sowa J. F. 1984 Conceptual Structures, Information Processing in Mind and Machine. Addison Wesley.
22. Srivastava N., Salakhutdinov R. 2013 and Hinton, G. E.: Modeling Documents with a Deep Boltzmann Machine. In Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI).
23. Sobha Lalitha Devi, Pattabhi RK Rao, Vijay Sundar Ram R and Malarkodi C.S. 2016. AUKBC Tamil Part-of-Speech Tagger (AUKBC-TamilPOSTagger2016v1). Web Download. Computational Linguistics Research Group, AU-KBC Research Centre, Chennai, India, May 2016.
24. Vijay Sundar Ram R, S.Menaka and Sobha Lalitha Devi. 2010. Tamil Morphological Analyser. In Morphological Analysers and Generators, (Ed.) Mona Parakh, LDC-IL, Mysore, 1 –18.
25. Vijay Sundar Ram R and Sobha Lalitha Devi. 2010. Noun Phrase Chunker Using Finite State Automata for an Agglutinative Language. In Proceedings of the Tamil Internet – 2010, Coimbatore, India, 218–224.

A Parser for Question-answer System for Tamil

Rajendran Sankaravelayuthan[1], M. Anandkumar[2], V. Dhanalakshmi[3],Mohan Raj S.N.[4]

[1]Amrita Vishwa Vidyapeetham, Coimbatore,
[2] NITK, Surathkal,
[3] Govt. Arts College for Women, Krishnagiri,
[4.] Amrita Vishwa Vidyapeetham, Coimbatore,
{Rajushush, manandinfo, dhanagiri, snmohanraj}@gmail.com

Abstract: The present paper aims to describe about the parser and a question-answer system we propose to build for Tamil. We have almost completed building of a parser for Tamil. We have made use of dependency grammar for the creation of parser for Tamil. We have made use of the Stanford parser modal to meet our mission. The Stanford parser is judged as a reliable parser for NLP applications. More overs as the dependency relations such as subject, object, indirect object and other relations of the participants of the verb are taken care of in the dependency parser it is a very useful approach for our purpose. We have made use of the POS tag sets, chunk sets and dependency sets of Stanford parser by modifying them to suit our purpose. We have made use of machine learning technique to prepare dependency parser for Tamil. Our question answer system is domain restricted and works on structured database. We are making use of the already available 50 thousand sentences in Tourist domain to find answer for the questions queried by the user.

1. Introduction

Though a number of syntactic parser is attempted for Tamil, a full-fledged syntactic parser is yet to be made. More over a highly workable dependency parser for Tamil for NLP application is the need of the day. One such attempt is made here. Machine learning approach is made use of for building dependency parser for Tamil. We have made use of Stanford parser as the modal for our parser. The Stanford POS tag sets, chunk sets and dependency tag sets are adopted with minor modification for Tamil. Parsing is followed by question-answer system. We have fifty thousand Sentences in Tamil in tourist domain. This has been made while creating English-Tamil parallel corpus for English Tamil MT system. These sentences will be parsed by the parser developed by us and kept in the data base. The question asked will be parsed by making use of the same parser. For the given question the correct answer will be extracted from the data base containing parsed out of fifty thousand sentences.

2. Parser

A natural language parser is a program that works out the grammatical structure of sentences. This groups words together (as "phrases") that could be considered as subject, object, indirect object or any other arguments (participants) of a verb. A number of syntactic parser has been developed for Tamil [8], [17] [24]. So far a full-fledged syntactic parser is yet to be developed. Chunker has been developed for Tamil to meet the requirement of MT systems. Kumara Shanmugam [17] developed a syntactic parser for Tamil. It has its own limitations. The parser at par with Stanford parser for English should be targeted for Tamil. Stanford Parser (SP) is a well-known and highly dependable parser for English. It gives only one output for a given sentence. It is a kind of decency parser and it provides dependency information which is very useful for any kind of NLP applications. It is a probabilistic parser which uses knowledge of the language gained from hand-parsed sentences and to try to produce the most likely analysis of new sentences. The statistical parser still makes some mistakes, but commonly works rather well. It is claimed that its development was one of the biggest breakthroughs in natural language processing in the 1990s. It is available for general use in online. We are adopting its POS tag sets, chunk sets, and dependency tag sets for our purpose my modifying them to suit our requirement.

3. Creation of Structured Domain Restricted Text

We are not going to develop a full-fledged general question answer retrieval system. It is not going to be an open-ended system. The domain is restricted to tourist domain. We have developed a tourist document consisting of 50 thousand sentences as an outcome of creation of parallel corpus between English and Tamil. This is going to be our domain restricted structured text.

4. Building Syntactic Parser Using Stanford Parser Labels

There are two steps in the system. First the questions will be processed by making use of the syntactic parser for Tamil. We will match the questions with the parsed output of the 50000 sentences in the data base of the tourist domain and select the correct answer. Named entity recognizer and co-reference recognizer will be plugged in to the parsing system.

For the given sentence the SP gives the tagged out put. This is be followed by parsing. Then universal dependencies are given. This is followed by enhanced universal dependencies. For example, the input sentence '*John hit the ball with the bat.*' is parsed as follows:

Your query

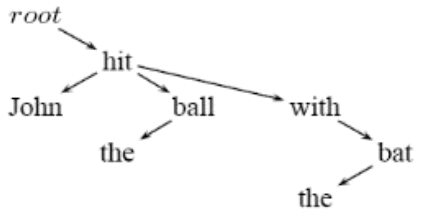
John hit the ball with the bat.

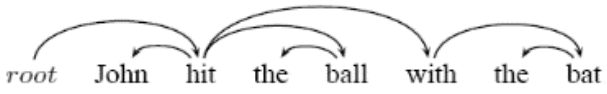
Tagging	**Parse**
John/NNP	(ROOT
hit/VBD	(S
the/DT	(NP (NNP John))
ball/NN	(VP (VBD hit)
with/IN	(NP (DT the) (NN ball))
the/DT	(PP (IN with)
bat/NN./.	(NP (DT the) (NN bat))))
	(. .)))

Universal dependencies

nsubj(hit-2, John-1)
root(ROOT-0, hit-2)
det(ball-4, the-3)
dobj(hit-2, ball-4)
case(bat-7, with-5)
det(bat-7, the-6)
nmod(hit-2, bat-7)

This can be diagrammatically represented as follows.



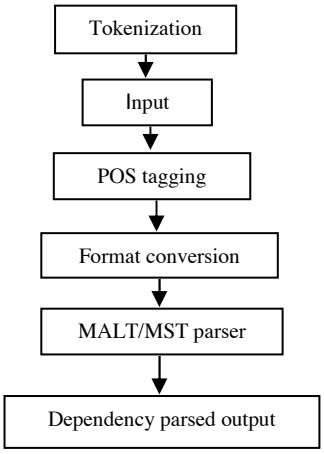


5. Machine Learning Approach to Dependency Parsing

Dependency parsing is one of the most important natural language processing works. Dependency parsing is the task of finding the dependency structure of a sentence. Dependency structure for a sentence is a directed graph originating out of a unique and artificially inserted root node. It is very important for machine translation and information extraction. The dependency parser helps to serve various NLP works: Relation Extraction, Machine Translation, Synonym Generation, Lexical Resource Augmentation, and Information Extraction.

The dependency parsing developed can serve the above requirement. The dependency parsing is can be implemented using rule based approach, but as it does not give the required result, an attempt is made here to full-fil our mission using machine learning approach. The machine learning approach tries to resolve the problems obtained in rule based approach. In rule based approach, a vast amount of linguistic knowledge is

required, whereas in machine learning approach only a reasonable amount of the knowledge is required. In rule based approach, if one rule fails we need to make changes in all others. In the machine learning approach, two kinds of algorithm are used: one is supervised algorithm and another is unsupervised algorithm. Here supervised learning is used for creating the model and using MST Parser Tool and MALT parser Tool, the dependency labels and the position of the head are obtained. For viewing the dependency tree structure Graphic tool is used. The results obtained by both the tools have been compared. The results obtained are very encouraging.



For Tamil parsing, data driven dependency parsers (MALT and MST) are applied for identifying the dependency graph. The general framework of
Tamil dependency parsing is illustrated in figure given below. This framework shows how the dependency structures for Tamil are identified using MALT and MST parsers.

The tokenized input sentence is fed to the POS tagger module, which is the primary process in parsing. The POS tagged sentences are given to Chunker module. The processed sentences are converted into the required format for Malt/MST parser. A PERL program is used for this conversion process.

5.1 POS Tagging

Parts of speech (POS) tagging is assigning grammatical classes i.e. appropriate parts of speech tags to each word in a natural language sentence. The POS tagging here is done using machine learning which makes it simple. There are two kinds of machine learning, one is supervised and the other is unsupervised learning. Here we use a supervised method of learning. In this method, we require pre-tagged Part of speech corpus to learn information about the tag set, word-tag frequencies, rule sets etc. The accuracy of the models generally increases with the increase in size of the corpus. Support Vector Machine (SVM) which is one of the powerful machine-learning methods is used here. A model is created for a considerable size of Tamil data and using this trained model the untagged data are tested. The SVM Tool for POS tagging gives laudable accuracy. For getting dependency parsing output, we need POS tagged structure. We make use of the following tag set adopted from Stanford parser for POS-tagging which is tabulated below.

S.N	TAG	DESCRIPTION	S.N	TAG	DESCRIPTION
1	<NN>	NOUN	16	<CNJ>	CONJUNCTION
2	<NNC>	COMPOUND NOUN	17	<CVB>	CONDITIONAL VERB
3	<NNP>	PROPER NOUN	18	<QW>	QUESTION WORDS
4	<NNPC>	COMPOUND PROPER NOUN	19	<COM>	COMPLIMENTIZER
5	<ORD>	ORDINALS	20	<NNQ>	QUANTITY

					NOUN
6	<CRD>	CARDINALS	21	<PPO>	POSTPOSITIONS
7	<PRP>	PRONOUN	22	<DET>	DETERMINERS
8	<ADJ>	ADJECTIVE	23	<INT>	INTENSIFIER
9	<ADV>	ADVERB	24	<ECH>	ECHO WORDS
10	<VNAJ>	VERB NON FINITE ADJECTIVE	25	<EMP>	EMPHASIS
11	<VNAV>	VERB NON FINITE ADVERB	26	<COMM>	COMMA
12	<VBG>	VERBAL GERUND	27	<DOT>	DOT
13	<VF>	VERB FINITE	28	<QM>	QUSTION MARK
14	<VAX>	AUXILIARY VERB	29	<RDW>	REDUPLICATION WORDS
15	<VINT>	VERB INFINITE			

Explanation of POS tags:

NN (noun): The tag NN is used for common nouns without differentiating them based on the grammatical information.

3. nalla kuzandai <NN>
'good child'

NNC (compound noun): Nouns that are compound can be tagged using the tag NNC.

4. kaN <NNC> vali <NNC>
'eye pain'

NNP (Proper Nouns): The tag NNP tags the proper nouns.

5. jaaN<NNP> angke niR-kiR-aan.
John there stand-PRES-3MS
'John is standing there'

NNPC (Compound Proper Nouns); Compound proper nouns are tagged using the tag NNPC.

6. aTal <NNPC> pikaari <NNPC> vaajpaayi <NNPC>
'Atal Bihari Vajpayi'

ORD (Ordinal): Expressions denoting ordinals will be tagged as ORD.

iraNTaavatu <ORD> kuzandtai.
'Second child'

CRD (Cardinal): Cardinal tag tags the cardinals like *onRu* 'one', iraNTu 'two', *muunRu* 'three' etc in the language as CRD.

7. iraNTu <CRD> puttakangkaL
'two books'

PRP (Pronoun): All pronouns are tagged using the tag PRP.

8. en <PRP> viiTu
'my house'

ADJ (Adjective): All adjectives in the language will be tagged as ADJ.

9. azakaana <ADJ> paaTTu
'beautiful song'

ADV (Adverb): Adverbial tag marks the adverbs in the language as ADV.

10. avan veekamaaka <ADV> ooT-i-kkoNTirukkiR-aan.
He fast run-PRES-CONT-3MS
'He is running fast'

VNAJ (Verb Non-finite Adjective): Non-finite adjectival forms of the verbs are tagged as VNAJ.

11. va-ndt-a <VNAJ> paiyan
come-PAST-ADJ boy
'the boy who came'

VNAV (Verb Nonfinite Adverb): Non-finite adverbial forms of the verbs are tagged as VNAV.

12. tuungk-i <VNAV>.ezu-ndt-aan
come-PAST-ADV go-PAST-3MS
'having come went'

VBG (Verbal Gerund): All gerundival forms of the verbs are tagged as VBG.

13. avan varu-kiR-atu <VBG>
he come-PRES-NOM
'that he is comming'

VF (Verb Finite): VF tag is used to tag the finite forms of the verbs in the language.

14. avan teervu ezhut-in-aan <VF>.
He exam write-PAST-3MS
'He wrote the examination'

VAX (Auxiliary Verb) : VAX tag is used to tag the auxiliary verbs in the language.

15. avan teervu ezut-i-koNTirikkiR-aan <VAX>.
He exam write-VNAV-CONTINUOUS ASPECT-3MS
'He is writing the examination'

VINT (Verb Infinite): The infinitive forms of the verbs are tagged as VINT in the language.

16. avan enn-aik kaaNa <VINT> va-ndt-aan
he I-ACC see-VINT come-PAST-3MS
'He came to see me'

CNJ (Conjuncts, both co-ordinating and subordinating): The tag CNJ can be used for tagging co-ordinating and subordinating conjuncts.

17. raaman-um kaNNan-um maRRum <CNJ> palar-um ingkee <ADV> irukkiR-aarkaL.
Raman-CNJ Kannan-CNJ CNJ many persons-CNJ here be-PAST3PHP
'Raman, Kannan and others are present here'
18. avan illaaViTTaal <CNJ> avaL varu-v-aaL
he or-CNJ she come-FUT-3FS
'He or she will come'

CVB (Conditional Verb): The conditional forms of the verbs are tagged as CVB.

19. avan-ai kaN-T-aal <CVB> pootum.
He-ACC see-CON enough
'It is enough to see him'

QW (Question Words): The question words in the language like *yaar* 'who', *endtaa* 'which' etc are tagged as QW.

20. yaar <QW> va-ndt-atu?
Who come-PAST-NOM
'Who came'

COM (Complimentizer): Complementizers are tagged as COM in the language.

21. avan avaL varu-v-aaL enRu <COM> kuuR-in-aan
He come-FUT-3FS that say-PAST-3MS
'He said that she would come'

NNQ (Quantity Noun): Quantitative nouns are tagged as NNQ in the language.

22. enakku konjca <NNQ> neeram taa.
me some time give
'Give me some time'

PPO (Postposition): All the Indian languages including Tamil have the phenomenon of postpositions. Postpositions are tagged using the tag PPO.

23. naan viiTu varai <PPO> varu-v-een.
I house upto come-FUT-1S
'I shall come up to the house'

DET (Determiners): The determiners in the language are tagged as DET.

24. andta <DET> kuzandtai
'that child'

INT (Intensifier) : Intensifier is used for intensifying adjectives or adverbs in a language. They are tagged as INT.

25. mika<INT> nalla kuzandtai
'Very good child'

ECH (Echo words): Echo words are common in Tamil language. They are tagged as ECH.

26. puli kili <ECH>
'tiger or that which resembles tiger'

EMP (Emphasis): The emphatic words in the language are tagged as EMP.

27. naan maTTum<EMP> varu-v-een.
'I only come-FUT-1S'
'Only I will come.'

COMM: The tag COMM tags the comma in a sentence.

28. cennai ,<COMM> tanjcaavuur , <COMM> maturai.
'Chennai, Thanjavur, Madurai'

DOT: The tag DOT tags the dots in the sentences.

29. naan angkee poo-n-een .<DOT>
I there go-PAST-1S
'I went there'

QM (Question Mark): The question marks in the language are tagged using the tag QM.

30. nii engkee poo-n-aay? <QM>
you where go-PAST-2S
'Where did you go?'

RDW (Reduplication Words): The reduplicated words are tagged as RDW.

31. mella mella <RDW>
'slowly slowly'

5.1.1 Training Corpus

The training data for POS-tagging is a two column format. First column contains input sentence and second column contains the POS tagged output. The following are the two examples.

32. avan oru kuTai vaangk-in-aan.		33. avan piyaanoo vaaci-kkiR-aan.	
he one umbrella buy-PAST-3MS		he piyaanoo paly-PRES-3MS	
'He bought an umbrella.'		'He is playing piano.'	
avan	<PRP>	avan	<PRP>
oru	<DET>	piyaanoo	<NN>
kuTai	<NN>	vaacikkiRaan	<VF>
vaangkinaan	<VF>	.	<DOT>
.	<DOT>		

A Model is created using this corpus data will be used for testing.

5.1.2. Testing sentences

The input sentences are aligned in a column manner using a program and then given to the SVM Tool for POS-tagging. Using the trained model the input sentences are tagged. The following is an example.

34. naan oru pazam paRikk-a mara-ttil eeR-in-een.
I one fruit pluck-PAST-INF tree-LOC claim-PAST-1S
'I climbed the tree to pluck a fruit'

Alignment Output (POS Tagging Input):	POS Tagged Output:
naan	naan <PRP>
oru	oru <DET>
pazam	pazam <NN>
paRikka	paRikka <VINT>
marattil	marattil <NN>
eeRineen.	eeRineen <VF>
	. <DOT>

5.2. Chunking

Chunking is an efficient and robust method for identifying short phrases in text, or chunks. The notion of phrase chunking is proposed by Abney. Chunking is considered as an intermediate step towards full parsing. After POS-tagging, the next step is chunking, which divides sentences into non-recursive inseparable phrases. A chunker finds adjacent, non-overlapping spans of related tokens and groups them together into chunks. Chunkers often operate on tagged texts, and use the tags to make chunking decisions. A subsequent step after tagging focuses on the identification of basic structural relations between groups of words. This is usually referred to as phrase chunking.

Chunking is comparatively easier for Indian languages than POS-tagging. The output of POS tagger is the input to the chunker. Chunking has been traditionally defined as the process of forming group of words based on local information. Hence, identifying the POS-tags and chunk-tags for the words in a given text is an important aspect in any language processing task. Both are important intermediate steps for full parsing. The word chunking tells something about how it is used for identifying short phrases or chunks in a text. Chunks are non-overlapping spans of text, usually consisting of a head (such as a noun) and the adjacent modifiers and function words (such as adjectives and determiners). A typical chunk consists of a single content word surrounded by a constellation of function words. Chunks are normally taken to be a non-

recursive correlated group of words. Tamil being an agglutinative language has a complex morphological and syntactical structure. It is a relatively free word order language, but in the phrasal and clausal construction it behaves like a fixed word order language. So the process of chunking in Tamil is less complex compared to the process of POS-tagging. We followed the guidelines mentioned in AnnCorra: Annotating Corpora Guidelines for POS and chunk annotation for Indian Languages while creating our tag set for chunking. Our customized tag set contains ten tags and is tabled below.

S.No.	Tag	Tag name	Possible POS Tags
1	NP	Noun Phrase	NN, NNP, NNPC, NNC, NNQ, PRP, INT, DET, CRD, ADJ, ORD
2	AJP	Adjectival Phrase	CRD,ADJ
3	AVP	Adverbial Phrase	ADV, INT, CRD
4	VFP	Verb Finite Phrase	VF, VAX
5	VNP	Verb Nonfinite Phrase	VNAJ, VNAV, VINT, CVB
6	VGP	Verb Gerund Phrase	VBG
7	CJP	Conjunctional	CNJ
8	COMP	Complimentizer	COM
9	PP	Post Position	PPO,NN
10	.?	Symbols	O

5.3. Dependency Parsing

Parsing is actually related to the automatic analysis of texts according to a grammar. Technically, it is used to refer to the practice of assigning syntactic structure to a text. It is usually performed after basic morphosyntactic categories have been identified in a text. Based on different grammars parsing brings these morphosyntactic categories into higher-level syntactic relationships with one another. The dependency structure of a sentence is defined by using dependency labels and dependency head. The following is the table for dependency tags used in the present system.

S.No.	**Tag**	**Description**
1	**<ROOT>**	**Head Word**
2	<NSUB>	Nominal Subject
3	<DOBJ>	Direct Object
4	<IOBJ>	Indirect Object
5	<.MOD>	Modifier
6	<CLSUB>	Clausal Subject
7	<CLDOBJ>	Clausal Direct Object
8	<CLIOBJ>	Clausal Indirect Object

9	<SYM>	Symbols
10	<X>	Others
11	<XCL>	Clause Boundary

Explanation of the Tag set

Noun Phrase (NP)

Noun Chunks will be given the tag NP. It includes non-recursive noun phrases and postpositional phrases. The head of a noun chunk would be a noun. Noun qualifiers like adjective, quantifiers, determiners will form the left side boundary for a noun chunk and the head noun will mark the right side boundary for it. Example for NP chunk is given below:

35.avaL oru azakaana peN.
she one beautiful woman
'She is a beautiful woman'

avaL	<PRP>	<B-NP>
oru	<DET>	<B-NP>
azakaana	<ADJ>	<I-NP>
peN	<NN>	<I-NP>
.	<DOT>	<O>

Adjectival Phrase (AJP)

An adjectival chunk is tagged as AJP**.** This chunk will consist of all adjectival chunks including the predicative adjectives. However, adjectives appearing before a noun will be grouped together with the noun chunk. It can be seen from the example for the noun phrase. Example for ADJ Phrase is given below:

36. palavakai aaraayccikkaaka ndaaTukaL tuNaikkooLkaL-ai viNveLi-kku anuppu-kinR-ana.
Different-type-ADJ research-ADV countries satellites-ACC space-DAT send-PAST-3NP
'The countries are sending satellites to space for different types of research works.'

palavakai	<ADJ>	<B-AJP>
aaraayccikkaaka	<ADV>	<B-ADP>
ndaaTukaL	<NN>	<B-NP>
tuNaikkooLkaLai	<NN>	<B-NP>
viNveLikku	<NN>	<B-NP>
anuppukinRana	<VF>	<B-VFP>
.	<DOT>	<O>

Adverbial Phrase (ADP)

Adverbial chunk is tagged in accordance with the tags used for POS tagging. It is tagged as AVP**.** An example for ADP is given below.

37. ndaan inRu cennai-kkup pookiReen.
I today Chennai go-PRES-1S
'I am going to Chennai today.'

ndaan	<PRP>	<B-NP>
inRu	<ADV>	<B-AVP>
cennaikkup	<NNP>	<B-NP>
pookiReen	<VF>	<B-VFP>
.	<DOT>	<O>

Conjunction

Conjunctions are the words used to join individual words, phrases, and independent clauses. The conjunctions are labelled as CJP. An example is given below.

38. raaNikku veLLimeTal kiTai-tt-atu enRaalum tangkameTal kiTaikkavillai
Rani-DAT silver medal get-PAST-3NS CNJ gold medal get-not
'Though Shaini got silver medal, she did not get gold medal'

raNikku	<NNP>	<B-NP>
veLLimeTal	<NN>	<B-NP>
kiTaittatu	<VF>	<B-VFP>
enRaalum	<CNJ>	<B-CJP>

tangkameTal	<NN>	<B-NP>
kiTaikkavillai	<VF>	<B-VFP>
.	<DOT>	<O>

Complimentizer

Complimentizers are the words equivalent to the term subordinating conjunction in traditional grammar. For example, the word *that* is generally called a Complimentizer in English. Complimentizer is tagged in accordance with the tags used for POS tagging. It is tagged as COMP.

39. uttaravai maaRRivaikk-a muTiy-aatu enRu vazakkaRinjar con-n-aar.
order postpone-INF possible-not COM advocate say-PAST
'The advocate said that the order cannot be postponed.'

uttaravai	<NN>	<B-NP>
maaRRivaikka	<VINT>	<B-VNP>
muTiyaatu	<VAUX>	<B-VFP>
enRu	<COM>	<B-COMP>
vazakkaRinjar	<NN>	<B-NP>
con-n-aar	<VF>	<B-VFP>
.	<DOT>	<O>

Verb Finite Phrase (VFP)

Verb chunks are mainly classified into verb finite chunk and verb non-finite chunk. Verb finite chunk includes main verb and its auxiliaries. It is tagged as VFP. An example of VFP chunk is given below.

40. avarkaL veelainiRuttam ceyy-a avan-ai azai-tt-anar.
They strike conduct-INF they-ACC invite-PAST-3HP
'They invited him to conduct the strike.'

avarkaL	<PRP>	<B-NP>
veelainiRuttam	<NN>	<B-NP>
ceyya	<VINT>	<B-VNP>
avanai	<PRP>	<B-NP>
azaittanar	<VF>	<VFP>
.	<DOT>	<O>

Verb Non-finite Phrase (VNP)

Non-finite verb comprise all the non-finite form of verbs. There are four non-finite forms in Tamil and they are relative participle form, adverbial participle form, conditional form and infinitive form. They are tagged as VNP. An example of VNP chunk is given below.

41. avaL 58 nimiSa-ttil ooT-i ett-in-aaL.
She 58 minutes-LOC run-VNAV reach-PAST-3FS
'He reached by running in 58 minutes'

avaL	<PRP>	<B-NP>
58	<CRD>	<B-NP>
nimiSattil	<NN>	<I-NP>
ooTi	<VNAV>	<B-VNP>
ettinaaL	<VF>	<B-VFP>
.	<DOT>	<o>

Verb Gerundial Phrase (VGP)

Gerundial forms are represented by a separate chunk. They are tagged as VGP. An example of VGP chunk is given below.

42. avanukku paTam varai-v-at-il atika viruppam uNTu.
he picture draw-PRE-NOM-LOC more interest is
'He has more interest in drawing pictures'

avanukku	<PRP>	<B-NP>
paTam	<NN>	<B-NP>
Varaivatil	<VBG>	<B-VGP>
Atika	<ADJ>	<B-NP>
Viruppam	<NN>	<I-NP>

uNTu	<VAX>	<B-VFP>
.	<DOT>	<O>

Symbol (O)

Special characters like Dot (.) and question mark (?) are tagged as O. Comma is tagged with the preceding tag.

43. aRaiy-il meecaikaLoo naaRkaalikaLoo illaatataal peritaakat teri-kiR-atu
room-LOC tables chairs not-having big-ADV appear-PRES-3NS
'As there are no tables and chairs in the room it appears big.'

aRaiyil	<NN>	<B-NP>
meecaikaLoo	<NN>	<B-NP>
naaRkaalikaLoo	<NN>	<B-NP>
illaatataal	<VNAV>	<B-VNP>
peritaaka	<ADV>	<B-AVP>
tooRRukiRatu	<VF>	<B-VFP>
.	<DOT>	<O>

5.3.1. Dependency Head and Dependency Relation

The parent child relation is specified using the arc. These arcs are symbolically represented using the position of the parent i.e. the number; this is explained below using an example.

44. raaman kaNNan-ukku oru pazam koTu-tt-aan.
Raman Kannan-DAT one fruit give-PAST-3MS
'Raman gave a fruit to Kannan'

1	2	3	4	5	6
raaman	kaNNanukku	oru	pazam	koTuttaan	.
<NNP>	< NNP>	<DET>	<NN>	<VF>	<DOT>

1	raaman	5	<NNP>	<NSUB>
2	kaNNanukku	5	<NNP>	<IOBJ>
3	oru	4	<DET>	<X>
4	pazam	5	<NN>	<DOBJ>
5	koTuttaan	0	<VF>	<ROOT>
6	.	5	<DOT>	<SYM>

This can be explained in the following way: *raaman* is the child of the parent *koTuttaan* which is in position 5; *kaNNanukku* is the child of the parent *koTuttaan* which is in the position 5; *oru* is the child of the parent *pazam* which is in position 4; *pazam* is the child of the parent *koTuttaan* which is in the position 5. *koTuttaan* which is the ROOT is in position 5; "." is the child of the parent *koTuttaan* which is in position 5. The children are linked to the parent by arcs and the arcs are labelled accordingly as NSUB, IOBJ, X, DOBJ, ROOT and SYM.

5.3.2. MALT Tool

The tool used for Dependency Parsing is the MALT Parser Tool. MALT stands for Models and Algorithms for Language Technology. It has been developed by Johan Hall, Jens Nilsson and Joakim Nivre at the Vaxjo University and Uppsala University of Sweden. MALT Parser is a system for data-driven dependency parsing. The parser can be used to induce a parsing model from the training data and to parse new data using the induced model. The parser uses the transition based approach to parse the new data. Transition based parsing builds on idea that parsing can be viewed as a sequence of state transitions between states and this approach uses a greedy algorithm. Parsers built using MALT Parser have achieved a high state-of-the-art accuracy for a number of languages.

There are 10 features in the MALT parser. The ten features are listed as follows: 1. Word Index, 2. Word, 3. Lemma, 4. Coarse Parts of Speech tag, 5. Parts of Speech Tag, 6. Morphosyntactic Features, 7. Dependency Head, 8. Dependency Relation, 9. Phrasal Head, 10. Phrasal Dependency Relation. These features are user defined for the training data and the features which are not defined are marked null

represented with the symbol "_". In our model, we have considered the following features: 1. Word Index, 2. Word 3. POS tag, 4. Chunk Tag, 5. Dependency Head, 6. Dependency Relation, The rest of the features is marked „_".

5.3.2.1. Training

The system is trained with more than 10,000 data which contains around 2000 sentences each of different patterns. This covers almost all the patterns for simple sentences and complex sentences of smaller length. The training data has the word id, word, pos tag, chunk tag, dependency head and dependency relation. The other columns are null and are denoted by an „_" (Underscore). The training data format is given below.

ID1	W1	P1	C1	_	_	H1	D1	_	_
ID2	W2	P2	C2	_	_	H2	D2	_	_
ID3	W3	P3	C3	_	_	H3	D3	_	_
.									
.									
.									
IDn	Wn	Pn	Cn	_	_	Hn	Dn	_	_

Here, ID refers to word index, W refers to word, P refers to parts of speech tag, C refers, to chunk tag, H refers to dependency read, and D refers to dependency relation. With this training data format, a model is developed. An example of the training data is given below.

45. raaman oru azaiipitaz koTuttuviTTu avan-ai viiTT-ukku azai-tt-aan

Raman one banana give-VNAV he-ACC house-DAT invite-PAST-3MS

'Having given banana to him, Raman invited him to the house.'

1	raaman	<NNP>	<B-NP>	-	-	4	<CLIOBJ>	-	-
2	oru	<DET>	<B-NP>	-	-	3	<X>	-	-
3	azaippitaz	<NN>	<I-NP>	-	-	4	<CLDOBJ>	-	-
4	koTuttuviTTu	<VNAV>	<B-VNP>	-	-	7	<XCL>	-	-
5	avanai	<PRP	<B-NP>	-	-	7	<DOBJ>	-	-
6	viiTTukku	<NP>	<B-NP>	-	-	7	<X>	-	-
7	azaittaan	<VF>	<B-VFP>	-	-	0	<ROOT>	-	-
8	.	<DOT>	<O>	-	-	7	<SYM>	-	-

5.3.2.2. Testing

The test data to be given to the malt parser has the Word ID, Word, pos tag and the Chunk tag. The remaining columns are given null. Testing input format is given below.

ID1	W1	P1	C1	_	_	_	_	_	_
ID2	W2	P2	C2	_	_	_	_	_	_

In the testing input format the head and dependency relation are given as NULL. The output data for the above test input data will be as follows.

ID1	W1	P1	C1	_	_	H1	D1	_	_

The head and dependency relation are obtained. It should also be noted that the training and the test data should have same features. An example of the input test data is given below.

46. raaman kaNNanukku uNavu koTuttuviTTu poonaan
Raman Kannan-DAT food give-VNAV go-PAST-3MS
'Having given Kannan the food, he went away'

1	raaman	<NNP>	<B-NP>	-	-	-	-	-
2	kaNNanukku	<NNP>	<B-NP	-	-	-	-	-

3	uNavu	<NPN>	<B-NP>	-	-	-	-	-	
4	koTuttuviTTu	<VNAV>	<B-VNP>	-	-	-	-		
5	poonaan	<VF>	<B-VFP>	-	-	-	-	-	
6	.	<DOT>	<O>	-	-	-	-	-	

The output for the test data is as follows.

1	raaman	<NNP>	<B-NP>	-	-	5	<N.SUB>	-	-
2	kaNNanukku	<NNP>	<B-NP	-	-	4	<CL.IOBJ>	-	-
3	uNavu	<NPN>	<B-NP>	-	-	4	<CL.IOBJ>	-	-
4	koTuttuviTTu	<VNAV>	<B-VNP>	-	-	5	<X.CL>	-	-
5	poonaan	<VF>	<B-VFP>	-	-	0	<ROOT>	-	-
6	.	<DOT>	<O>	-	-	5	<SYM>	-	-

In the test input data, the Dependency Head and Dependency Relation are given as NULL.

5.3.2.3. Learning Algorithm

The version of MALT used for training and testing is MALT 1.4.1. MALT Parser uses the Shift Reduce Parser for parsing the data. The arc labels are identified using the LIBSVM classifier. The MALT parser has two algorithms for classifying the tags. One is LIBSVM and the other learning algorithm is LIBLINEAR. LIBSVM uses the Support Vector Machines and LIBLINEAR uses various linear Classifiers. Both the learning algorithm has its own advantages and disadvantages. The developed model for parsing uses the LIBSVM for classifying the parse data.

5.3.3. MST Parser Tool

MST Parser Tool is also another machine learning tool which uses supervised learning algorithm. Using this tool dependency labels and position of each word parent are obtained.

5.3.3.1 Training

A corpus is created for training in the format given below.

47. raaman oru paamp-aik kon-R-aan.
    Raman one snake-ACC kill-PAST-3MS
    'Raman killed a snake'

raamu	oru	paamp-aik	konRaan	.
NNP	DET	NN	VF	.
NSUB	X	DOBJ	ROOT	SYM
4	3	4	0	4

48. pacu pul tin-kinR-atu
    cow grass eat-PRES-3MS
    'The cow is eating grass'

pacu	pul	tinkiRatu	.
NN	NN	VF	.
N.SUB	D.OBJ	ROOT	SYM
3	3	0	3

49. avan oru kappu koNTuva-ndt-aan.
    He one stick bring-PAST-3MS
    'He brought a stick.'

avan	oru	kappu	koNTuvandtaan	
PRP	DET	NN	VF	.
N.SUB	DET	D.OBJ	ROOT	SYM
4	3	4	0	4

50. ciitai oru caari vaangk-in-aaL
    Sita one sari buy-PAST-3FS
    'Sita bought a sari'

ciitai	oru	caari	vaangkinaaL	
NN	DET	NN	VF	.
N.SUB	DET	D.OBJ	ROOT	SYM

4 3 4 0 4

In this way a corpus is created for Malayalam language. This corpus is trained using the MST Parser Tool and thus a model is created. Using this model as source the new inputs are tested.

5.3.3.2. Testing

POS-tagged Output is the input for MST Parser Tool. The above pos-tagged output is converted into the MST input format using a Perl program. This is then given to MST Parser Tool and by using the trained model, required output is obtained, i.e. position of each word parent and dependency labels of the Pos-tagged Sentence are obtained. An example is given below.

51. naan oru pazam paRikka marattil eeR-in-een.
I one fruit pluck-VINT tree-LOC climb-PAST-1S
'I climbed the tree to pluck a fruit'
POS-tagging Output:

naan <PRP>
oru <DET>
pazhaM <NN>
paRikka <VINT>
marattil <NN>
eeRineen <VF>
. <DOT>

MST input format:

naan oru pazam paRikka marattil eeRineen .
PRP DET NN VINT NN VF .
0 0 0 0 0 0 0
0 0 0 0 0 0 0

MST output:

naan oru pazam paRikka marattil eeRineen .
PRP DET NN VINT NN VF .
N.SUB DET CL.DOBJ XCL X ROOT SYM
6 3 4 6 6 0 6

5.4. Dependency Tree Viewer

For viewing the dependency structures as tree we use dot Software i.e. Graphiz Tool. To change the MST output and MALT output into the input format of the Grapphiz Tool (Diagraph Format Conversion), two different Perl programs are used. The following is an example for MST parser tool.

MST output:

1 2 3 4 5 6 7
naan oru pazam paRikka marattil eeRineen .
PRP DET NN VINT NN VF .
NSUB X CLDOBJ XCL X ROOT SYM
6 3 4 6 6 0 6

Digraph Format conversion:

Conversion 1

1 naan 6 <N.SUB>
2 oru 3 <X>
3 pazam 4 <CL.DOBJ>
4 paRikka 6 <X.CL>
5 marattil 6 <X>
6 eeRineen 0 <ROOT>
7 . 6 <SYM>

Conversion 2

<N.SUB>(6_ eeRineen,1_ naan)
<X>(3_ pazam,2_ oru)
<CL.DOBJ>(4_ paRikka,3_ pazam)

<X.CL>(6_ eeRineen,4_ paRikka)
<X>(6_ eeRineen,5_ marattil)
<ROOT>(0_ROOT,6_ eeRineen)
<SYM>(6_ eeRineen,7_.)

Conversion 3 (Input of Graphiz Tool)

digraph 1 {
"1_ naan~"[label=" naan~"];
"6_ eeRineen" -> "1_ naan"[label="<NSUB>"];
"2_ oru "[label=" oru "];
"3_ pazam " -> "2_ oru "[label="<X>"];
"3_ pazam "[label=" pazam "];
"4_ paRikka" -> "3_ pazam"[label="<CLDOBJ>"];
"4_ paRikka"[label=" paRikka"];
"6_ eeRineen " -> "4_ paRikka"[label="<XCL>"];
"5_ marattil"[label=" marattil"];
"6_ eeRineen " -> "5_ marattil"[label="<X>"];
"6_ eeRineen "[label=" eeRineen "];
"0_ROOT" -> "6_ eeRineen "[label="<ROOT>"];
"7_."[label="."];
"6_ eeRineen " -> "7_."[label="<SYM>"];
}

The above data is the input format for the Graphiz Tool. An output is obtained using Graphiz Tool (dependency tree of the sentence). The dependency tree thus formed is similar to that given under MALT parser tool. The following the example for MALT Parser Tool.

MALT output:

1	naan	_	<PRP>	<PRP>	_	6	N.SUB	_	_
2	oru	_	<DET>	<DET>	_	3	X	_	_
3	pazam	_	<NN>	<NN>	_	4	CL.DOBJ	_	_
4	paRikka	_	<VINT>	<VINT>	_	6	X.CL	_	_
5	marattil	_	<NN>	<NN>	_	6	X	_	_
6	eeRineen	_	<VF>	<VF>	_	0	ROOT	_	_
7	.	_	<DOT>	<DOT>	_	6	SYM	_	_

Digraph Format conversion:

Conversion 1

1	naan		6	N.SUB
2	oru		3	X
3	pazam		4	CL.DOBJ
4	paRikka	6	X.CL	
5	marattil		6	X
6	eeRineen		0	ROOT
7	.		6	SYM

Conversion 2

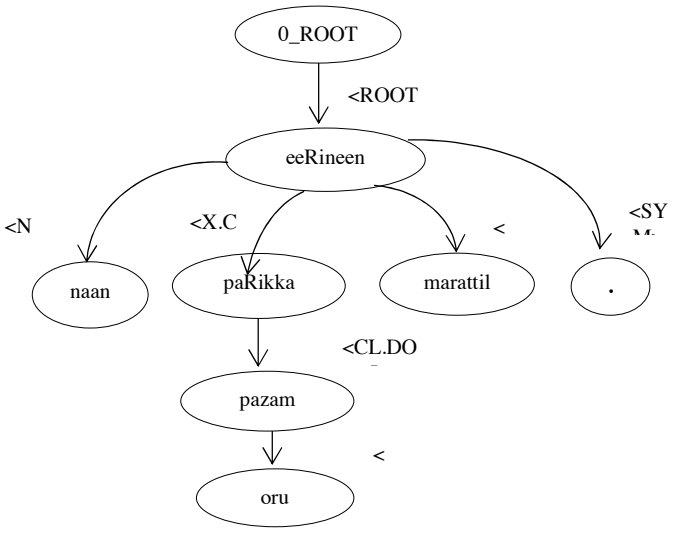
NSUB(6_ eeRineen,1_ naan)
DET(6_ eeRinee,2_ oru)
DOBJ(6_ eeRineen,3_ pazam)
VPCL(6_ eeRineen,4_ paRikka)
LOCMOD(6_ eeRineen,5_ marattil)
ROOT(0_ROOT,6_ eeRineen)
SYM(6_eeRineen,7_.)

Conversion 3

naan oru paham paRikka marattil eeRineen
digraph 1 {
"1_ naan"[label="1_ naan"];
"6_ eeRineen" -> "1_ naan"[label="NSUB"];
"2_ oru "[label="2_ oru "];

```
"6_ eeRineen" -> "2_ oru "[label="X"];
"3_ pazam"[label="3_ pazam "];
"6_ eeRineen " -> "3_ pazam "[label="CLDOBJ"];
"4_ paRikka "[label="4_ paRikka~"];
"6_ eeRineen " -> "4_paRikka "[label="XCL"];
"5_ marattil"[label="5_ marattil~"];
"6_ eeRineen " -> "5_ marattil~"[label="X"];
"6_ eeRineen "[label="6_ kayaRi "];
"0_ROOT" -> "6_ kayaRi "[label="ROOT"];
"7_."[label="7_."];
"6_ eeRineen " -> "7_."[label="SYM"];
}
```

The above data is the input format for the Graphic Tool. Output using Graphic Tool is given below.



The machine learning approach requires huge training data as the accuracy of the output depends on the training data. We need to have sumptuous training data at the POS tagging level, chunking level and dependency parsing level. MALT parser is a data driven system for dependency parsing that can also be used for syntactic parsing. MALT parser generally achieves good parsing accuracy. MALT parser can achieve robust, efficient and accurate parsing for wide range of languages. MST parser tool is a language independent tool used for Dependency Parsing which is implemented in English language. Using these tools the dependency labels and the position of head of Malayalam language is obtained. The results of both the tools are encouraging. Evaluation of the Dependency Parsing is done using both MST and MALT parser Tools. Both give commendably good result. Moreover for short range Dependencies, MALT Parser Tool holds good.

6. Question Answering System

Question Answering is a computer science discipline within the fields of information retrieval and natural language processing, which focuses on building systems that automatically answer questions posed by humans in a natural language. Valid answers mean answers relevant to the questions posed by the user. Stanford Question Answering Dataset (SQuAD) is a new reading comprehension dataset, consisting of questions posed by crowd workers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. In information retrieval, an open domain question answering system aims at returning an answer in response to the user's question. The returned answer is in the form of short texts rather than a list of relevant documents. The system uses a combination of

techniques from computational linguistics, information retrieval and knowledge representation for finding answers.

We are proposing a domain specific structured database from which the information related to Tourism can be retrieved. The questions will be parsed with the parser we have developed and the answers will be extracted from the structured database by matching. We are yet to build the question answer system.

7. Conclusion

The dependency parser based on Stanford parser is very much needed for any kind of NLP tasks in Tamil. The dependency information is very helpful for finding the subject, object and indirect object and other arguments (participants) in a sentence. The dependency information is very useful for building a question-answer system for Tamil. The parser built by us works efficiently for Tamil. By augmenting the training corpus we can bring efficiency to the parser. The parser is used for parsing questions and parsing the 50 thousand sentences kept in the structured data base. Correlating the question with the sentences in the structured data base and getting the correct answer is the challenging one. We are yet to work on this direction.

Reference

1. Lehmann, T. 199. A Grammar of Modern Tamil. Pondicherry: Pondicherry Institute of Linguistics and Culture.
2. Attardi, G. and Orletta, F.D. Chunking and Dependency Parsing. Technical paper.
3. Attardi, G., Chanev, A., Ciaramita, M., Dell'Orletta, F. and Simi, M. 2007. Multilingual Dependency Parsing and Domain Adaptation using DeSR. Proceedings of the CoNLL Shared Task Session of of EMNLP-CoNLL, Prague.
4. Buchholz, S., Marsi, E., Dubey, A. and Krymolowski, Y. 2006. Shared task on multilingual dependency parsing. In Proceedings of the Conference on Computational Natural Language Learning (CoNLL).
5. Chang, C. and Lin, C. 2011. LIBSVM: A Library for Support Vector Machines. Technical Report, National Taiwan University, Taipei, Taiwan.
6. CRF Website http://crfpp.sourceforge.net.
7. Dhanalakshmi V, Anand Kumar M., Rajendran S., and Soman K. P. 2009. POS Tagger and Chunker for Tamil Language. International Forum for Information Technology in Tamil.
8. Dhanalakshmi, V., Anand Kumar, M., Loganathan R, Soman, K.P. and Rajendran S. 2008.Tamil part-of-Speech-Tagger based on SVM Tool. In proceeding of the COLIPS International conference on Asian Language Processing (IACP). Chang mai, Thailand.
9. Dhanlakshmi,V. Shallow parser for Tamil. PhD thesis submitted to Tamil University, Thanjavur.
10. Ghosh, A., Das, A., Bhaskar, P. and Bandyopadhyay, S. 2010. Bengali Parsing system. In Proceedings of International Conference On Natural Language Processing (ICON), Tool Contest.
11. Gimenez J. and Marquez, L. 2006. SVMTool Technical Manual v1.3. Technical Manual. TALP Research Center, LSI Department, Universitat Politecnica de Catalunya, Barcelona.
12. Hsu, C. Chang C. and Lin, C. 2010. A Practical Guide to Support Vector Classification. Technical Report, National Taiwan University, Taipei, Taiwan.
13. Kesidi, S.R., Kosaraju, P., Vijay, M. and Husain, S. 2010. A two stage Constraint based Hybrid Dependency Parsing. Proceedings of International Conference On Natural Language Processing (ICON), Tool Contest.
14. Kolachina, S., Kolachina, P., Agarwal, M. Husain, A. 2010. Experiments with MaltParser for parsing Indian Languages. Proceedings of International Conference On Natural Language Processing (ICON), Tool Contest.
15. Kosaraju, P., Kesidi, S.R., Ainavolu, V.B.R. and Kukkadapu, P. 2010. Experiments on Indian Language Dependency Parsing. Proceedings of International Conference on Natural Language Processing (ICON) Tool Contest.
16. Kubler, S., McDonald, R. Nivre, J. 2009. Dependency Parsing. Morgan & Claypool publishers.
17. Lee, H. Park, S., Sang-Jo Lee, S. and Park, S. 2006. Korean Clause Boundary recognition. PRICAI'06 Proceedings of the 9th Pacific Rim international conference on Artificial intelligence. Springer-Verlag Berlin, Heidelberg.
18. Kumara Shanmugam, B. 2004. Parse representation of Tamil syntax. Thesis submitted for Master of Science (by research), Department Of Electronics Engineering, Faculty Of Science And Humanities, Anna University. Chennai.
19. MALT Parser website http://www.maltparser.org/userguide.html .
20. Maneffe, M. De and Manning, C.D. 2008, 2016. Stanfod typed dependency manual. Revised for the Stanford parser v.3.7.0 in September 2016.

21. McDonald, R. 2006. Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing. Ph.D thesis. University of Pennsylvania.
22. Nivre, J. 2005. Dependency Grammar and Dependency Parsing. Technical Report. School of Mathematics and systems Engineering, Vaxjo University.
23. Ohno, T., Matsubara, S., Kashioka, H., Kato, N. and Inagaki, Y. 2005. Incremental Dependency Parsing of Japanese Spoken Monologue Based on Clause Boundaries. Proceedings of 9th European Conference on Speech Communication and Technology (Interspeech-2005). Pages 3449-3452.
24. Rajendran, S. 2006. Parsing in Tamil: Present State of Art. Indian Linguistics vol. 67, pages 159-67.
25. Rekha R.U. 2010. Dependency Parsing for Tamil using Machine Learning Approaches. A Project Report. Amrita School of Engineering, Amrita University, Coimbatore.
26. Sundar Ram, R.V. and Devi, S.L. 2008. Clause Boundary Identification using Conditional Random Fields. Springer, Heidelberg, pages 140-150.
27. Titov, I. and Henderson, J. 2007. "A Latent Variable Model for Generative Dependency Parsing", Proceedings of International Conference on Parsing Technologies (IWPT -07), Prague.

Towards Building a Dependency Parser for Tamil: A Discussion on Tags

Keerthana B. And K. Parameswari
Centre for Applied Linguistics and Translation Studies,
University of Hyderabad
tulips91, parameshkrishnaa{@gmail.com}

Abstract. This paper discusses about the available annotation guidelines for building a parser in Indian languages and does a detailed comparative study between AnnCorra and Universal Dependencies tagset as these two are the prominent tagsets used for building a parser for Tamil in the recent past. A statistical study of tags used and the importance of language specific tags are highlighted.

1. Introduction

Annotation guidelines are backbone in developing treebanks for parsers. These guidelines are built based on available grammar formalisms and are framed at various levels- morphological tags, POS tags and syntactic relation tags. The widely used annotation guidelines for Indian languages in the recent past are AnnCorra (Bharati, A., Sangal, R., Sharma, D. M., & Bai, L., 2006) and Universal Dependencies guidelines (Nivre, J., De Marneffe, M. C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., ... & Tsarfaty, R. 2016), which are built based on dependency grammars. A statistical and comparative study is done between these tagsets in this paper.

2. A Survey on Grammar Formalisms

Grammar formalisms are essential in building the annotation guidelines as they define the linguistic properties. Some suitable grammar formalisms for building a parser includes:

2.1. Generalized Phrase Structure Grammar (GPSG)

It is a constraint-based grammar, deriving from constituency grammar, developed by Gerald Gazdar in the 1970s with Ewan Klein, Ivan Sag, and Geoffrey Pullum for English (Cf. Gazdar, G., et.al, 1985). Implemented languages include English, Persian, French, Chinese and Arabic (Bahrani, M. et.al., 2011). For example,

He gave him a book

((NP-he (N)))((VP-gave (V)))((NP- him (N)))((NP-a (DET) book (N)))

2.2. Head-driven Phrase Structure Grammar (HPSG)

It is lexical- based, constrained PSG, developed by Carl Pollard and Ivan Sag (Cf. Pollard, C., and Sag, I. A., 1994). Applicable languages include Romance languages, Slavic languages, German, Japanese, Welsh, English, Korean and Warlpiri (Levine, R. D. and Meurers, W. D., 2006). For example,

Felix chased the dog

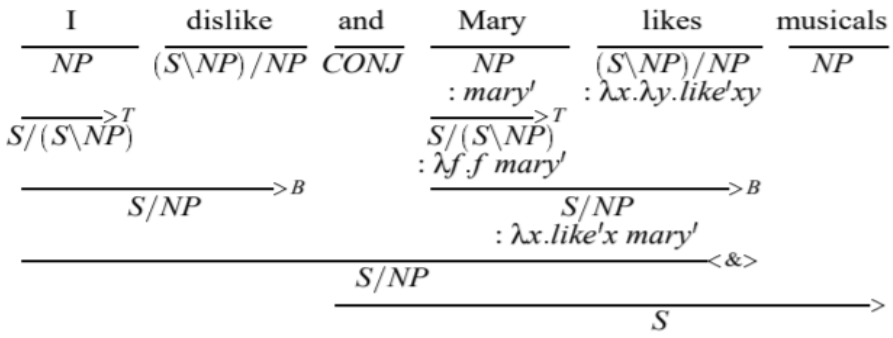
$$\begin{bmatrix} hd\text{-}spr\text{-}ph & \\ \text{PHON} & \langle \text{Felix,chased,the,dog} \rangle \\ \text{SYNSEM} & \text{'S'} \\ \text{NON-HD-DTRS} & \left\langle \begin{bmatrix} \text{PHON} & \langle \text{Felix} \rangle \\ \text{SYNSEM} & \text{'NP'} \end{bmatrix} \right\rangle \\ \text{HEAD-DTR} & \begin{bmatrix} hd\text{-}comp\text{-}ph & \\ \text{PHON} & \langle \text{chased,the,dog} \rangle \\ \text{SYNSEM} & \text{'VP'} \\ \text{HEAD-DTR} & \begin{bmatrix} word & \\ \text{PHON} & \langle \text{chased} \rangle \end{bmatrix} \\ \text{NON-HD-DTRS} & \left\langle \begin{bmatrix} \text{PHON} & \langle \text{the,dog} \rangle \\ \text{SYNSEM} & \text{'NP'} \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}$$

(Extracted from Sag, I. A., 1995:15)

2.3. Combinatory Categorial Grammar

A lexicalised grammar form where categorial grammar is extended with functional operators, developed by Mark Steedman and Remo Pareschi (1987) and Szabolcsi (1992). It is applied in English (Hockenmaier, J., and Steedman, M., 2002, 342). For example,

I dislike and Mary likes musicals



(Extracted from Mark Steedman, 1996 (4))

2.4. Lexical-Functional Grammar

LFG is first published by Joan Bresnan (1982), represented in constituent and functional structure. It is used in parsing Wall Street Journal (WSJ) by Stefan Riezler, et.al. (2002). The major advantage found in LFG is that the mismatch between the surface structure and the deep argument structure as discussed in Chomskyan framework is not found here. For example,

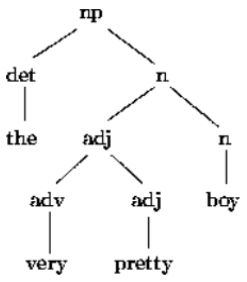
He gave the woman the gift

$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'he'} \\ \text{NUM} & \text{sg} \\ \text{PERS} & 1 \\ \text{DEF} & + \end{bmatrix} \\ \text{PRED} & \text{'give <agent,goal,patient>'} \\ & \text{SUBJ,OBJ2,OBJ1} \\ \text{TENSE} & \text{past} \\ \text{OBJ2} & \begin{bmatrix} \text{PRED} & \text{'the woman'} \\ \text{NUM} & \text{sg} \\ \text{PERS} & 3 \\ \text{DEF} & + \end{bmatrix} \\ \text{OBJ1} & \begin{bmatrix} \text{PRED} & \text{'the gift'} \\ \text{NUM} & \text{sg} \\ \text{PERS} & 3 \\ \text{DEF} & + \end{bmatrix} \end{bmatrix}$$

2.5. TreeAdjoining Grammar (TAG)

Tree Adjoining Grammar, formulated by Aravind Joshi (A. K. Joshi, Levy, and Takahashi, 1975) has both lexicalised and constraint-based variations. Elementary trees are combined here with substitution and adjunction operations (Kroch, A. S., & Joshi, A. K., 1985). It is applied in English and results obtained are better than previously mentioned formalisms (XTAG Research Group 1998; Abeille, A.; Bishop, K., Cote, Sharon, & Schabes, Y. 1990). For example,

The very pretty boy

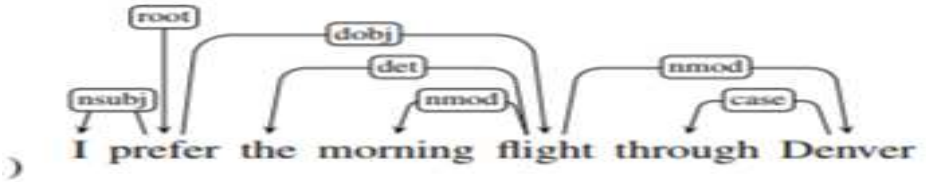


(extracted from Tree Adjoining Grammars, https://www.let.rug.nl/~vannoord/papers/diss/diss/node59.html)

2.6. Dependency Grammar (DG)

Dependency approach (both projective and non-projective) follows dependency grammar, tracing back to Panini's grammar. The modern thought of DG was proposed by a French linguist Lucien Tesnière during post-1950s. It represents the relation between the head and its dependents. Content words are marked by dependency relations; functional words attach to the content words they modify and punctuation attach to the head of the phrase/ clause. For example,

I prefer the morning flight through Denver.



(extracted from Speech and Language Processing (Jurafsky, D., & Martin, J. H., 2018))

For a morphologically rich and constituent-free language like Tamil, implementing dependency model is better (Falavarjani, S. A. M., & Ghassem-Sani, G., 2015). Faster manual annotation and more efficient parsing is applicable for any language in DG (Jurafsky, D., & Martin, J. H., 2018).

3. Survey on Available Tagsets

A detailed survey is done on the major available tagsets, including AnnCorra tagset, Universal Dependencies tagset, Stanford dependencies tagset, Penn tagset, Prague tagset and Chinese Dependency tagset (Appendix 1). Among these, Universal Dependencies and AnnCorra tagsets are found to be implemented for Tamil following dependency grammar.

3.1. AnnCorra Tagset

Annotated Corpora (AnnCorra), a Pāninian Dependency grammar based tagset is built on the basis of *kāraka* and non-*kāraka* relations. It major goal is to have a uniform representation of annotated corpus of Indian languages (Bharati, A., et.al, 2002). It is initially built for parsing Hindi sentences and thus, the tags presented are according to Hindi grammar. Later, the same guidelines were adapted for other Indian languages (Marathi, Urdu, Bengali, Kannada, Telugu, Tamil, and Malayalam) (Cf. Tandon, J. and Sharma, D. M., 2017). It was even used by Amita, A. J. (2015) for English, in which HyDT annotation scheme and hybrid approach (statistical+ rule based) were used for parsing 2000 words.

There are 19 *kāraka* relations1 and 25 Non-*kāraka* relations2 existing in the tagset. The unique relations include jk1, mk1, k1s, k2g, k2p, k2s, k4a, k7t, k7p, and k7a. The following table represents the relations with examples:

S. No.	Tag	Examples
1	pk1	*eṉṉai* ***avaṉ*** *vēlai ceyvittāṉ* 'He made me do the work'
2	jk1	***eṉṉai*** *avaṉ vēlai ceyvittāṉ* 'He made me do the work'
3	mk1	*eṉṉai avaṉ* ***ammāvaik koṇtu*** *vēlai ceyvittāṉ* 'He made mother to make me do the work'
4	k1s	***nāṉ*** *maruttuvar* 'I am doctor'
5	k2g	*nēruvai* ***māmā*** *eṉavum aẓaittaṉar* 'They also called Nehru as uncle'
6	k2p	*nāṉ* ***amērikkāviṙkuc*** *ceṉṟēṉ* 'I went to America'
7	k2s	*eṉṉai* ***putticāli*** *eṉak karutiṉar* 'They considered me as intelligent'
8	k4a	***eṉakku*** *kuḷirkiṉṟatu* 'I am feeling cold'

1 k1 (*karta* 'doer/agent/ subject'), pk1 (*prayojaka karta* 'causer'), jk1 (*prayojya karta* 'causee'), mk1 (*madhyastha karta*'mediator-causer'), k1s (*karta samanadhikarana*- 'noun complement of *karta*'), k2 (*karma* 'object/patient'), k2p (Goal, Destination), k2g (secondary *karma*), k2s (*karma samanadhikarana* 'object complement'), k3 (*karana* 'instrument'), k4 (*sampradana* 'recipient'), k4a (*anubhava karta* 'Experiencer'), k5 (*apadana* 'source'), k5prk (prakruti apadana 'source material'), k7t (*kAlAdhikarana* 'location in time'), k7p (*deshadhikarana* 'location in space'), k7 (*vishayadhikarana* 'location elsewhere'), k7a (according to) and k*u (s*Adrishya* 'similarity/comparison')

2 r6 (*shashthi* 'genitive/possessive'), r6-k1, r6-k2 (*karta* or *karma* of a conjunct verb (complex predicate)), r6v (*kA* 'relation between a noun and a verb), adv (*kriyAvisheSaNa* 'manner adverbs'), sent-adv (Sentential Adverbs), rd (direction), rh (*hetu* 'reason'), rt (*tadarthya* 'purpose'), ras-k* (*upapada sahakArakatwa* 'associative'), ras-neg (Negation in Associative), rs (noun elaboration), rsp (address terms), nmod__relc, jjmod__relc, rbmod__relc (relative clauses, *jo-vo* constructions), nmod (participles etc. modifying nouns), vmod (verb modifier), jjmod (D-Rel modifiers of the adjectives), pof (part of units such as conjunct verbs), ccof (co-ordination and sub-ordination), fragof (Fragment of), enm (enumerator), rsym (ag for a symbol) and psp_cl (relation between clause and postposition following that clause)

9	k7t	*iŋku **nēr̲r̲u** maẓai peytatu* 'It rained here yesterday'
10	k7p	*puttakam **paiyil** u[[atu* 'The book is in the bag'
11	k7a	*en̲ nāy **amērikkāvil** u[[atu* 'My dog is in America'

3.2. Universal Dependencies

Universal Dependency is a cross-linguistic project, built with the goal of facilitating multilingual parser development, cross-lingual learning, and parsing research from a language typology perspective (http://universaldependencies.org). The idea of annotation scheme has been taken from Stanford dependencies, Google universal part-of-speech tags and the Interset interlingua for morpho-syntactic tagsets in 2013 (McDonald et al., 2013). The parser has a lesser number of modules (Pre-processing (transliteration, sentence segmentation, tokenization); M-layer annotation (positional tagging) and A- layer annotation (dependency annotation), making it universal for any language typology. Indian languages like Hindi, Marathi, Sanskrit, Tamil, Telugu, and Urdu are included in the existing UD Treebank and Kannada and Pnar are upcoming languages listed in the UD website. There tagset is rich with 37 coarse grained tags3. Added to it, there are 198 language specific tags that are used in various languages parsing system.

The uniqueness of this tagset lies in the inter-clausal tags, including *acl, advcl, ccomp, xcomp,* and *csubj*. The following table describes the richness of this tagset with examples:

S. No.	Tag	Examples
1	acl	***itan̲āl** avarkaḷ kaitu ceyyap paṭakkūṭum enak karutappaṭṭatu* 'It was thought that she might get arrested because of this'
2	advcl	*nān̲ **iruntāl** uān̲kku en̲n̲a payam?* 'What is scary for you if I am there?'
3	ccomp	*un̲akkup paṭikkap **piṭikkum** en̲a avar con̲n̲ār* 'He said that you like to read'
4	xcomp	*avar **varaiya** ārampittār* 'He started to draw'
5	csubj	*avar **con̲n̲atu** arttamu[[a on̲r̲u* 'What he said is meaningful'

3 *acl* (clausal modifier of noun (adjectival clause)), *advcl* (adverbial clause modifier), *advmod* (adverbial modifier), *amod* (adjectival modifier), *appos* (appositional modifier), *aux* (auxiliary), *case* (case marking), *cc* (coordinating conjunction), *ccomp* (clausal complement), *clf* (classifier), *compound* (compound), *conj* (conjunct), *cop* (copula), *csubj* (clausal subject), *dep* (unspecified dependency), *det* (determiner), *discourse* (discourse element), *dislocated* (dislocated elements), *expl* (expletive), *fixed* (fixed multiword expression), *flat* (flat multiword expression), *goeswith* (goes with), *iobj* (indirect object), *list* (list), *mark* (marker), *nmod* (nominal modifier), *nsubj* (nominal subject), *nummod* (numeric modifier), *obj* (object), *obl* (oblique nominal), *orphan* (orphan), *parataxis* (parataxis), *punct* (punctuation), *reparandum* (overridden disfluency), *root* (root), *vocative* (vocative) and *xcomp* (open clausal complement).

4. Comparative Study between Anncorra and Universal Dependencies Tags

(i) AnnCorra tagset has a better representation of case which is essential for any morphologically rich language. A unique tag is given to each case and thus, a deep analysis is seen. For example, locative case has multiple functions and accordingly case is marked.

Tag: k7 (location elsewhere), k7t (location in time), k7p (location in space)

For instance:

(1) *iŋku* ***nēr̲r̲u*** *maz̲ai peytatu* 'It rained here yesterday' is marked k7t
(2) *puttakam* ***paiyil*** *u[[atu* 'The book is in the bag' is marked k7p
(3) *en̲* *nāy* ***amērikkāvil*** *u[[atu* 'My dog is in America' is marked location elsewhere

The same is not found in the UPOS tag of UD. Instead, the differentiation is done in language specific tags. The tag 'obl' (oblique nominal) is generally used for all the non-core arguments of a clause. The distinction is done in language specific tags according to the language's requirement. (1) is marked obl:tmod (temporal modifier), (2) is marked as obl:arg (argument), (3) is marked as obl:loc (location). Among these, 'obl:arg' is already introduced in UD. The rest of the tags are available in other languages and similar occurrences are found in Tamil as well.

Similar cases are seen in other case markers as well.

(ii) UD has an inter-chunk and intra-chunk representation unlike AnnCorra. AnnCorra has only intra-chunk tags. For instance,

Tag: acl in UD

(4) pa[[ikku ***celvataṟkāṉa*** *kāraṇam eṉṉa?* 'What is the reason for going to school?'

Here, in *celvataṟku+āṉa,* the former is marked 'acl' to the latter. This inter-clausal relation is absent in AnnCorra.

(iii) Dative subject constructions are marked in AnnCorra as k4a, whereas it is yet to be given a tag in UD for Tamil. The tag 'nsubj:nc' (non-canonical subjects) is marked in Telugu UD, which has to be extended for Tamil as well.

For instance,

(5) ***eṉakku*** *ku[irkiṉṟatu* 'I am feeling cold' is an experiencer and not the real subject of the sentence.

5. Conclusion

The UD tags seem to be shallow with respect to the AnnCorra tags. But they are accommodated as language specific relations. The problem of inter-chunk tags seems to be unresolved in AnnCorra, which is an added plus point to UD system. Thus, Universal Dependencies has a wider scope in parsing Tamil sentences.

References

1. Bharati, A., Sangal, R., Sharma, D. M., & Bai, L. 2006. AnnCorra: Annotating Corpora Guidelines for POS and Chunk Annotation for Indian Languages. *LTRC-TR31*, 1-38.
2. Gazdar, G., Klein, E., Pullum, G. K., and Sag, I. A. 1985. *Generalized Phrase Structure Grammar*. Cambridge: Harvard University Press.

3. Nivre, J., De Marneffe, M. C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., ... & Tsarfaty, R. 2016. Universal dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016),* pp. 1659-1666.
4. Steedman, Mark and Remo Pareschi 1987. A Lazy way to Chart-Parse with Categorial Grammars. *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, Stanford CA, 81-88.
5. Falavarjani, S. A. M., & Ghassem-Sani, G. 2015. Advantages of Dependency Parsing for Free Word Order Natural Languages. In *International Conference on Current Trends in Theory and Practice of Informatics*. **Berlin: Springer,** pp. 511-518.
6. Jurafsky, D., & Martin, J. H. 2018. *Speech and Language Processing*. London: Pearson. Vol.3, pp 270. https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf
7. Joshi,A.K., L. S. Levy, & M. Takahashi 1975. Tree adjunct grammars. *Journal Computer Systems Science*. *Vol*.10.
8. Bahrani, M., Sameti, H., and Manshadi, M. H. 2011. *A Computational Grammar for Persian Based on GPSG*. Retrieved on 21st November, 2017. https://link.springer.com/article/10.1007/s10579-011-9144-1
9. Pollard, C., and Sag, I. A. 1994. *Head-Driven Phrase Structure Grammar*. US: University of Chicago Press.
10. Levine, R. D., and Meurers, W. D. 2006. Head-Driven Phrase Structure Grammar. In *Encyclopedia of Language and Linguistics*. Vol. 5, pp. 237-52.
11. Szabolcsi, Anna 1992. On Combinatory Grammar and Projection from the Lexicon. In Ivan Sag & Anna Szabolcsi (ed.), Lexical Matters. *CSLI, pp*. 241-268.
12. Hockenmaier, J., & Steedman, M. 2002. Generative Models for Statistical Parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics,* pp. 335-342.
13. Bresnan, J. 1982. Control and complementation. In *Linguistic inquiry*. Vol. 13. No. 3, pp. 343-434.
14. Riezler, S., King, T. H., Kaplan, R. M., Crouch, R., Maxwell III, J. T., and Johnson, M. 2002. Parsing the Wall Street Journal Using a Lexical-Functional Grammar and Discriminative Estimation Techniques. In *Proceedings of the 40th ACL,* pp. 271-278.
15. Bharati, A., Sangal, R., Chaitanya, V., Kulkarni, A., Sharma, D. M., and Ramakrishnamacharyulu, K. V. 2002. AnnCorra: Building Tree-banks in Indian Languages. In *Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization: Association for Computational Linguistics*. Vol. 12, pp. 1-8.
16. *Universal Dependency (UD) Tagset*. Retrieved on 1st June, 2018. https://universaldependencies.org
17. Taylor, A., Marcus, M., and Santorini, B. 2003. The Penn Treebank: An Overview. In *Treebanks*. Dordrecht: Springer, pp. 5-22.
18. Kroch, A. S., & Joshi, A. K. 1985. The Linguistic Relevance of Tree Adjoining Grammar. *Technical Reports (CIS)*, pp.671.
19. XTAG Research Group 1998. A Lexicalized Tree Adjoining Grammar for English. https://arxiv.org/abs/cs/9809024
20. Abeille, A., K.Bishop, Sharon Cote, and Y. Schabes 1990. *A Lexicalized Tree Adjoining Grammar for English*. Technical Report MS-CIS-90-24, Department of Computer and Information Science, University of Pennsylvania.
21. Tandon, J., and Sharma, D. M. 2017. Unity in Diversity: A Unified Parsing Strategy for Major Indian Languages. In *Proceedings of the Fourth International Conference on Dependency Linguistics,* pp. 255-265.
22. Amita, A. J. 2015. An Annotation Scheme for English Language Using Paninian Framework. *IJISET*, Vol. 2, pp. 616-619.
23. McDonald, R., Nivre, J., Quirmbach-Brundage, Y., Goldberg, Y., Das, D., Ganchev, K., ... and Bedini, C. 2013. Universal Dependency Annotation for Multilingual Parsing. In *Proceedings of the 51st ACL*. Vol. 2, pp. 92-97.

Enhancing Tamil Morphological Analyser for Manipravalam

Pravinvignesh S K, Krishnaraj N and Maruthamani M.
Anna University, Chennai

Abstract. Many traditional texts and their commentaries are written in Manipravalam language, which is a combination of Tamil and Sanskrit. In order to make such Manipravalam texts computationally accessible, the Manipravalam texts need to be processed. One of the first steps in the processing is morphological analysis. The Manipravalam texts use many Sanskrit words, and some uncommon Tamil words, but the grammar rules and forms are similar or closer to that of Tamil. Hence, an attempt is made in this paper to extend an existing Tamil morphological analyser to handle the Manipravalam words. Any enhancement to an analyser, requires addition of new rules to handle different grammatical morphemes, and addition of words to a dictionary to handle the lexical morphemes. The Tamil morph analyser is enhanced with additional rules to tackle the morphemes unique to Manipravalam. To handle the dictionary construction, a two step semi-automatic approach is used. Given a Manipravalam text, we first identify native (Tamil) and non-native words (Sanskrit) using a classifier. The native words are then given as input to the enhanced Tamil analyser. The words that are not handled by the analyser are listed, and an interface is built to add the words with their categories to the dictionary. The modified analyser has been tested with 4000 words and the accuracy is seen to increase from 43% to 76%.

1 Introduction

Many old documents and commentaries on classic texts have been written in Manipravalam language, which is a combination of Tamil and Sanskrit. Understanding these texts requires knowledge of both Sanskrit and Tamil. Even a native speaker, well-versed in Tamil, would find it difficult to interpret these texts. However, the knowledge embedded in the Manipravalam texts is valuable and has to be uncovered for access by posterity. Manual translations of some Manipravalam to Tamil texts have been carried out by interested individuals[9], but it is a tedious task. Hence, an effort towards machine translation of Manipravalam to Tamil is being envisaged. One of the first steps in this direction is the development of natural language processing (NLP) tools for Manipravalam. Among the various tools, this paper proposes the construction of a Morphological Analyser (MA) for Manipravalam.

The grammar rules in Manipravalam, are predominantly similar to those of Tamil. However, many Sanskrit words are used as lexical morphemes, and grammatical morphemes from Tamil are added to them. Also, some grammatical morphemes unique to Manipravalam are used. Thus, considering all these factors, this paper proposes to use an existing Tamil MA, but with significant enhancements to handle the challenges of Manipravalam.

The approach taken is as follows: Firstly, the Manipavalam text is directly given as input to an existing Tamil MA[1]. The words that are not analysed by the Tamil MA are studied. It is found that some of these words are Sanskrit words used as such (but written in Manipravalam (Grantham sript), Such words are classified as non-native words, and cannot be handled by the Tamil MA as the grammatical morpheme forms are different. Hence, such words are not considered for MA. The other words not handled by the Tamil MA are analysed to check if it is a case of a missing lexical morpheme or grammatical morpheme. Based on this study, we have devised a mechanism to add the missing lexical morphemes in a human-guided manner, and added new rules in the Tamil MA to handle the missing grammatical morphemes.

Thus the enhancement of the Tamil MA consists of the following tasks:

1. Classify the words in the text into native and non-native categories.
2. Add new rules for the suffixes of Manipravalam language.
3. Develop a dictionary creator to add missing root words of Manipravalam language.

The improvement in the accuracy of the analysis is observed. This can be treated as an iterative process, and additional words and rules can be added to further improve the accuracy.

The rest of the paper is organized as follows. Section 2 presents the related work, which is essentially about techniques used for MA for Tamil. To our knowledge, this is the first work on NLP for Manipravalam.

Section 3 presents the proposed methodology that modifies the existing Tamil MA to handle Manipravalama. Section 4 presents the evaluation of the morphological analyser, and Section 5 presents the conclusion.

2.Related Work

This section discusses existing work on morphological analysis and text classification.

2.1 Morphological Analysis

Morphological analyzer is a tool which takes words as input and produces its grammatical structure in terms of root words, affixes, parts of speech etc., as output. It identifies and segments the words and assigns the grammatical information. Since this is a tedious and challenging job, many researchers have developed rules and rule based syntax for performing morphological analysis. Tamil morphological analyzers were originally built by Aanandhan et al. [1] in 2002 and then by AU-KBC Research Centre in 2003 [2].

Since then research on Tamil morphological analysis has continued in two directions, using machine learning and using rule based approaches. Akilan and Naganathan [3] have developed a morphological analyzer for classical Tamil texts using rule based approach. The rules developed by them are dependent on each other and their MA produces 72 % of accuracy for classical Tamil texts.

Selvam and Natarajan [4] carried out research on morphological analysis and POS tagging for Tamil using a rule based approach via projection and induction techniques. Another morphological analyzer for Tamil was implemented using the sequence labeling based machine learning approach by Kumar et al. [5]. It was a supervised machine learning approach and a corpus with morphological information was used for training.

2.2. Text Classification using SVM

Text classification is an automated process of classification of text into predefined categories. This can be done with the help of Natural Language Processing and different Classification Algorithms like Naive Bayes, SVM, and Neural Networks.

Joachims [6] explains the use of Support Vector Machines (SVMs) for learning text classifiers. It analyzes the particular properties of learning with text data and identifies why SVMs are appropriate for this task. This paper explores and identifies the benefits of Support Vector Machines (SVMs) for text categorization. Moreover, in contrast to conventional text classification methods SVMs prove to be very robust, eliminating the need for expensive parameter tuning. This has been proven by considering theoretical and experimental results. The experiments compare the performance of SVMs using polynomial and RBF kernels with four conventional learning methods commonly used for text categorization. The test collection is taken from the Ohsumed corpus compiled by Hersh. From the 50216 documents in 1991 which have abstracts, the first 10000 are used for training and the second 10000 are used for testing. The classification task considered here is to assign the documents to one or multiple categories of the 23 MeSH diseases categories. The results show that SVM works fine and better than the conventional learning methods. Empirical results support the theoretical findings. SVMs achieve substantial improvements over the currently best performing methods and behave robustly over a variety of different learning tasks.

Mohamed and Shanmugasundaram [7] explain an approach to cluster words in a document containing Tamil words. This uses vector space model to cluster the documents. Vector space model is otherwise known as “Term-frequency approach”. Stop words which are frequent, meaningless terms are removed from the input text document to decrease the size of the document to be processed. Then the cosine similarity measure is applied to find the similarity between the input text documents. Then clustering is done using K-Medoid algorithm and optimal number of medoids and corresponding clusters are found.

Rani and Satvika [8] proposed text categorization on multiple languages based on classification technique. The objective of the work is the representation and categorization of Indian language text documents using text mining techniques, such as Support Vector Machine, KNN (KNearest Neighbor), and Decision Tree.

Based on the survey of the related work, we use the existing Tamil MA, which is rule-based, and SVM for text classification. The proposed methodology is presented in the next section.

3. PROPOSED METHODOLOGY

The tasks involved in adapting the existing MA to handle Manipravalam text are shown in Figure 3.1. The tasks involved are:

1. Native/Non-native text classification,
2. Addition of rules in Morphological Analyser, and
3. Semi-automatic Dictionary Creator.

Manipravalam document is given as input to the classifier. It separates the words into two classes, native(Tamil and Manipravalam words) and non-native(Sanskrit or Sanskrit sounding words). The Tamil words are given as input to the morphological analyser. The analyser splits those words whose root words are found in the dictionary. For the missing root words, the dictionary creator is used to add those to the dictionary.

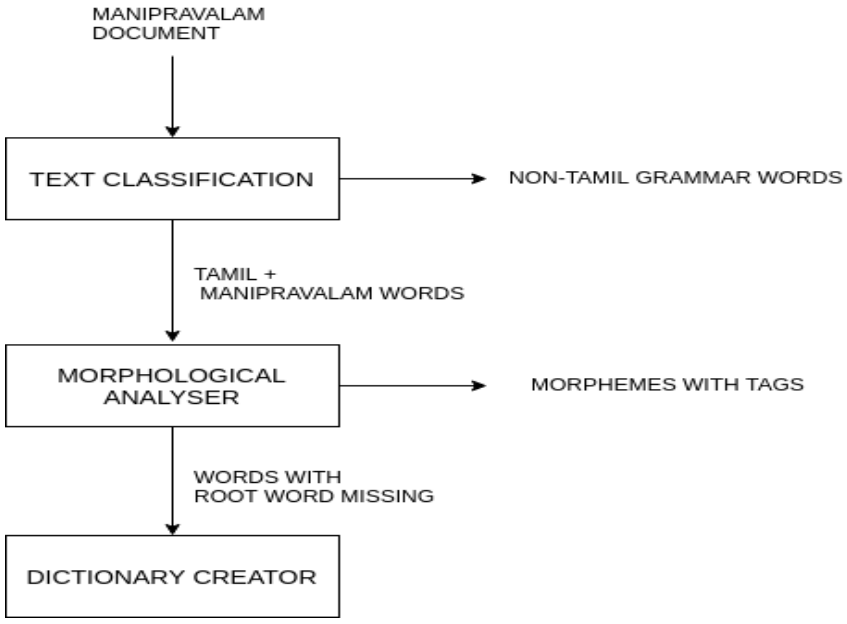


Figure 3.1 Proposed Methodology

3.1 Text Classifier

An SVM classifier is used to classify the text into native and non-native words. The text classifier consists of two phases namely training and testing. The training dataset used for classification has two classes labeled as 0 and 1. Label 1 indicates native words which can be split and processed by the analyser which includes Tamil words and Manipravalam words. Label 0 indicates words which cannot be processed by the MA, which consists mainly of Sanskrit words. The feature used for classification is character frequency weighted with inverse term frequency (CF-ITF), which is akin to tf-idf. The vector of characters (which is indicative of the pronunciation or sound of each word is used as the distinguishing feature to differentiate native and non-native words. The words in the training dataset are vectorized using CF-ITF vectorization. CF-ITF features are numerical values. The vectorized values and labels are trained using SVM linear kernel to develop a SVM predictive model. In the testing phase, Manipravalam text document is given as input and preprocessed to remove spaces and commas. The preprocessed data (manipravalam words) are fed into the SVM predictive model to classify the words as native and non-native. Figure 3.2 depicts the flow of the classifier.

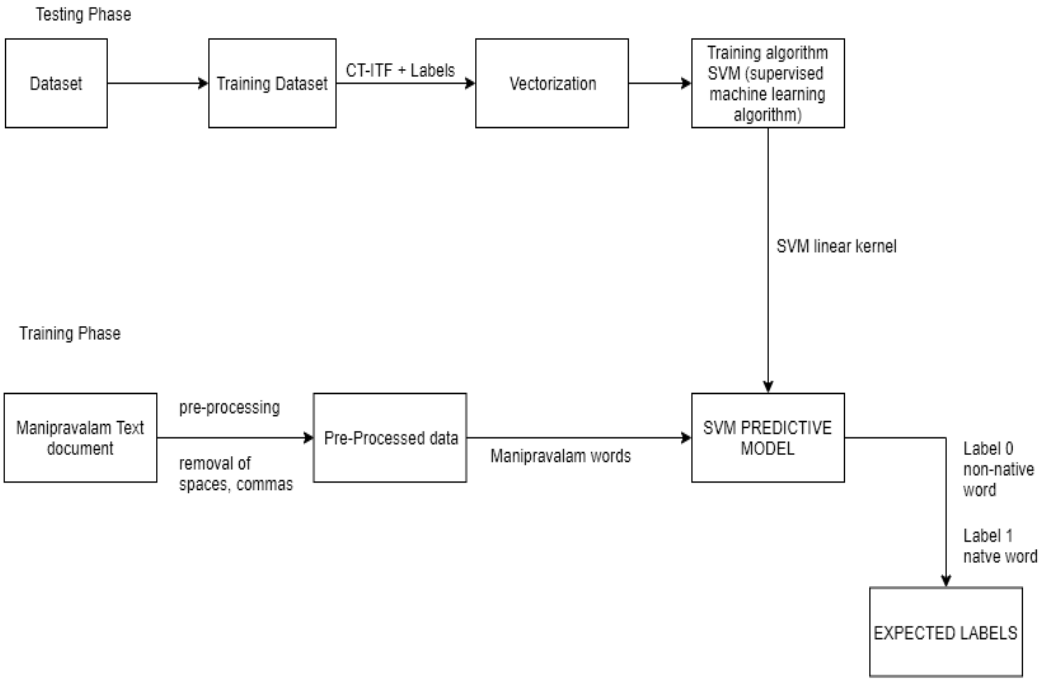


Figure 3.2 Classifier

3.2 Modification of Tamil MA

The Tamil MA considered takes a derived word as input and separates its root word and associated morphemes. The rule based approach is used and the function of each suffix is indicated.

Example for Morphological analysis:
1.Veetilirundhu
Veedu(noun) + il(Locative case) + irundhu(Postposition).

Tamil nouns can take case suffixes after the plural marker. They can also have post positions after that. Tamil words consist of a lexical root to which one or more affixes are attached. Most Tamil affixes are suffixes. Tamil suffixes can be derivational suffixes, which either changes the part of speech of the word or its meaning, or inflectional suffixes, which mark categories such as person, number, mood, tense, etc. The words can be analyzed like the one above by identifying the constituent morphemes and their features can be identified as Verb, Noun, Adjective, Adverb, Particle, Negative finite Verb, Conjunction, Interjection, Interrogative, Adjective, Finite Verb, Postposition, and Intensifier. Figure 3.3 gives a partial list of suffixes and their functions.

PostPosition	அற்க், படும், ஆல்தான், வைத்து, உடைய, கொண்டு, எதிரே, பதிலாக, படி, பற்றிய, மூலமாக ,அன்படி,இன்படி,ஆததால், அதால், மாறு
Clitics	என்னும், ஆகிலும், ஆயினும், ஆவது, அம்மா, அப்பா, அய்யா
Suffix	அலாம், கலாம், இடம், உக்கு, உக்க், இற்கு, இற்காக, அற்காக, அற்கு, அதற்கு, அக்கு , க்கு இலிருந்து,இடமிருந்து,இலிருந்த்,இடமிருந்த், உடைய, அற்று,
Tense	கிற், கிறு, க்கிறு, கின்ற், கின்று, க்கின்று, க்கிற், க்கின்ற்
Sandhi	ஆண், தின், உண், பூண், த்து, ப், வ், ந், இன், அன், ற்று,ஓடு, ழ், ஓடு, ஆள், ஆன், ஆர், அடா, அடி, ஏ, ஆம்,கள், தான்,

Figure 3.3 Partial List of Suffixes

Figure 3.4 depicts the working of morphological analyser. The word is checked for presence in the dictionary. If it is not present, the affix split takes place according to the morphology rules, and the morphemes identified. Again a dictionary check is performed for the root word, and the process is repeated until it is found in the dictionary, and the appropriate tag is assigned.

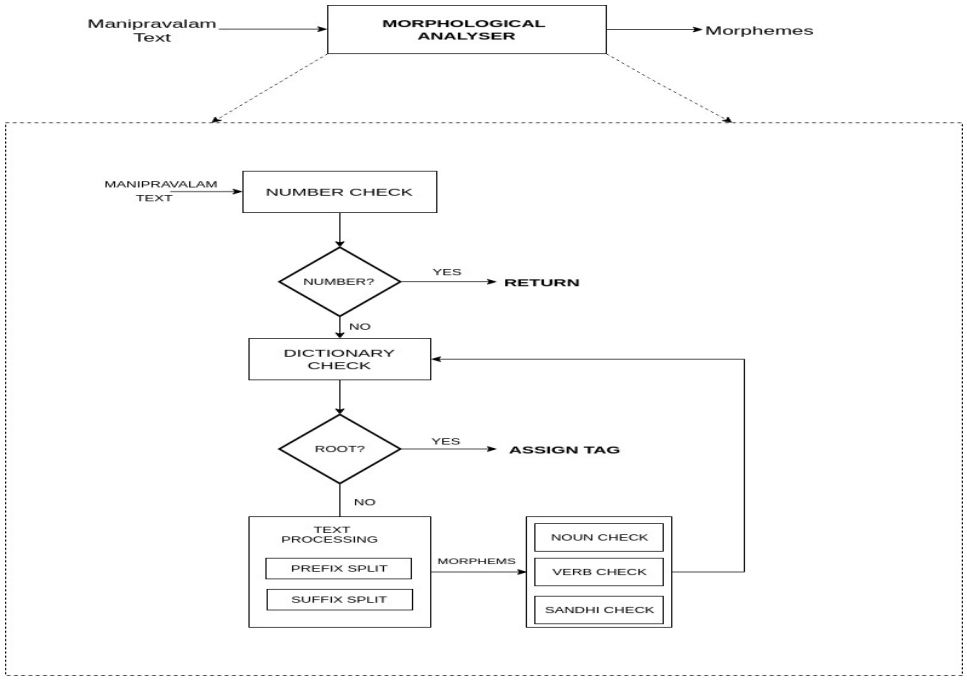


Figure 3.4 Working of Morphological Anayser for Manipravalam

Rules in the morphological analyser denote the possible suffixes of the language. Existing morphological analyser for Tamil has all the suffix rules related to Tamil language which makes the analyser to split those suffixes from the root words. Manipravalam language has some rules which are not part of Tamil. Those rules are found by testing the analyser with inputs of Manipravalam text. Rules are added to the analyser into the possible suffix category based on the output of testing. Figure 3.5 has the list of newly added rules and their categories with examples.

RULE SUFFIX	CATEGORY	MANIPRAVALAM WORD	Words covered in each rule
Irukkai	locative case	sonnadhaayirukkai	50
pOlE	postposition	ennumAppOle	250
AlE	adjective	kidakkaiyAlE	180
vENdu	Verbal participle suffix	aatravendittru	30
enkiRa	adverb	vaasagamenkira	25
vENum	verb	pirakkavenumo	70
enRum	adverb	barathathvamenRum	40
AyiRRu	Finite verb suffix	amruthamaayiRRu	10
Ittu	Auxiliary verb	ivattraiyittu	5
irukkum	Finite verb suffix	thoorasthanaayirukkum	60
thoRRu	Past tense marker	aagaranthoRRa	40

Figure 3.5 Rules and their Categories

3.3 Dictionary Creator

Dictionary creator gets as input, the words which are not analysed by the analyser. It has an interface to edit the words before it is put into the dictionary and buttons based on the dictionary files like verb, noun, adverb, and adjective. Using these buttons and the interface the unanalysed words are loaded into the appropriate dictionary. Figure 3.6 shows the user interface for the dictionary creator.

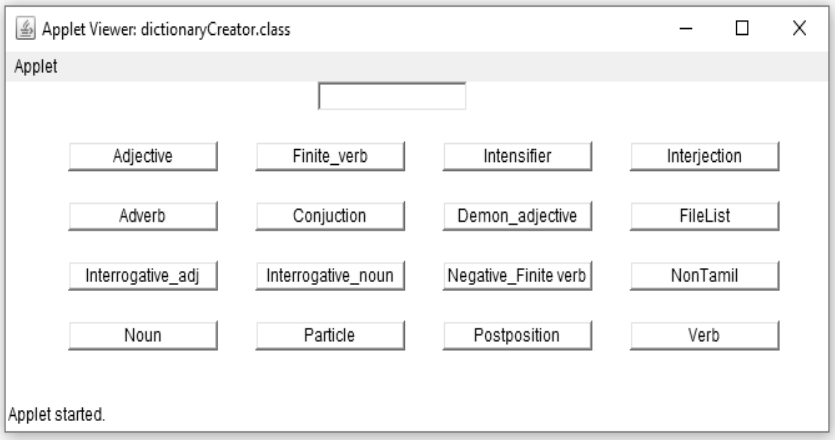


Figure 3.6 Dictionary Creator

4. Performance Evaluation

4.1 Text Classification

The SVC linear kernel module and scikit-learn python library are used to develop the classifier. The classifier is tested with a corpus of 3000 words. Ten-fold cross-validation is used and an accuracy of 0.88 is obtained for the classifier. Sample test results are shown in Figure 4.1.

s.no	words	label
1	இப்படி	1
2	பரஸ்வரூபம்	0
3	நிர்ணீதமாயிற்று	1
4	பரஸ்வரூப	0
5	ப்ரகிஸம்பந்தியான	0
6	ஸ்வஸ்வரூபமிருக்கும்படி	0
7	என்னென்னில்	1
8	உன்மிசை	1
9	உயிரெனக்	1
10	கரந்தெங்கும்	1
11	பரந்து	1
12	என்று	1
13	சரீராத்ம	1
14	பாவத்தைக்	1

Figure 4.1: Sample Test Results

4.2 Analyser

The modified MA has been tested with about 4000 words. When a Manipravalam text is given as an input to the existing analyzer it only separates the suffixes of the words for which the root words are found in the dictionary. Hence, some root words are added to the dictionary by analyzing the output to make it work for Manipravalam text. The semi-automatic tool is used to add the root words to the dictionary. Figure 4.2 gives the results before and after adding root words to the dictionary.

Manipravalam words	Root words	Before adding the root words	After adding the root words
1.உபயவிபூதியும்	உபயவிபூதி	த் + உ + இ + ய் + உம	உபயவிபூதி + ய் + உம்
2.கிஞ்சித்கரிக்க	கிஞ்சித்கரு	Analyser Not Found	கிஞ்சித்கரு + இ + க்க் + அ
3.ஸம்ச்லேஷத்தை	ஸ்அம்ச்லேஷம்	அத்து + ஐ	ஸ்அம்ச்லேஷம் + அத்து + ஐ
4.அநுஸந்தித்து	அநுஸ்அந்து	Analyser Not Found	அநுஸ்அந்து + இ + த்த் + உ
6.நிர்வஹிக்கும்படி	நிர்வஹ்	Analyser Not Found	நிர்வஹ் + இ + க்க் + உம் + படி.
7.நிர்வ்ருதியென்று	நிர்வ்ருது	Analyser Not Found	நிர்வ்ருது + ய் + என்று
8.ஸுகமாய்	ஸ்உகம்	ய் + என்று	ஸ்உகம் + ஆய்
9.ஸுகிக்கிறார்	ஸ்உகி	ஆய்	ஸ்உகி + க்கிறு + ஆர்
10.அவிச்சேதத்தைப்	அவிச்சேதம்	Analyser Not Found	அவிச்சேதம் + அத்து + ஐ + ப்

Figure 4.2: MA output Before and After Adding Root Words

Figure 4.3 shows a sample of the output of the MA before and after adding the rules mentioned in Figure 3.5. The number of words that have benefitted from the rule are given in Figure 3.5.

Suffix	Word	Before adding the rule	After adding the rule
இருக்கை	சமைந்திருக்கை	Analyser Not Found	சமை+ந்த்+உ+இருக்கை
போலே	என்னுமாபோலே	Analyser Not Found	என்னும்+ஆ+போலே
ஆலே	வருகையாலே	Analyser Not Found	வருகை+ய்+ஆலே
வேண்டு	ஆற்றவேண்டிற்று	ட் + உ + இற்று	ஆற்ற+வேண்டு+ உ +இற்று
என்கிற	வாசகமென்கிற	ன் + கிறு + அ	வாசகம்+என்கிற
வேணும்	பிறக்கவேணுமோ	உம் + ஓ	பிறக்க+வேணும் +ஓ
என்றும்	பரதத்வமென்றும்	என்று + உம்	பரதத்வம் +என்றும்
ஆயிற்று	அம்ருதமாயிற்று	Analyser Not Found	அம்ருதம்+ஆயிற்று
இட்டு	இவற்றையிட்டு	Analyser Not Found	இவற்றை+ய்+இட்டு
இருக்கும்	விஷயமாயிருக்கும்	Analyser Not Found	விஷயம்+ஆய்+இருக்கும்
தோற்று	ஆகாரந்தோற்ற	ற் + உ + அ	ஆகார் +தோற்று +அ

Figure 4.3 MA Results Before and After Adding Rules

Figure 4.4 shows the accuracy of analyser before and after the modifications. It can be seen that the accuracy has gone up from 43% to 76% with the use of the three tasks used to enhance the existing MA.

STATE	ACCURACY
EXISTING MORPHOLOGICAL ANALYSER	43%
AFTER CLASSIFIER	51%
AFTER ADDING ROOT WORDS	64%
AFTER ADDING RULES	76%

Figure 4.4 Analyser Accuracy

5. Conclusion

This paper has presented a methodology to use existing Tamil morphological analyser to support Manipravalam texts. The methodology consists of a classification scheme to identify non-native words, addition of morphological rules, and dictionary enhancement. The accuracy of the modified morphological analyser has improved from 43% to 76% for the limited corpus of 4000 Manipravalam words that have been considered. This is work-in-progress, and additional rules have to be added to handle some Manipravalam-specific morphemes. Further, the dictionary needs to be enhanced. The methodology proposed eases both these tasks, and with a few iterations and more sample data, a robust MA for Manipravalam can be developed.

References

1. P. Aanandhan, Ranjani Parthasarathi, T.V. Geetha, and K. Saravanan. "Morphological analyzer for Tamil". In the proceedings of Natural language Processing, Vol. 2, pp. 257-267, 2002.
2. R. Akilan and Naganathan. "Morphological Analyzer for classical Tamil Texts". In the proceedings of Natural Language Processing, Vol. 3, pp. 437-447, 2014.
3. Selvam and Natarajan. "Improvement of rule based morphological analysis and POS tagging in Tamil language via projection and induction techniques." In the journal of Computers, Vol. 3, pp. 357-367:357–367, 2009.
4. M. Kumar, V. Dhanalakshmi, K.P. Soman, and Rajendran Sankaravelayuthan."A Sequence Labeling Approach to Morphological Analyzer for Tamil Language". In the journal of Computer Science and Engineering , Vol. 2, pp. 345-349, 2010.
5. Thorsten Joachims. "Text categorization with Support Vector Machines: Learning with many relevant features". In the journal of Computer Science and Engineering, Vol. 4, pp. 137-142, 2009.
6. Syed Sabir Mohamed and Hariharan Shanmugasundaram. "Experiments on document clustering in Tamil language." In the journal of Engineering and Applied Sciences , Vol. 13, pp. 125-134, 2018.
7. Kapila Rani and Satvika. "Text Categorization on Multiple Languages Based On Classification Technique." In the journal of Computer Science and Information Technologies , Vol. 7, pp. 167-177:167–177, 2016.
8. Tamil morphological analyser. Au-kbc.org , 2003.
9. http://www.namperumal.com/divya-prabhandam.html

Morphological Stemmer and Lemmatizer for Tamil

Rethanya. V[1] , Dr.V.Dhanalakshmi[2], Dr.MAnandkumar[3], Dr.S.Rajendran[4]

[1]Crescent Institute of Science & Technology, Chennai.
[2]Dept. of Tamil, Govt. Arts College for Women, Krishnagiri
[3]Department of Information Technology, NITK, Suratkal
[4]CEN, Amrita Vishwa Vidyapeetham, Coimbatore.
rajushush@gmail.com

Abstract. Stemming and Lemmatization are important steps in many of the Information Retrieval (IR) and Natural Language Processing (NLP) tasks. Stemming is usually done to remove the attached suffixes and prefixes from inflected words or the word forms . Lemmatization is the process of converting the stem word to its root word or the dictionary word. This paper aims to identify and implement novel stemmer and lemmatizer methodology for Tamil, which is much complex due to its rich agglutinative nature

1. Introduction

Stemming is usually done before the actual assignment of the word to the index, which is done to reduce the size of the index. Stemming system is referred to as Stemmer. Stemmer is used in search and retrieval systems. Stemming systems have been developed since 1960's. Lovins stemmer has been developed in 1968. There are many stemmers available for English and other European languages. The most commonly used Stemmer for English is Porter Stemmer.

Lemmatization is the process of converting the stem word to its Lemma or root word or the dictionary word. Unlike stemming, lemmatization depends on identifying the part of speech and meaning of the intended word in a sentence. Lemmatization is more complex process than stemming. Tamil is morphologically rich and agglutinative language (Thomas Lehmann, 1992). Tamil words are postpositionally inflected with various grammatical features. Tamil verb specifies almost everything like gender, number, and person markings and also with auxiliaries which represents mood and aspect. Tamil noun inflects for plural, case suffixes and post positions. To solve this complexity, a classification of Tamil verbs based on tense markers and inflections is made. The verbs have been classified into thirty-two paradigms, based on their tense markers and morphophonemic change. Nouns are classified into twenty-five paradigms (S.Rajendran, et.al, 2001). The same paradigm classification has been used here to develop the stemmer and lemmatizer.

In this paper we describe a fast and simple Stemmer and Lemmatizer for Tamil using an efficient algorithm. This novel approach can be applied to any morphologically rich language. As the user passes the wordforms , the developed algorithm stems and produce the Lemma using two different modules. The first module (Stemmer) takes the wordform as input and gives the lemma's paradigm number and word's stem as output by passing to the Suffix table database. The second module (Lemmatizer) exract root suffix from the paradigm number information and add to the stem to get the Lemma or the root word.

2. Creation of Suffix Table

The Suffix table is the most essential file in this algorithm. This is a simple two-dimensional (2D) table where row corresponds to the morpho-lexical form and column corresponds to the paradigm number. Each syntactic category has its own suffix table. Here we have only created for noun and verbs. The noun suffix table contains 325 rows (word-forms including postpositions) and 25 columns (paradigms) similarly verb suffix table contains 628 rows (wordforms including auxillaries) and 32 paradigms.

Number of paradigms for each word class (noun/verb) is defined. In Tamil there are 32 paradigms for verb and 25 for noun (S.Rajendran, et.al, 2001). Table -1 show the number of paradigms and inflections of verb and noun.

Table 1. Paradigms and Inflections

	No. of Paradigms	**No. of Inflectional word forms**	**Total no. of word forms**
Verb	32	1884 (Including Auxiliary verbs)	60,288
Noun	25	325 (Including Postpositions)	8,125

For every paradigm a word is selected and this is termed as head word. For this head word, all morpho-lexical forms are created for noun and verb individually. In Tamil there are more than thousand word-forms are possible for each verb. Here we have selected 628 most frequently used wordforms for verb including 25 auxiliary verbs and for noun it is 325 including postpositions. The similar verb/noun morpho-lexical information pattern should be followed for all the paradigms. A morpho-lexical Information list is also created for the above morpho-lexical forms. Using all the word-forms a table is created, each column of the table corresponds to its paradigm. In that table, stem of the each paradigm is removed from its word-form. Now this table is represented as a Suffix table. Table.2 illustrates the sample suffix-table for Tamil verbs.

Table.2. Suffix Table

	P-1	P-2	P-3	P-4	P-5	
MLI-1	ththAn	wAn	inAn	thAn	Ran	.....
MLI-2	ththAL	wAL	inAL	thAL	RAL	.....
MLI-3	ththAr	wAr	inAr	thAr	RAr	.....
MLI-4	kinRAn	kinRAn	kinRAn	kinRAn	kinRAn	.....
MLI-5						.....

3. System Implementation

The system needs to extract the stem and then the Lemma or the root word. By the way the system is implemented makes it distinct from other systems. The input which is in Unicode format is first Romanized and then the paradigm number is identified by end characters. For sake of easy computation we are using romanized form. A Java program has been written for identifying paradigm number, which is referred as column index. The morpho-lexical information of the required word class is given by the user as input. From the morpho-lexicon information list the index number of the corresponding input is identified, this is referred as row index. A verb and noun suffix tables are used in this system. Using the word class specified by the user it uses the corresponding suffix table. In this two-dimensional suffix table rows are morpho-lexical information index and columns are paradigm numbers.

Input= wordform

1. Unicode = Romanized input
2. Romanized input= Split(stem+Morpholexical form) ~output=stem
3. ML information= pass(paradigm dataset)
4. Paradigm dataset= extract(root suffix)
5. Word= join(stem+root suffix)

6. Output=UNICODE(word)

Flow chart

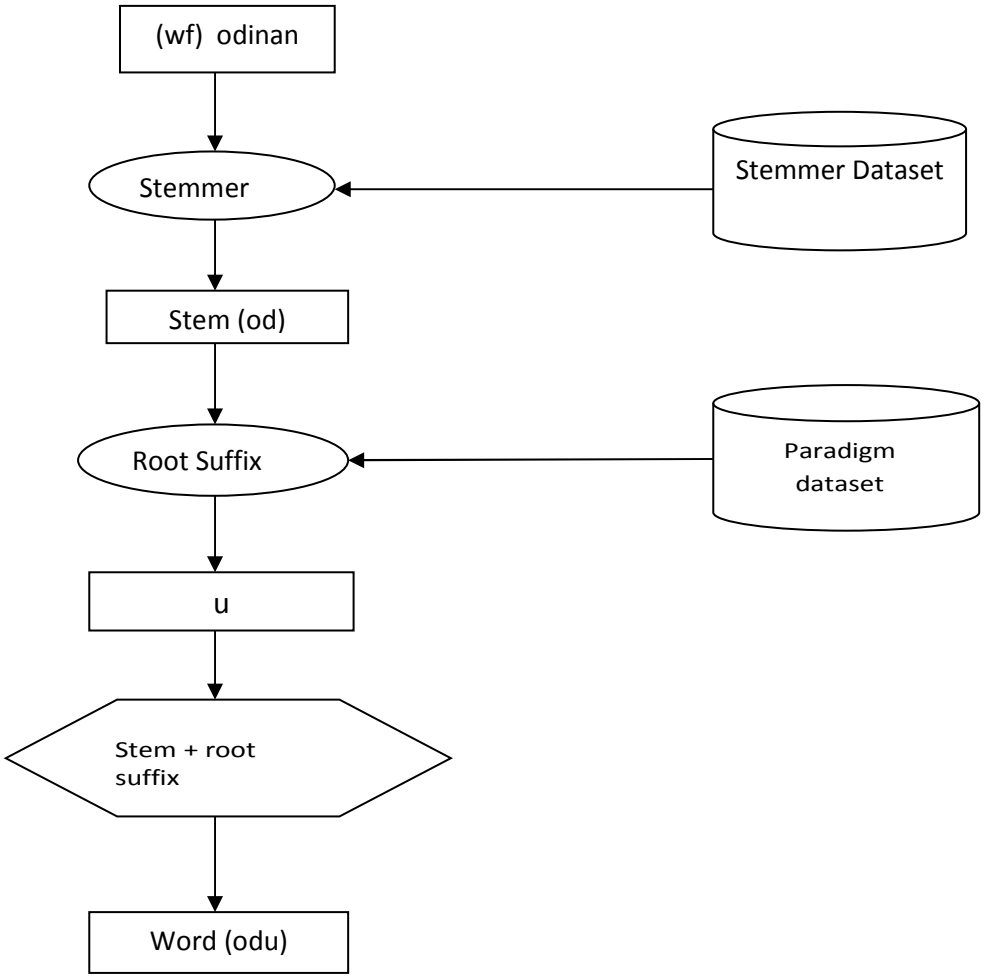


Figure 1. System Architecture

4. Conclusion

Stemming and lemmatization are pre-processing steps in Text Mining applications and basic requirement for many areas in Natural Language Processing, particularly in information retrieval system for improving their performance. The stemmer and Lemmatizer for Tamil is developed using a very simple and efficient method. This is not a language specific method .So this can be applicable for any morphologically rich languages.

Reference

1. Anand Kumar M, Dhanalakshmi V, Rekha R. U, Soman K. P, and Rajendran S, "A Novel Data Driven Algorithm for Tamil Morphological Generator", International Journal of Computer Applications(IJCA) - Foundation of Computer Science, Vol 6(12):52,56, 2010.

2. Dhanalakshmi V, Anand Kumar M, Rekha R, Arun Kumar C, Soman K P and S. Rajendran. "Morphological analyzer for agglutinative languages using machine learning approaches". In Advances in Recent Technologies in Communication and Computing, 2009. ARTCom'09. International Conference on , pages 433–435. IEEE, 2009.
3. Dalwadi Bijal, Suthar Sanket, "Overview of Stemming Algorithm for Indian and Non-Indian Languages", International Journal of Computer Sciences and Information Technologies (IJCSIT) Vol. 5 (2), PP. 1144-1146, 2014.
4. Rajendran, S., Arulmozi, S., Ramesh Kumar, Viswa-nathan, S. Computational morphology of verbal complex. Paper read in Conference at Dra-vidan University, Kuppam, December 26-29, 2001.
5. Vimala Balakrishnan, Ethel Lloyd-Yemoh, "Stemming and Lemmatization: A Comparison of Retrieval Performances", Lecture Notes on Software Engineering, Vol. 2, No. 3, August 2014.

தரமான பேச்சுத் தமிழ்த் தரவு வங்கியை உருவாக்குதல் - ஒரு மேலோட்டப்பார்வை

முனைவர் சீதா லட்சுமி
ஆசிய மொழிகள் மற்றும் பண்பாடுகள்
தேசியக கல்விக்கழகம, சிங்கப்பூர்

ஆய்வுச்சுருக்கம்

இந்த ஆய்வுக்கட்டுரை தமிழில் வகுப்பறையில் இடம்பெறும் பேச்சுத்தரவை ஆயும் வகையில் அமைந்துள்ளது. எழுத்து தமிழ், பேச்சுத் தமிழ், இந்த இரண்டு வகையையும் தன்னுள் கொண்ட வகை, ஆங்கிலப்பயன்பாடு, பிற மொழிப்பயன்பாடு என அமைந்த வகுப்பறைப் பேச்சு எவ்வாறு அமைந்துள்ளது, அது எத்தகைய கூறுகளைத் தன்னுள் கொண்டுள்ளது, இவற்றைக்கொண்டு தரவு வங்கி அமைப்பதில் எதிர் நோக்கிய சிக்கல்கள் யாவை, அவை குறித்த அடுத்த கட்ட முயற்சிகள் யாவை என்று நோக்கும் வகையில் கட்டுரை அமைகிறது. சிங்கப்பூரைப் போன்ற நாட்டில் தரவு வாங்கி முயற்சி ஏற்படுத்தக்கூடிய தாக்கத்தையும் தரக்கூடிய பயன்களையும் கட்டுரை எடுத்துக்கூறும்,

முக்கிய சொற்கள்: தரவு வங்கி, தமிழை இரண்டாம் மொழியாக கற்றல், கற்பித்தல், தரமான பேச்சுத் தமிழ், சவால்கள்

சிங்கப்பூரில் இந்திய இல்லங்களில் 1980-1990களில் குறைந்துவரும் தமிழ் மொழியின் பயன்பாட்டை(Ramiah, 1991; Kuo & Chan, 2016) அதிகரிக்கும்வகையில் வகுப்பறையில் தரமான பேச்சுத்தமிழ்)Schiffman, 1995, 1998) தேவை என்று 1996இல்(Mahizhnan, 1996) உணரப்பட்டது; பரிந்துரைக்கப்பட்டது. 2005இல் இடம்பெற்ற கல்வி அமைச்சின் தமிழ் மொழிப் பாடத்திட்டம், கற்பித்தல் குறித்த மறு ஆய்வுக்குழு பரிந்துரைத்த தரமான பேச்சுத்தமிழ் வகுப்பறையில் அறிமுகப்படுத்தப்பட்டது(MOE, 2005). சிங்கப்பூரில் தரமான பேச்சுத் தமிழ் என்பது தற்போது தமிழ்க் கல்வியிலும், ஆசிரியர் பயிற்சியிலும் பொதுத் தொடர்புச் சாதனங்களிலும் பயன்படுத்தப்பட்டு வருகிறது. இந்தத் தரமான பேச்சுத் தமிழ் வகை வகுப்பறையில் எவ்வாறு இருக்கிறது என்று ஓர் ஆய்வு மேற்கொள்ளப்பட்டது. தரமான பேச்சுத் தமிழின் தாக்கத்தைப் புலனாயும் வகையில் தமிழில் தரவு வங்கியை உருவாக்குதலும் சிங்கப்பூரில் தமிழாசிரியர்களுக்குத் தரமான பேச்சுத் தமிழ் குறித்த ஆசிரியவியல் வழிகாட்டுதலை வழங்குதலும் (DEV03/15SL) என்ற தலைப்பிலான ஆய்வுத்திட்டம் ஒன்றை மேற்கொண்டோம்.. வகுப்பறைக் கற்றல் கற்பித்தல் தொடர்பான இந்த ஆய்வில் தமிழ் மொழி கற்பித்தலில் தமிழ் மொழிப் பயன்பாடு, ஆசிரியர்-மாணவர் மொழிப் புழக்கம் குறித்தும், தரமான பேச்சுத்தமிழின் (Schiffman 1995, 1998) தாக்கம் **குறித்தும் அமைந்துள்ள இந்த ஆய்வில்**, இங்கு சில ஆரம்ப நிலை முடிவுகள் (Preliminary findings) குறித்தும் **ஆய்வுக்குழு எதிர்ப்பட்ட சவால்கள் குறித்தும்** மேலோட்டமாக நோக்கப்படுகிறது.

ஆய்வு எதற்கு? எப்படி இடம் பெற்றது?

இந்த ஆய்வு சிங்கப்பூரில் தொடக்க நிலை ஒன்று முதல் உயர்நிலை ஐந்து வரை வகுப்புத் தரநிலைக்கு ஒன்று என்ற நிலையில் இடம்பெற்ற வாசிப்பு சார்ந்த பாடங்களை ஒலி, ஒளிப்பதிவு செய்து ஒலி பெயர்ப்பும் மொழி பெயர்ப்பும் செய்து தரவு வங்கியாக உருவாக்குதல் என்ற முறையில் அமைந்தது.

இந்த தரவு இதற்கு முன்பு மேற்கொள்ளப்பட்ட ஆய்வுகளிலிருந்து பெறப்பட்ட தரவுகளுடன் ஒப்பிடும்போது ஒரு முழுமையான அதாவது எல்லா நிலை வகுப்புகளிலும் பெறப்பட்ட தரவுத் தொகுப்பு என்று கூறமுடியும். இதை பயன்படுத்தி ஆசிரியர்களுக்குப் பயிற்சி கொடுப்பதிலும் பாடப்புத்தகங்களை எழுதுவதிலும் கூடுதல் பயன்களைக் கொண்டு வரலாம்; தாக்கத்தையும் உருவாக்கலாம்.

ஆசிரியருடைய பேச்சு, மாணவர்களுடைய பேச்சு பற்றியும், ஆசிரியர் மாணவர்களிடையே வினாக்கள் கேட்கும்போது, மாணவர்கள் பதிலளிப்பதற்குரிய வசதிகளை எவ்வாறு செய்கிறார் என்பது பற்றியும், தமிழ் வகுப்பில் பிற மொழிப்பயன்பாடு, தமிழ் வகுப்பறையில் இடம்பெறும் சொற்கோவை, வாக்கிய பயன்பாடுகள் ஆகியன குறித்தும் இந்த ஆய்வு அலசி ஆராய வழி வகுக்கும். இதனால் என்ன பயன் என்றால், எதிர்காலத் தமிழாசிரியர்களுக்குத் தரவு அடிப்படையில் பயிற்சிகளைத் தந்து அவர்களையும் அவர்களது பட்டத்தொழில் திறன்களையும் மேம்படுத்தலாம். சிங்கப்பூரில் தமிழ் மொழியின் தொடர்ச்சியையும் புழக்கத்தேர்ச்சியையும் நீடித்த பயன்பாட்டையும் இன்னும் கூடுதலாக உறுதி செய்யலாம். அந்த ஆய்வைப் பார்க்கும்போது இங்கு மாணவர்கள் தற்போது ஓரளவிற்கு அதை நன்றாகவே பயன்படுத்துகிறார்கள் என்பதையும், அதைப் பயன்படுத்துவதை அவர்கள் இலகுவாக உணர்வதையும் அறிய முடிகிறது.

இந்த மாணவர்கள் தரமான பேச்சுத்தமிழை வகுப்பில் பேசும்போது அவர்கள் அதாவது தமிழ்மொழியைச் சிந்தையிலிருந்து பெற்று அதைப் பயன்படுத்தும் படிநிலையில் அதிகம் யோசித்துப் பேசவேண்டிய ஒரு நிலை இல்லாமல் இயல்பாகவே, அவர்கள் தாமே தங்களுடைய கருத்தைப் பரிமாறுவதற்கு இந்தப் பேச்சுத்தமிழ் உதவியாக இருக்கிறது என்பதை அறிய முடிகிறது. அதுமட்டுமல்லாமல் அவர்கள் அதிகமாக ஆங்கில வார்த்தைகளைப் பயன்படுத்துவதைக் குறைக்கவும் பேச்சுத்தமிழ் வழி செய்கிறது. தற்போது மக்கள் தொடர்புச் சாதனங்கள், வானொலி பாடல்கள், திரைப்படப் பாடல்கள், பிறகு நகைச்சுவைத் துணுக்குகள் ஆகிய இவையும் அதிகமாகப் பிற நாடுகளில் இருந்து பேச்சுத்தமிழோடு இணைந்து வருவதால், இவர்களுடைய பேச்சுத் தமிழென்பது சிங்கப்பூரில் மட்டுமே வகுப்பறையில் பயன்படுத்தக் கூடிய ஒரு வகை என்றும் சிந்திக்காமல் பரவலாகப் பயன்படுத்தப்பட்டு வரும் ஒரு வகை என்றும், புலம் பெயர்ந்த நாடுகளிலும் பயன்படுத்தப்பட்டு வரும் ஒரு வகை என்றும், அவர்கள் உணரும் வகையில் இந்தத் தரமான பேச்சுத் தமிழ் இருக்கிறது. அதுமட்டுமல்லாமல் இப்பொழுது தமிழ்மொழி மாதம், தமிழ்மொழி விழாக்கள் என்று நடத்தும்பொழுது அங்கும் தமிழ்மொழி அதிகமாகப் பயன்படுத்தப்படுவதால் மாணவர்களுக்குத் தமிழைக் கேட்பதோ வாசிப்பதோ, பேச்சுத் தமிழில் அதைப் பற்றிக்

கருத்துரைப்பதோ சிரமமான ஒன்றாகத் தெரியவில்லை. அதனால் அவர்கள் அதை நன்றாகவே பயன்படுத்துகிறார்கள் என்று நாம் சொல்லலாம்.

மேலும் இந்தப் பேச்சுத்தமிழ் என்று வரும்பொழுது அவர்கள் பல கருத்துகளை இயல்பாக யோசித்துச் சொல்வதற்கு மட்டுமல்ல, மாணவர்களின் வீடுகளிலும் தமிழைப் புழங்குவதால் அதை அவர்கள் கூடுதல் சொற்களுடன் பயன்படுத்துவதை நன்றாக உணரமுடிகிறது. அது சிறப்பான ஒன்றாக இப்பொழுது தெரிகிறது. ஏனென்றால் ஒருகாலத்தில் ஒரு வார்த்தை அல்லது இரண்டு வார்த்தை மட்டுமே பேசுவது அல்லது தலையை மட்டுமே அசைப்பது / ஆட்டுவது அல்லது ஆம் என்று சொல்வது அல்லது இல்லை என்று சொல்வது அல்லது அதிகம் பேசாமல் இருப்பது என்ற நிலை மாறி, இப்பொழுது மாணவர்கள் அதிகமாகப் பேசுகிறார்கள். ஆசிரியர்கள் கேள்வி கேட்டால் அதைப் பற்றிக் கருத்துச் சொல்ல முடிகிறது அல்லது அவர்களாகவே வகுப்பில் ஆசிரியர் ஏதாவது கேட்கும் முன்பே தங்களுடைய கருத்துகளைச் சொல்லும் வகையில் இருக்கின்றது.

இந்த நிலை ஒரு வரவேற்கத்தக்க ஒருநிலை என்பதுடன், ஆய்வு என்று நோக்கும்போது இது ஒரு சிறப்பான ஒரு நிலையாக அதாவது சிறப்பான ஒரு மாற்றமாக, முன்னேற்றமாக நாம் கருதுவதற்கு வழி வகுக்கிறது. மேலும் "இன்று மாணவர்கள் இருபத்தோராம் நூற்றாண்டு மாணவர்கள் என்பதால் அவர்களுடைய படைப்பாக்கத் திறன், புத்தாக்கத் திறன், மென்திறன், வழக்கத்திற்கு மாறாக புதிய வகையில் சிந்தித்தல், எதையும் ஏன்? எதற்கு? எப்படி? என்ற கருத்தாய்வு நிலையில் சிந்தித்தல்" என்று கல்வியமைச்சு தன்னுடைய 2015 ஆம் ஆண்டு வெளியிட்டுள்ள பாடத்திட்டப் புத்தகத்தில்)ப. 43) குறிப்பிட்டது போல அவர்கள் பல நிலைகளில் சிறப்பாகச் சுய காலில் நிற்க வேண்டியவர்களாக இருக்கும்போது இந்தப் பேச்சுத்தமிழ் அவர்களுடைய தொடர்புத்திறன்களை வளர்க்கப் பெரிதும் உதவுகிறது. ஒரு தமிழரைப் பார்த்தால் மற்றொரு தமிழர் இயல்பாகப் பேசக்கூடிய ஒரு நிலை இருக்கிறது. தமிழ்நாட்டில் உள்ள ஒரு தமிழருடன் பேசுவதற்கு இந்த மாணவர்களால் இயல்பாக முடியும் என்று கூறும் அளவிற்குப் பெரும்பாலான மாணவர்கள் வகுப்பறையில் பேசுவதை இந்த ஆய்வு எடுத்துக் காட்டுகிறது) அட்டவணை 1.(

. அட்டவணை 1: ஆசிரியர் - மாணவர் பேச்சு

பேசுபவர்	**பேச்சு**
ஆசிரியர்	பளிச்சுனு இருக்கற மாதிரி நிறங்கள போட்டுருப்பாங்க.. சரி வேற?
மாணவர்	இந்த இந்த சட்டைலாம் இந்த இந்த மாரி சட்டைலாம் fold..
ஆசிரியர்	ம்.. அது என்னது?
மாணவர்	மடிச்சி

அதுமட்டுமல்ல இந்த மாணவர்களுக்கு அந்தப் பாடங்கள் அவர்களுக்குப் பிடித்தமான முறையில் அவர்களுடைய வாழ்க்கையோடு தொடர்புள்ள முறையில், இன்னும் சொல்லப்போனால் சிங்கப்பூர் சார்ந்த முறையில் இருந்தால் இன்னும் எளிமையாக இருப்பதை அறிய முடிகிறது.

ஏனென்றால் அவர்களுக்கு அந்தப் பாடம் தொடர்புடைய சொற்கள், அதன் பின்னால் உள்ள கருத்தமைவுகள் (concepts) ஆகியனதாம் காரணம் என்று கூறமுடியும்.

தமிழ் வகுப்பில் புதியன கண்டறியும் திறன், மாணவர்கள் அதிகம் பேச வழி வகுக்கும் அணுகுமுறை, கருத்தமைவு அடிப்படையிலான பாடக்கலைத் திட்டமும்/ பாடத் திட்டமும் அணுகுமுறையும் (Little, 2017, 2017) கருத்தமைவு அடிப்படையில் கற்பித்தலும் ஆகிய பரிந்துரைகள் மாணவர்கள் இன்னும் கூடுதலாகப் பேச வழிவகுக்கும் என்று கூறப்பட்டது.

ஆசிரியர்கள் பாடத்தைக் கற்பிக்கும்போது மாணவர்களுக்குப் பாடம் தொடர்பாகக் கூடுதல் செய்திகள் கிடைக்கும் வகையில் அவரது விளக்கப்பேச்சு(Explanatory Talk) அதிகரிக்க வேண்டும். இது மாணவர்கள் தமிழ்ச் சொற்களையும் மொழியையும் நீண்ட கால நினைவுப்பெட்டகத்தில் சேமித்து வைக்கவும், சேமித்த மொழியை இயல்பாகப் பயன்படுத்தித் தங்கள் நண்பர்களுடன் உரையாடவும் வழி கிடைக்கும். மாணவர்கள் பேசும்போது ஓரிரு சொற்களுக்குப் பதிலாகக் கூடுதல் சொற்களைப் பயன்படுத்திப் பேசுவது அவர்களுடைய மொழிச் செயல்பாட்டையும் எடுத்துக்காட்டுவதை அறிய முடிந்தது) அட்டவணை 2).

அட்டவணை 2: மாணவர் பேச்சு

பேசுபவர்	**பேச்சு**
மாணவர்	முதவாட்டி இத பார்க்கும் போது சோகமான பாட்டுமாதிரி இருக்கு ஏன்னா எல்லாரும் அழ ஆரமிச்சிட்டாங்க
மாணவர்	“ஆசிரியர், கழுத்து உடஞ்சிருமா?
மாணவர்	இது வந்து தினமும் வந்து போடலாமா? இல்லை வருஷத்திற்கு ஒரு குறிப்பிட்ட நாளுலே போடலாமா?”

ஆசிரியரும் மாணவர்களும் தமிழைப் பயன்படுத்திப் பேசினாலும் தம் தேவைக்கேற்பவும் சில சமயங்களில் வலிந்தும் இடையில் ஆங்கிலத்தையும், பயன்படுத்தும் நிலையும் வகுப்பறையில் இடம் பெறுகிறது (அட்டவணை 3).

அட்டவணை 3: தமிழுடன் சில இடங்களில் ஆங்கிலமும் இடம்பெறும் நிலை

பேசுபவர்	**பேச்சு**
ஆசிரியர்	அவர் தன்னைத்தானே நொந்துக்கிறார். வயசாகீட்டாலே, கண்ணு சரியா தெரியமாட்டேங்குது, காது சரியா கேட்கமாட்டேங்குது, முட்டி வலிக்குது, நடக்க முடியல, அது இருக்கு, இது இருக்கு, வயசானாலே இப்படித்தான் இருக்கும்னு தன்னைத்தானே புலம்பிக்கிட்டு நொந்துக்கிறார். சட்டைப்பையில என்ன வச்சிருந்தாரு, கண்ணாடியை எடுத்துப்போட்டுக்கிறார், யாரோ ஒரு தெரிஞ்ச ஆள் மாதிரி தெரியுது, ஆனா

	தெளிவாத் தெரியமாட்டேங்கு. வர்றவன் சுப்பிரமணி மாதிரி தெரியுது, அவந்தானான்னு தெரியலை, சரி வயசானாலே கண்ணு இப்படித்தான் இருக்கும்போலன்னு நெனைச்சி உடனே கண்ணாடி எடுத்துப்போட்டுக்கிறாரு, யாருன்னு பாக்கறதுக்காக சட்டைப் பையில இருந்து கண்ணாடி எடுத்துப் போட்டுப் பார்க்கிறாரு. சரி இந்தப்பகுதியில நான் ஒரு விளக்கம் சொல்லனூன்னு நினைக்கிறேன். இந்த வார்த்தைய பாருங்க.
மாணவர் 1:	மூக்குக்கண்ணாடி
ஆசிரியர்	ஆ இது நிச்சயமா தெரிஞ்சிருக்கு இல்ல? கண்ணாடி, இந்த வார்த்தையோட பழைய அர்த்தத்த சொல்லப் போறேன். இப்ப கண்ணாடின்னா எத சொல்வோம்?
மாணவர் 1:	)தெளிவாக இல்லை) கண்ணாடி
ஆசிரியர்	அப்ப கண்ணாடி இருக்கு, ஜன்னல்ல கண்ணாடி இருக்கு, இப்ப அத சொல்றதுக்கு additionalலா வார்த்தைங்க தேவப்படுது. ஆனா, பழை காலத்துல ஆடின்னு சொன்னாலே mirrorன்னு அர்த்தம். ஆடி, car பேரு இல்ல. ஆடினாக்கா, mirror முகம் பாக்கற கண்ணாடி
மாணவர் 1:	ஆடி மாதம்
ஆசிரியர்	ம்ம் அதுக்கும் அதே spelling தான்.
மாணவர் 1:	அப்ப ஆடி (தெளிவாக இல்லை)
ஆசிரியர்	இரு இரு இதுக்கு வர்றேன், ஒன்னொன்னா clear பண்ணிட்டு போயிடுவோம். ஆடில முகம் பார்த்தாங்கன்னா கண்ணாடில முகம் பாத்தாங்கன்னு அர்த்தம் பழைய காலத்துல. இப்ப புரியாது. அந்த ஆடி போயி அப்புற இந்த ஆடி வர ஆரம்பிச்சது. அதுக்கும் இதுக்கு எப்படி மாத்தறது? அப்ப இத differ பண்றதுக்காக என்ன பண்ணாங்க, கண்ணாடின்னு ஆக்கனாங்க. கண் plus ஆடி எப்படி வந்துடும்?
மாணவர் 1:	கண்ணாடி

இந்த ஆய்வில் ஈடுபடும்போது சில சவால்களையும் சிக்கல்களையும் எதிர்நோக்கினோம். அவை ஆய்வின் பிற்பகுதியில் சிறிது தாமதம் ஏற்படக் காரணமாயின எனலாம். இவ்வகையிலான சவால்களும் சிக்கல்களும் வருமாறு;

- ஏற்ற தமிழ்-ஆங்கிலம் தெரிந்த ஆய்வு உதவியாளரைத் தேடிக் கண்டுபிடித்தல்
- அவ்வாறு கிடைத்தாலும் தமிழை வாழ்நாள் தொழிலாக அவர் ஏற்கும் நிலையில் இல்லாத / இயலாத பின்னணி

- ஆய்வு உதவியாளருக்கு ஆய்வு நோக்கிய திறன்களை மேம்படுத்துதல், அவருக்கு ஆய்வு சார்ந்த தொடர்ச்சியில் உள்ள நம்பிக்கை, அதைத்தொடர்ந்து வெளிப்படுத்துவதில் இருந்த தொலைநோக்கு ஆகியன
- ஆய்வு உதவியாளர் கிடைத்தாலும் ஆரம்பத்தில் தரவு சேகரிப்பதற்கான பள்ளிகள் கிடைப்பதில் சுணக்கம்
- பள்ளிகள் கிடைத்தாலும் தவிர்க்க இயலாத நிலையில் தரவு சேகரிப்பதில் தென்பட்ட நேர மெதுவடைவு
- தரவை ஒலிபெயர்த்து முடிக்குமுன் ஆய்வு உதவியாளரின் ஒப்பந்தம் நிறைவு பெறும் நிலை
- ஒலி பெயர்ப்பிலும் மொழி பெயர்ப்பிலும் தேர்ச்சி பெற்ற பகுதி நேர ஆய்வு உதவியாளர்கள் கிடைப்பதிலும் சீரிய செயல்பாட்டைப் பெறுவதிலும் தென்பட்ட சிரமங்கள்
- கணினியில் ஆய்வுத் தரவைத் தரவு வங்கியாக மேம்படுத்துவதில் மென்பொருள் தந்த தொடர்ச்சியும் நீட்சியும் இல்லா நிலை
- இதனால் இயந்திரத்தை விட மீண்டும் மனித வழியில் தரவை ஒழுங்குபடுத்திச் சேமிக்க முனைதல்
- தரவைச் சிங்கப்பூரிலேயே பகுப்பாய்வு செய்வதற்கு ஏற்ற காலம் கிடைக்காமை
- தரவைச் சிங்கப்பூரிலேயே பகுப்பாய்வு செய்வதற்கு வசதியாக இங்கேயே பணிபுரியக் கூடிய, தகுதி வாய்ந்த தமிழ் இளையரை அடையாளம் காண இயலாமை
- உடனடியாகப் பயிற்சி அளிக்கவும் இயலாத இக்கட்டான நிலை
- மென்பொருள் கிடைப்பதிலும் சிறிது பின்னடைவு
- ஆங்கிலத்தில் செய்த அளவிற்குத் தமிழில் இத்தகு மேம்பாடு அல்லது அது குறித்த பரவலான தகவல்கள் கிடைக்காமை
- இதனால் ஆய்வுக்கான காலத்தை நீட்டிக்க அனுமதி கோரல்
- அயல் நாட்டு ஆய்வறிஞரைச் சிங்கப்பூருக்கு வரவழைப்பதில் எதிர்பாராது இடம்பெற்ற ஒரு சிரமம்.
- விருப்பம் இருந்தும் தன்முனைப்பு இருந்தும் ஏற்ற நேரம் கிடைக்காமை

இவை வெறும் தொழில் நிலை சவால்களும் சிக்கல்களும் மட்டும்தான். ஆய்வுத்தரவைச் சேகரிப்பது மட்டும் பெரிதல்ல. அதை ஒழுங்குபடுத்தி வடிவமைப்பதும் ஓர் இன்றியமையாப் படிநிலை. எதிர்காலத்துக்காக அதைச் செம்மைப்படுத்தி வைப்பதும் இங்கு இன்றியமையா ஒரு படிநிலை, எனினும் முடிவில் எங்களது ஆய்வுக்குழுவில் ஒருங்கிணைப்பாளராக இருக்கும் முனைவர் வாசு அரங்கநாதன் காட்டிய தரவு ஒழுங்குபடுத்தும் வழியை இப்போது செயல்படுத்திப் பார்க்கும் நிலையில் உள்ளோம்.

வகுப்பறையில் திரட்டப்பட்ட தரவுகளை ஆராய்ந்து அதைப் பயன்படுத்திப் பல கட்டுரைகளை எழுதுவது மிகவும் அவசியம். அது குறித்த சில செயல்பாடுகள் தொடர்ந்து இடம் பெறுகின்றன. இத்தகு ஆய்வுகள் புலம் பெயர்ந்த தமிழர்களுக்கும் பேருதவியாக அமைபவை. நிதியுதவியுடன் கூடிய பல ஆய்வுத்திட்டங்களை வழி நடத்தியதுடன் அவற்றில் ஐம்பதுக்கும் மேற்பட்ட கல்வியாளர்களுடனும் ஆய்வறிஞர்களுடன் வேலை செய்த அனுபவம் பெற்றிருந்தாலும்

அவ்வப்போது மிகுந்த மன வருத்தம் உண்டானாலும் தொடர்ந்து ஈடுபடுவதில் கட்டுரையாளருக்கு விருப்பமும் ஈடுபாடும் அதிகமாக இருப்பது குறிப்பிடத்தக்கது.

எதிர்பார்ப்புகளும் சாத்தியமாகக்கூடிய தீர்வுகளும்

சவால்கள் எப்போதும் தீர்க்கமுடியாச் சிக்கல்களல்ல; அவை சிறந்த தீர்வையும் முன்னேற்றத்தைக் காட்டும் கலங்கரை விளக்கங்கள். அவ்வகையில் சில கூறுகள் குறித்து ஆழமாகச் சிந்திக்கலாம்:

- மாணவ ஆசிரியர்களுக்கு கணினியின் வழியே மொழியியலைக் கற்பிக்கலாம்
- தரவு வங்கி குறித்த ஆய்வுத்திட்டங்களைக் கொடுக்கலாம்
- தரவு வங்கி குறித்த விழிப்புணர்வைக் கொடுத்து, அதை மெல்ல அதிகரிக்கலாம்
- திறமையான இளம் கணினி மொழியியலாளர்களை இங்கு உருவாக்கி இங்கு அதிகமாகச் செயல்பட வழி வகுக்கலாம்.

முடிவுரை

சிங்கப்பூரில் தமிழ் வகுப்பறைகளில் தமிழ்மொழியின் பயன்பாடு கடந்த 2003 முதல் 2018 வரை எவ்வாறு அமைந்துள்ளது என்பதை பல ஆய்வுத்திட்டங்கள் வழியே தரவுகளாகக் கொடுத்திருக்கிறோம். தற்போது பாலர் பள்ளியிலும் இத்தகைய தரவுகள் எவ்வாறு அமைந்துள்ளன என்று நோக்குவது இடம் பெற்று வருகிறது. அவ்வகையில் இது குறிப்பிடத்தக்க ஒரு வாய்ப்பாகும்.

தரவு வங்கி என்பது ஒரு மொழியின், சமூகத்தின், நாட்டின் வரலாற்றையும் அது சார்ந்த பண்பாட்டுக்கூறுகளையும் எடுத்து கூறும் ஆவணம். அதிலும் வகுப்பறை சார்ந்த உயிரோட்டமான இத்தகு ஆவணத்தை முறையாய் ஆராய்ந்து மேம்பாடுகளையும் வளர்ச்சியையும் அறிமுகப்பதி செயற்படுத்துவது அவசியம். அத்துடன் எதிர்கால ஆராய்ச்சிக்கு வழி காட்டும் வகையில் முறையாக சேமித்து வைப்பதும் இன்னும் பெரும் பயனை வரவிருக்கும் பல நூற்றாண்டுகளில், அப்போது வாழவிருக்கும் பல தலைமுறைகளுக்கும் உதவியாக இருக்கும். மனித குலத்திற்கு உதவும் அந்த அற்புதமான நம்பிக்கைதான் ஆய்வுலகைத் தொடர்ந்து மேம்படுத்தி வருகிறது என்றால் மிகை இல்லை.

மேற்கோள் நூல்கள்:

1. கல்வி அமைச்சு. (2015). தமிழ் மொழிப் பாடத்திட்டம் 2015 தொடக்கநிலை. சிங்கப்பூர்: கல்வி அமைச்சு. இணையப்பக்கம்: https://www.moe.gov.sg/docs/default-source/document/education/syllabuses/mother-tongue-languages/files/tamil-primary-2015.pdf
2. Kuo, C Y Eddie and Chan, B. (2016). *Language*. Singapore: Institute of Policy Studies.
3. Little, C. A. (2017). Designing and Implementing Concept-Based Curriculum: An International Perspective. In L. S. Tan, L. D. Ponnusamy, & C. G. Quek (Eds.), *Curriculum for High Ability Learners: Issues, Trends and Practices* (pp. 43-59). Singapore: Springer.
4. Ministry of Education, Singapore. (2005). *Report of the Tamil Language Curriculum and Pedagogy Review Committee,* Singapore: Ministry of Education.
5. Mahizhnan, Arun. (Ed.) (1996). Report of the Tamil Education Review Committee. Singapore: SINDA.
6. Ramiah, K. (1991). The pattern of Tamil language use among primary school Tamil pupils in Singapore. Singapore Journal of Education, 11(2), 45-53.
7. Schiffman, H F., (1995). Question of language maintenance and language shift in Malaysia and Singapore, published in *Language Loss and Public Policy,* I , Garland Bills (ed.), *Southwest Journal of Linguistics*, 14, Nos. 1-2, 151-165.
8. Schiffman, H F., (1998). Standardization or Restandardization: The Case for "Standard" Spoken Tamil. *Language in Society*. 27, no.3: 359-385.

Acknowledgements:

This paper refers to data from the research "Creating a Corpus Data Bank (CDB) in Tamil to investigate the Impact of Standard Spoken Tamil (SST) and to provide pedagogical guidance to Tamil Teachers on Standard Spoken Tamil in Singapore"(DEV 03/15 SL) funded by the Education Research Funding Programme, National Institute of Education (NIE), Nanyang Technological University, Singapore. The views expressed in this paper are the author's and do not necessarily represent the views of NIE.

நன்றி!

இந்தக் கட்டுரையின் ஒரு பகுதியைத் தட்டச்சுச் செய்வதிலும் மெய்ப்புத் திருத்தம் செய்வதிலும் உதவிய முனைவர் சா சுந்தரராஜனுக்கு என் நன்றி.

CORPUS ANALYSIS OF `-*pōla*$_{]suf.}$' and `-*mātiri*$_{]suf.}$'

Dr. Ganesan Ambedkar

Dravidian University
Kuppam – 517 426

1. Grounding the problem

Tamil, like many other living languages, has many forms to denote functions of comparative narratives to a speaker. Tamil society and language is no exemption to this well known axiom, as in the data, below:

1) "vāḻntā avaṉ *mātiri* vāḻaṉum" eṉṟu poṟumikkoṇṭāṉ.
2) "tāy*aip pōla* piḷḷaiṉṉu" coṉṉatu caritāṉ.
3) "nī eṉṉa avaṉ*aiviṭa* periya āḷā?" eṉṟatum rājaṉiṉ mukam curuṅkiyatu.
4) "eṉ taṅkacci avaḷu*kku mēla* yārum varakkūṭātuṉṉu irukkaṟava"
5) avaruṭaiya aṟivu*kku muṉṉāl* uṉ aṉupavam vēlaikkākātu
6) avaṉu*kku* eppaṭi aṟivu vantatu?
7) tavaḷaikk*um* kaḻutaikk*um* kalyāṇam.
8) pēṭṭaiy*aik kāṭṭilum* vicuvācattiṟku vacūl atikamā?
9) vijay cētupatik*ku patilā* pacupati naṭiccā nallāyiruntirukkumō?
10) aṉṉaṇōṭa tampiy*ai kampēr paṇṟatu*, nallavāyirukku?
11) ṭuleṭ. itu*kku mēla* evaṉāvatu irāṉ paṭam, koriya paṭamṉṉu vantīṅka....
12) uṅkappā *muṉṉāla* nī emmāttiram. avarukku nī camamā?
13) iru kuḻantaikaḷai *oppiṭ*ātē
14) karuṇāniti*kku kīḻa* yārum vaḷara muṭiyātāmē?

The above illustrative data shows that the *italicized* forms represent tokens of comparisons, and are used for comparison in day-to-day-life.

2. Types of Comparison

Tamil language has all three standard degrees of comparison: 1) superlative (>), 2) same degree (=) and 3) lower degree (<). In the set (1), below, examples are of superlatives, and in the (2) examples are of same degree and in the (3) examples are of lower degree, as in below:

1

1) ṭāṭṭāvu*kku mēla* evaṅkiṭṭa paṇamirukku?
2) viyāparattila, putup paṇakkāraṉ tāṉ taṉa*kku mēl* evaṉum varakkūṭātuṉṉu naṭappāṉ. āṉā, ṭāṭāvai pāru, pirlāvai pāru. eṉṉa oru aṭakkam.
3) amiḻtu*kku mēla* eṉṉayirukku?
4) kaṭaṉu*kku mēla* kaṭaṉ vāṅkiṉā, curukku tāṉ ...
5) vāṉattu*kku mēla* eṉṉappā eṉṟu makaḷ kēṭṭatum, curukkeṉṟatu maṉam

2

1) icaiyil iḷaiyarājāvu*kku nikar* iḷaiyarājāvē
2) timukāviṉ mumperum talaivarkaḷil, karuṇāniti*kku camam* karuṇānitiyē. periyārum aṉṉāvum vēṟa leval. oppiṭakkūṭātu.
3) civakōttirattiṟ*ku iṇai*yāṉa cāmikōttiraṅkaḷ
4) naṭippil civāji*kku nikar* civājiyē.

3

1) nāy*kku kīḻ*ayiruntā ,vālai pārttappellām, āṭṭittāṉ kāriyam cātikkaṇum
2) kuraṅkukk*um* maṉucaṉukk*um naṭuvil* varṟavaṉaip patti eṉakkeṉṉa?
3) kāllu*kku kīḻ* cūttirarkaḷ

In the above sentencial Tamil examples, the italized forms, syncretistic to ego-centric coordinates or spatio-temporal-coordinates or case-cum-post-position-markers, display the presence of above mentioned three layers of comparison. Out of the above given and italicized twelve tokens through all the above examples, `-*pōla*$_{]suf.}$' and `-*mātiri*$_{]suf.}$' are the subject matter in this paper.

2.1. The Aim

Consider the following two sets of data that are syntatically identitical except the presence and absence of comparatives -*pōla*$_{]suf.}$ and -*mātiri*$_{]suf.}$

I

1) civāji *mātiri* naṭikka vēṇṭām.
2) rajiṉi *mātiri* pēca vēṇṭām.
3) kuḻantai uṅkammā *mātiri*yirukku eṉṟavuṭaṉ cirippalai eḻuntatu.
4) nāy *mātiri* kulaikkātē eṉṟu captamāka tiṭṭiṉāḷ
5) caṉ ṭivi *mātiri* pāppular ṭivi eṅkēyirukku?

II

6) civājiy*aip pōla* naṭikka vēṇṭām.
7) rajiṉiy*aip pōla* pēca vēṇṭām
8) kuḻantai uṅkammāv*aip pōla*yirukku eṉṟavuṭaṉ cirippalai eḻuntatu.
9) nāy*ai pōla* kulaikkātē eṉṟu captamāka tiṭṭiṉāḷ
10) caṉ ṭiviy*ai pōla* pāppular ṭivi eṅkēyirukku?

From these two sets of same data, one can surmise and show that 1) these two tokens are mutually replaceable, 2) -*pōla*$_{]suf.}$ is phonologically conditioned, 3) -*mātiri*$_{]suf.}$ has more frequency in spoken variety of Tamil and 4) -*pōla*$_{]suf.}$ has more frequency in written or +H variety of Tamil.

But, corpus analysis of these tokens portray an altogether different picture, i.e. usages differ vastly one against another. Thus, the aim of this article is to demonstrate usage differences between these two comparatives by showing examples from Tamil corpus.

2.2. With Grammatical Categories

Distribution of these suffixes across grammatical categories, such as nouns, verbs and verbal adjectives, shows that `-*mātiri*$_{]suf}$ appears, predominantly, after nouns and verbal adjectives, as in the examples below:

1) āccariyappaṭṭa$_{]v.adj}$ *mātiri*$_{]suf}$
2) accaṭicca$_{]v.adj}$ *mātiri*$_{]suf}$

3) acaikiṟa]v.adj *mātiri]suf*
4) acainta]v.adj *mātiri]suf*
5) mukkāṭu]n *mātiri]suf*

Note, the use of same set of examples of `*-mātiri'*]suf *on* `*-pōla'*]suf generates *ungrammaticality* in the examples bearing numbers from (1) to (4) and , *grammaticality* in the example bearing number (5).

1) *āccariyappaṭṭa]v.adj *pōla]suf.*
2) *accaṭicca]v.adj *pōla]suf.*
3) *acaikiṟa]v.adj *pōla]suf.*
4) *acainta]v.adj *pōla]suf*
5) *mukkāṭu]n pōla]suf*

Based on the above examples, a surmise can be drawn: the suffix '*-pōla*]suf.' appears after nouns, and never appears with the grammatical category of verbal adjectives. In the above example, starred one (*), alone, is ungrammatical.

2.1.2. With nounhood suffixes

With nounhood making suffixes that are around 148 in number, as in the below inventory, the comparative suffixes *pōla]suf and mātiri]suf* do not appear, either. Illustrative examples are below:

1) *avaṉ*ai aṭutta mātiri* pārkkavillai

2) *avaṉ*ai aṭutta pōla* pārkkavillai

3) *avaṉ*aic cuṟṟi mātiri* ētuvumillai

4) *avaṉ*aic cuṟṟi pōla* ētuvumillai

5) *avaṉ*aik kaṇṭu mātiri*yillai

6) *avaṉ*aikkaṇṭu pōla*villai

7) *avaṉ*aimīṟi mātiri*yillai

8) *avaṉ*aimīṟi pōla*ētumillai

The starred-ones generate *un*grammatical sentences, as in the examples above. In a similar fashion, it is found in the corpus that the starred-ones-alone generate *un*grammatical sentences to the inventory of suffixes, given below:

1) -ai, 2) *-ai_aṭutta, 3) *-ai_aṭuttu, 4) *-ai_eṅkēyō, 5) *-ai_cuṟṟi, 6) *-ai_kaṇṭu, 7) *-ai_mīṟi, 8) *-ai_nōkki, 9) *-ai_toṭarntu, 10) *-ai_viṭṭu, 11) *-arukil, 12) *-arukiliruntu, 13) *-aṭiyil, 14) *-aṭiyiliruntu, 15) *-etir, 16) *-etiriliruntu, 17) *-etiril, 18) *-iṭaiyē, 19) -iṭam, 20) *-iṭamiruntu, 21) *-iṭattil, 22) *-iṭattiliruntu, 23) *-kāṭṭi, 24) *-kaṭaiciyil, 25) *-kaṭaicikku, 26) *-kaṭaicikkuḷ, 27) -kīḻ, 28) *-kīḻē, 29) *-kīḻiruntu, 30) -kkāṉa, 31) *-kku-arukāmaiyil, 32) *-kku-antap-pakkam, 33) *-kku_arukil, 34) *-kku_pakkattil, 35) *-kku_aṭutta, 36) *-kku_aṭuttu, 37) *-kku_etirāka, 38) *-kku_etiril, 39) *-

kku_etiriliruntu, 40) *-kku_intap_pakkam, 41) *-kku_iṭaiyē, 42) *-kku_iṭaiyil, 43) *-kku_iṭaiyiliruntu, 44) *-kku_kīḻē, 45) *-kku_mēliruntu, 46) *-kku_mīṟi, 47) -kku_nēr, 48) -kku_nēr_etir, 49) *-kku_nēr_etiriliruntu, 50) *-kku_nēril, 51)*-kku_pakkam, 52) *-kku_pakkattil, 53) *-kku_pakkattiliruntu, 54) *-kku_puṟampāṉa, 55) *-kku_veḷiyē, 56) *-kkaṭiyil, 57) *-kkum_mattiyil, 58) *-kuṟukkē, 59) *-kuṟukkēyirunta, 60) *-kūṭa, 61) *-mattiyiliruntu, 62) *-mattiyil, 63) *-mēl, 64) *-mītu, 64) *-mītiruntu, 65) *-mūlamāka, 66) *-naṭuvil, 67) *-naṭuviliruntu, 68) -nēr_etir, 69) *-nēr_etirāka, 70) *-nēr_etiril, 71) *-nēr_etiriliruntu, 72) *-neṭuka, 73) *-nōkki, 74) *-ōramāka, 75) *-orattil, 76) *-ōrattiliruntu, 77) *-pakkattil, 78) *-pakkattiliruntu, 79) *-piṉṉāliruntu, 80) *-uḷ, 81) *-uḷḷiruntu, 82) *-ūṭē, 83) *-vaḻiyāka, 84) *-vaḻiyil, 85) *-vaḻiyiliruntu, 86) *-veḷiyē, 87) *-veḷiyil, 88) *-veḷiyiliruntu, 89) *-āka, 90) *-mutalil, 91) *-mutaliliruntu, 92) *-piṉpu, 93) -pōtu, 94) -um_pōtu, 95) *-um_pōtiliruntu, 96) -um_varai, 97) *-ai_etirpārttu, 98) *-ai_kaṭantu, 99) *-ai_tāṇṭi, 100) *-il, 101) *-illiruntu, 102) *-kkāka, 103) -kku, 104) *-kku_appāl, 105) *-kku_naṭuvil, 106) *-kku_piṟaku, 107) *-kku_appuṟam, 108) *-kku_muṉpāka, 109) *-kku_piṉpu, 110) *-kkuḷ, 111) *-kku_mēl, 112) *-kkum, 112) *-kku_muṉṉāl, 113) *-kku_muṉpu, 114) *-kku_piṉ, 115) *-kku_piṉṉāl, 116) -muṉ, 117) -muṉṉāl, 118) -muṉpu, 119) *-mutal, 120) -muḻuvatum, 121) *-ōṭu, 122) *-piṉṉāl, 123) *-uṭaṉ, 124) *-varai, 125) *-āka, 126) *-āl, 127) *-ālāṉa, 128) *-āṉa, 129) *-iṉ, -130) *-uṭaṉ, 131) *-uṭaṉirunta, 132) *-uṭaiya, 133) -um, 134) *-umcērntu, 135) *-um_viṭa, 136) *-uḷpaṭa, 137) *-ai_kāṭṭilum, 138) *-ai_koṇṭu, 139) *-ai_tavira, 140) *-ai_pōl, 141) *-ōṭu, 143) *-ōramāka, 144) *-kku_ātaravāka, 145) *-kku_iṇaṅka, 146) -kku_ēṟṟa, 147) -kku_ēṟpa, 148) *-kkāṉa

2.3. Usage and nature of comparison

These tokens are of useful to distinguish between a) direct comparison, b) clarificatory comparison and c) illustrative comparison, as mentioned and adopted in this research article, from the multiple references of given under the reference section, in Tamil. Consider the examples below:

1) kuḻant*ai mātiri* pūṉai kattuccu.

2) kuḻantaiy*aip pōla* pūṉai kattuccu

3) ammā *mātiri* eppaṭiṭā irukkamuṭiyum?

4) ammāv*aip pōla* eppaṭiṭā irukkamuṭiyum?

The above examples illustrate a direct comparison of two entities by giving bi-focal importance. Observe, the given examples demonstrate no particular importance to any of the said two entities, i.e. two comparables, for instance, cry of a cat and its similarity to the cry of a child: Two similar entities between loud, pitch, energy, manner and tone, etc. On account of these facts, one may consider and conclude that they are demonstrative examples to the clarificatory comparison, too. But, consider the examples below:

1) tāyaip*pōla* piḷḷai. nūlaip*pōla* cēlai

2) vāṉattai*p pōla* maṉam paṭaitta maṉṉavaṉē

These above examples give evidence to illustrative comparison. In these illustrative comparison examples, note that the suffix *-pola*]$_{suf}$ is used for nonsimilar entities. In other words, out of many qualities of an entity, similarity and compared to one quality can be found through the suffix *-pōla*]$_{suf.}$

Thus, motherhood may have more internal structural characters, such as decision making to manner of address to adjustment to speaking to manners of dress, etc. Thus, usage of *-pōla* is acceptable.

Where as, out of many qualities, similarities to maximum number of qualities can be found through the suffix *-mātiri*]$_{suf}$. The evidence for such crucial distinction comes from the ungrammaticality of the below sentences:

1) *tāy (ai) **mātiri** piḷḷai. *nūl mātiri cēlai

2) *vāṉam ***_mātiri_*** maṉam paṭaitta maṉṉavaṉē

In short, out of one against many is the quality of the suffix *-pōla* and out of more against many is the quality of the suffix *-mātiri*.

To strengthen the above said claim with further evidence, thus, one may find acceptablity and unacceptablity to the usages below, because the suffix – *pōla,* which is used for comparison of one against many, gives grammaticality to the sentences three and five, where as the sufix *-mātiri,* never occurs in those comparisons, because it is used for more against most in comparisons. Thus

3) maṉam *pōla* vāl̲kkai

4) *maṉam *mātiri* vāl̲kkai

5) pāl *pōla* maṉam

6) *pāl *mātiri* maṉam

3. Conclusion

From the above sections, one may surmise with evidence that there are *qualitative* usage differences between these two suffixes. The suffix *-mātiri*]$_{suf}$ is used for comparison of mapping between more against most, and the suffix *-pōla]$_{suf}$* is used for comparison of mapping between one against many.

References

Annamalai, E. 1997. Adjectival Clauses in Tamil. Tokyo: Tokyo University.

Dale, R., and Milosavljevic, M. (1996). Authoring on demand: Natural language generation of hypermedia documents. In *Proceedings of the First Australian Document Computing Symposium (ADCS'96)*. Melbourne, Australia.

Dale, R., Milosavljevic, M., and Oberlander, J. (1997). The web as dialogue: The role of natural language generation in hypertext. In *Proceedings of the AAAI Spring Symposium on Natural Language Processing for the World Wide Web*, 35-43. Stanford University, California.

Encyclopædia Britannica. (1996). Copyright ©1996 Encyclopædia Britannica, Inc.

McCoy, K. F. (1995). Generating context-sensitive responses to object-related misconceptions. *Artificial Intelligence* 41:157-195.

McKeown, K. R. (1985). Discourse strategies for generating natural-language text. *Artificial Intelligence* 27:1-41.

Microsoft (R) Encarta'95 Encyclopedia. (1995). Copyright ©1994 Microsoft Corporation. Copyright ©1994 Funk and Wagnall's Corporation.

Milosavljevic, M. (1996). Introducing new concepts via comparison: A new look at user modeling in text generation. In *Proceedings of the Fifth International Conference on User Modeling, Doctoral Consortium*, 228-230.

Milosavljevic, M., and Dale, R. (1996). Strategies for comparison in encyclopædia descriptions. In *Proceedings of the 8th International Workshop on Natural Language Generation*, 161-170. Sussex, UK.

Milosavljevic, M., Tulloch, A., and Dale, R. (1996). Text generation in a dynamic hypertext environment. In *Proceedings of the Nineteenth Australasian Computer Science Conference*, 417-426. Melbourne, Australia.

Milosavljevic, M. (1997). Content selection in comparison generation. In *Proceedings of the Sixth European Workshop on Natural Language Generation*. Duisburg, Germany.

Moore, J. D. (1989). *A Reactive Approach to Explanation in Expert and Advice-Giving Systems*. Ph.D. Dissertation, University of California, Los Angeles.

Moore, J. D. (1995). *Participating in Explanatory Dialogs: Interpreting and Responding to Questions in Context*. Cambridge, MA: MIT Press.

Paris, C. L. (1987). *The Use of Explicit User Models in Text Generation: Tailoring to a User's Level of Expertise*. Ph.D. Dissertation, Columbia University.

Paris, C. L. (1993). *User Modeling in Text Generation*. London: Pinter Publishers.

Tversky, A. (1977). Features of similarity. *Psychological Review* 84:327-352.

தமிழ்சொல்லுருவாக்கி, கிரந்தநீக்கி, ஆண்ட்ராய்டு அகராதி

Tamil New word creator, Grantha remover, Android Tamil dictionary

பிச்சைமுத்து முத்தையா, Pitchaimuthu Muthiah
Isaiyini Yahoo group,
(https://thanithamizhakarathikalanjiyam.github.io)
pitchaimbox-TIC2019@yahoo.com

Abstract. Tamil language has lot of treasures itself from Sanka-Age to modern years. Tamil language travels tightly coupled with grammar, so that this language is living still from the two thousand years ago. Tamil grammar books deals with letters, letters couplets, word formation, and creations too. This is a try to making Tamil language sanka-age prescriptions to computer age. In this essay we deals mainly about Creating new Tamil words using TTAK software new word creation module, another module is deals with removing Grantha letters using Nannool, Tholkappiyam rules. At last, an android application can able to add new dictionaries, can able to search words from the newly added dictionaries to the mobile phone.

Keywords: Tamil new word generation, Grantha remover, all in one dictionary android application.

1. Introduction

Time is rolling always, language is living with people. People age is limited, but Language age is not limited when the language provides an option to create new words to the recent living generation. In this world lot of languages born, lives with people but some languages only able to live still even if the language was born before thousands of years ago. If we try to list out those languages Tamil is one of them remains are Sanskrit, Greek, Latin, Chinese, Persian and Arabic.

Scope of this research is how to generate new words in Tamil using classic grammar rules with the help of Softwares, here mainly focusing the software is called ThanithamizhakarathiKalanjiyam (TTAK abbreviation in this paper). Tamil is independent language that is not need other language words to expressing things; yes without any language word Tamil can able to create new words independently. Tamil has set of rules to create new words, the set of rules are available in tholkappiyam and nanool also.

Second scope of this research is how to remove the Grantha letters. What is Grantha? The answer is around 10th century in tamil land some scholars knows sankrit also, they try to mingle sanskrit words in to tamil words; named as ManiPirvalaNadai; through finding new set of characters - letters - which are not available in the tholkappiyam age as well as before the sanka age. The set of characters is called Grantha Letters. So there is a need for eradicate those words, through eradicate the Grantha letters from the Tamil words. So the Nannool author pavananthi find out set of rules to eradicate the Grantha letters - from the Tamil people normal life usage. ThanithamizhakarathiKalanjiyam software implemented those rules to remove grantha letters from the Gratha words.

Third scope of this research is making and android application can able to use as a dictionary, as well as a Tamil book plain text reader. The name of the android application is Akga (அஃக), A little bit introduction will be given in this essay.

First we will see about Tamil words formation scenario, then how to use TTAK for Tamil generating new words. Then we will see a set of rules to remove grantha letters from a given grantha mixed word. Then

an introduction is given about how to use the all in one android application for dictionaries, essays, book reader.

2. Tamil Letters

As per tamil literature tamil language is combination of main-letters, and compound letters and one single letter. Main letters are combination of Vowels and consonants.

Tamil Letters		
Vowels (உயிர் - Soul)	அ - ஔ	012 letters
Consonants(மெய் - body)	க் - ன்	018 letters
Compound letters	க - னௌ	216 letters
One Single letter	ஃ	001 letter
Total		247 letters

3. Tamil word formation

Tamil word formation is done by combining the above 247 letters. Where some letters are standing near by. Some letters are not standing near. Some letters stand first of the Tamil words. Some letters can be standing at the end of a word. Let's see what are the letters in a Tamil word.

3.1 First Letters of a Word

The nanool rules 102 - 106 deals about the first letters of a word. The rule is,

(மொழிக்கு முதலில் வரும் எழுத்துக்கள்)
பன்னீ ருயிருங் கசதந பமவய
ஞஙவீ ரைந்துயிர் மெய்யு மொழிமுதல். 102
(வகரம்)
உஊ ஒஓ வலவொடு வம்முதல். 103
(யகரம்)
அஆ உஊ ஓஔ யம்முதல். 104
(ஞகரம்)
அஆ எஒவ்வோ டாகு ஞம்முதல். 105
(ஙகரம்)
சுட்டியா வெகர வினாவழி யவ்வை
யொட்டி ஙவ்வு முதலா கும்மே. 106

தமிழ் சொற்களின் முதல் எழுத்துக்கள்

சொல்லின் முதலெழுத்து												
ஃ	அ	ஆ	இ	ஈ	உ	ஊ	எ	ஏ	ஐ	ஒ	ஓ	ஔ
க்	க	கா	கி	கீ	கு	கூ	கெ	கே	கை	கொ	கோ	கௌ
ங்	ங	ஙா	ஙி	ஙீ	ஙு	ஙூ	ஙெ	ஙே	ஙை	ஙொ	ஙோ	ஙௌ
ச்	ச	சா	சி	சீ	சு	சூ	செ	சே	சை	சொ	சோ	சௌ
ஞ்	ஞ	ஞா	ஞி	ஞீ	ஞு	ஞூ	ஞெ	ஞே	ஞை	ஞொ	ஞோ	ஞௌ
ட்	ட	டா	டி	டீ	டு	டூ	டெ	டே	டை	டொ	டோ	டௌ
ண்	ண	ணா	ணி	ணீ	ணு	ணூ	ணெ	ணே	ணை	ணொ	ணோ	ணௌ
த்	த	தா	தி	தீ	து	தூ	தெ	தே	தை	தொ	தோ	தௌ
ந்	ந	நா	நி	நீ	நு	நூ	நெ	நே	நை	நொ	நோ	நௌ
ப்	ப	பா	பி	பீ	பு	பூ	பெ	பே	பை	பொ	போ	பௌ
ம்	ம	மா	மி	மீ	மு	மூ	மெ	மே	மை	மொ	மோ	மௌ
ய்	ய	யா	யி	யீ	யு	யூ	யெ	யே	யை	யொ	யோ	யௌ
ர்	ர	ரா	ரி	ரீ	ரு	ரூ	ரெ	ரே	ரை	ரொ	ரோ	ரௌ
ல்	ல	லா	லி	லீ	லு	லூ	லெ	லே	லை	லொ	லோ	லௌ
வ்	வ	வா	வி	வீ	வு	வூ	வெ	வே	வை	வொ	வோ	வௌ
ழ்	ழ	ழா	ழி	ழீ	ழு	ழூ	ழெ	ழே	ழை	ழொ	ழோ	ழௌ
ள்	ள	ளா	ளி	ளீ	ளு	ளூ	ளெ	ளே	ளை	ளொ	ளோ	ளௌ
ற்	ற	றா	றி	றீ	று	றூ	றெ	றே	றை	றொ	றோ	றௌ
ன்	ன	னா	னி	னீ	னு	னூ	னெ	னே	னை	னொ	னோ	னௌ

From the above rules we can construct the table below which shows the starting letters of a word.

The colored letters can be the first letter of a word. We can able to take decision given word is a Tamil rooted word or not, through considering the first letter of a word. For example take word ராமர் it is not rooted from tamil origin. Because the letter ரா is not standing at the beginning of a Tamil word. So that Kampar add the extra letter இ front with the word so the ராமர் word becomes இராமர் after applying rules.

3.2 Last Letter of a Word

The nanool rules 107 - 109 deals about the lastletters of a word. The rule is,

(மொழிக்கு இறுதியில் வரும் எழுத்துக்கள்)

ஆவி ஞணநமன யரலவ ழளமெய்

சாயு முகரநா லாறு மீறே. 107

(சிறப்பு விதி)

குற்றுயி ரளபி னீறா மெகர

மெய்யொ டேலாதொந் நவ்வொ டாமௌக்

ககர வகரமோ டாகு மென்ப. 108

(எழுத்தின் முதலும் ஈறும்)

நின்ற நெறியேயுயிர் மெய்முத லீறே. 109

தமிழ் சொற்களின் இறுதி எழுத்துக்கள்

சொல்லின் ஈற்றெழுத்து												
ஃ	அ	ஆ	இ	ஈ	உ	ஊ	எ	ஏ	ஐ	ஒ	ஓ	ஔ
க்	க	கா	கி	கீ	கு	கூ	கெ	கே	கை	கொ	கோ	கௌ
ங்	ங	ஙா	ஙி	ஙீ	ஙு	ஙூ	ஙெ	ஙே	ஙை	ஙொ	ஙோ	ஙௌ
ச்	ச	சா	சி	சீ	சு	சூ	செ	சே	சை	சொ	சோ	சௌ
ஞ்	ஞ	ஞா	ஞி	ஞீ	ஞு	ஞூ	ஞெ	ஞே	ஞை	ஞொ	ஞோ	ஞௌ
ட்	ட	டா	டி	டீ	டு	டூ	டெ	டே	டை	டொ	டோ	டௌ
ண்	ண	ணா	ணி	ணீ	ணு	ணூ	ணெ	ணே	ணை	ணொ	ணோ	ணௌ
த்	த	தா	தி	தீ	து	தூ	தெ	தே	தை	தொ	தோ	தௌ
ந்	ந	நா	நி	நீ	நு	நூ	நெ	நே	நை	நொ	நோ	நௌ
ப்	ப	பா	பி	பீ	பு	பூ	பெ	பே	பை	பொ	போ	பௌ
ம்	ம	மா	மி	மீ	மு	மூ	மெ	மே	மை	மொ	மோ	மௌ
ய்	ய	யா	யி	யீ	யு	யூ	யெ	யே	யை	யொ	யோ	யௌ
ர்	ர	ரா	ரி	ரீ	ரு	ரூ	ரெ	ரே	ரை	ரொ	ரோ	ரௌ
ல்	ல	லா	லி	லீ	லு	லூ	லெ	லே	லை	லொ	லோ	லௌ
வ்	வ	வா	வி	வீ	வு	வூ	வெ	வே	வை	வொ	வோ	வௌ
ழ்	ழ	ழா	ழி	ழீ	ழு	ழூ	ழெ	ழே	ழை	ழொ	ழோ	ழௌ
ள்	ள	ளா	ளி	ளீ	ளு	ளூ	ளெ	ளே	ளை	ளொ	ளோ	ளௌ
ற்	ற	றா	றி	றீ	று	றூ	றெ	றே	றை	றொ	றோ	றௌ
ன்	ன	னா	னி	னீ	னு	னூ	னெ	னே	னை	னொ	னோ	னௌ

From the above rules we can construct the table below which shows the ending letters of a word.
The colored letters can be ending letters of a tamil word. If a word not ending with the given letters means that word root is not in tamil origin. For example take enlish month *மார்ச்* this word ending with the letter *ச்*, as per the tamil grammar rule a word not ending with the letter *ச்*. So that word is not tamil origin. If we try to make the word *மார்ச்* to tamil grammar accepted format means we can add *சு* at the end of the word (ie. *மார்ச்சு*)

3.3 In Between Letters Of A Word

From above tables (nanool rules) we can able to construct the first and last letter of a tamil word. Now we consider the between letters of a tamil word. The nanool rules 110 - 119 give the nearby letters which are present in the between letters of first and last letters of a word. For the simplicity of the paper between letters rules skipped and more details available at page https://thanithamizhakarathikalanjiyam.github.io/tamil_idaieluthukkal. And those details are explained at the presentation. And the TTAK software module usage is given at annexure A, end of this paper.

Hope above details help to understand the word construction of tamil language. Next we will see about how to remove the non tamil letters from the tamil words. THe non tamil letters called grantha, nanool shows rules about how to remove the Grantha letter based word in to pura tamil letters.

4. Grantha Letters

ஜ, ஷ, ஸ, ஹ ,க்ஷ, ஶ்ரீ those letters are called grantha letters, those letters are mingled with tamil words, with the help of ManiPravalaNadai - scholars who live in tamil land introduce the new approach to language. So book authors like nanool - pavanathiyar - find out set of rules which eradicate the letters from the tamil words.

Nanool rules from 146 to 150 defines the removing grantha methods from a grantha mingled letters. In TTAK KiranthaNekki module is used to eliminate the grantha letters from a word. The software module details are given in Annexure B.

4.1 Algorithm for Grantha Remover

Using the following algorithm we can able to remove the grantha letters from a word. Hope the websites are implementing this algorithm to remove the grantha letters from the Tamil computing word.

1. Get user input word. Contains letters [l1,l2,..,ln-1,ln]

2. Assign WordLen = length([l1,l2,..,ln-1,ln])
3. Assign i = 0, n = 0
4. FOR EACH letters in [l1,l2,..,ln-1,ln]
    - 4.1. n = i
    - 4.2. IF ln == Kha or ga or Gha then
        - 4.2.1. ln = க
    - 4.3. ELSE IF ln == tha or da or dha then
        - 4.3.1. ln = ட
    - 4.4. ELSE IF ln == ttha or ddha or dhaa then
        - 4.4.1. ln = த
    - 4.5. ELSE IF ln == pha or ba or bha then
        - 4.5.1. ln = ப
    - 4.6. ELSE IF ln == Chha Then
        - 4.6.1. ln = ச
    - 4.7. ELSE IF ln == Ja Then
        - 4.7.1. ln = ய
    - 4.8. ELSE IF ln == Jha Then
        - 4.8.1. ln = ச
    - 4.9. ELSE IF ln == ஸ Then
        - 4.9.1. ln = ச
    - 4.10. else Calculate n position
        - 4.10.1. if n is 0 then
            - 4.10.1.1. if ln == Sa Then
                - 4.10.1.1.1. ln = ச
            - 4.10.1.2. ELSE IF ln == ஷ Then
                - 4.10.1.2.1. ln = ச
            - 4.10.1.3. ELSE IF ln == ஹ Then
                - 4.10.1.3.1. ln = அ
        - 4.10.2. ELSE IF n is > 0 and < n then
            - 4.10.2.1. if ln == Sa Then
                - 4.10.2.1.1. ln = ச
            - 4.10.2.2. ELSE IF ln == ஷ Then
                - 4.10.2.2.1. ln = ட
            - 4.10.2.3. ELSE IF ln == ஹ Then
                - 4.10.2.3.1. ln = க
        - 4.10.3. ELSE IF n == (WordLen - 1) then
            - 4.10.3.1. if ln == Sa Then
                - 4.10.3.1.1. ln = ச

4.10.3.2. ELSE IF ln == ழ Then

4.10.3.2.1. ln = ட

4.10.3.3. ELSE IF ln == ற Then

4.10.3.3.1. ln = க

4.11. i++

5. END FOR

Android dictionary and essay, book reader:

Also our next generation application for tamil dictionary called Akga (அஃக) which is already available online at https://thanithamizhakarathikalanjiyam.github.io/android. Using this we can able to read plain text books, essays and dictionary too.

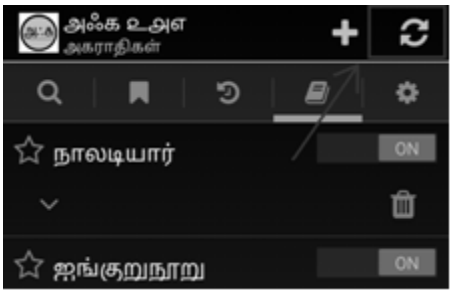
Individual persons making tamil new words to english technical terms, but the dictionary is cant able to use; because they are residing in PDFs, or word documents. If we provide an option for those words available in android application means those words reach vast amount of tamil people. So Akga is an application can able to adopt any number of dictionaries, it can able to search those words in different dictionary.

We already distribute those dictionary through https://thanithamizhakarathikalanjiyam.github.io/tag_/அகராதிகள் In paticularly we should give credits to the authors of dictionary namely,

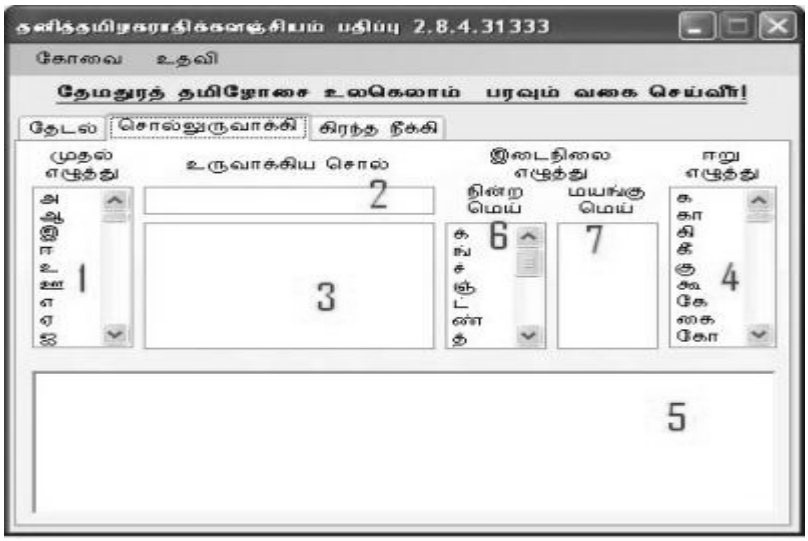
- Automic based dictionary by Jayabharathan, Canda
- Medical terms dictionary by KanmaniGaneshan and Pugalendhipandiyan
- Hotels food terms by Thiruththam Pon Saravanan

Those dictionaries also can able to access from the Akga android application. Also we distribute certain books from sangam age to Modern age. If an authors provide essays, books means those books also can able to read by this application. The books are distributed at https://thanithamizhakarathikalanjiyam.github.io/more_books . The user just download the book and refresh the books list at the android application. User can able to access the book from this android application itself. For example assume users want to download a book from the site means user follows steps are,

1. User download Akga dictionary application from this link அஃக உ_அ - Download for Android
2. User Download the particular distributed book on the books page.
3. Click refresh button on the android application
4. Then search the “NoolName/” at the search box, User can able to view the book Contents page. By clicking each link he can able to read a particular chapter of the book.



Annexure A:



Box 1 contains list of letters for a tamil word, If user click a letter on this box list, It will append to the box 2.

Box 2 is the constructing word by the user.

Box 3 is software suggested list of possible combination of the given word, present in the Box 2.

Box 4 is possible last letters of tamil words, If user click on this, the clicked letter will be append to the box 2.

Box 5 is used to displaying currently available word with starting of the letters available in the box2.

Box 6 if the user append consonant letter to the box 2 means, user can click on this box list of letters.

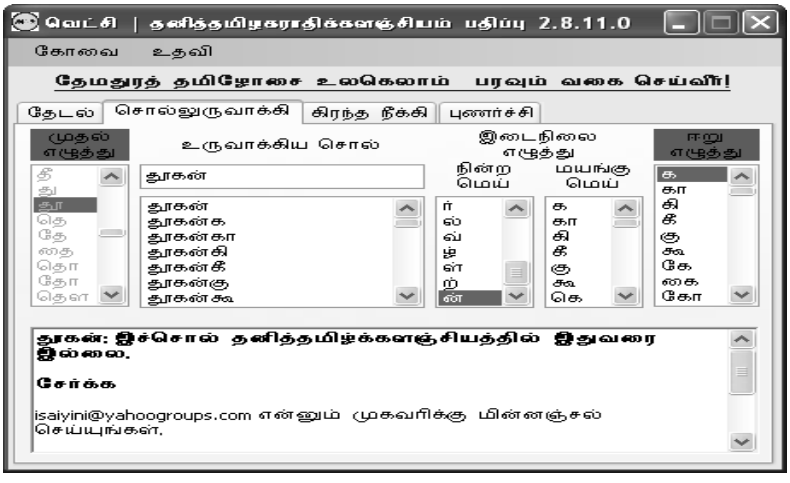
Box 7 show the list of letters can able to come after user click on the box 6 letters.

உருவாக்கிய முதல் புதிய சொல் தூகன்!?

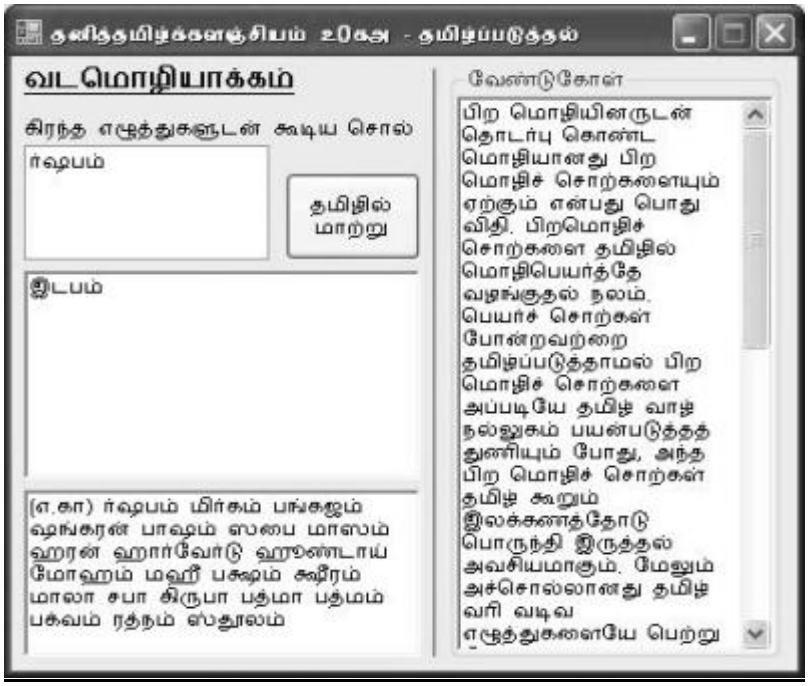
- தூ எனும் முதல் எழுத்தை அழுத்தினேன் (1 - இடத்திலிருந்து).

பல சொற்கள் வந்தன. நான் அடுத்த எழுத்து க இருக்கலாம் என எண்ணினேன் (2 - இடத்திலிருந்து).

- தூகம் என்றால் காற்று என களஞ்சியம் பரிந்துரைக்கிறது. எனவே ஒரு சொல்**காற்றை போல வேகமானவன் என்னும் பொருளில்** ன் என பரிந்துரைக்கும் எழுத்தை 3 - ம் இடத்திலிருந்து தேர்ந்தெடுத்தேன். தூகன் எனும் சொல் உருவானது.



Annexure B:



If user gives the grantha mixed word in the given textbox then click the button, the correspoding tamil word is generated except the grantha letter.

கணினிவழி தமிழ்மொழி கற்றல் கற்பித்தலில் Quizziz என்னும் இணையத்தளத்தின் பயன்பாடு

Magah Letchimi, Karpagam Manickavasagam and Arul Jothy Ravindran
MOE Jurong West Secondary School
Singapore

முன்னுரை

இன்றைய 21-ஆம் நூற்றாண்டில் தகவல் தொடர்பு தொழில் நுட்பத்தின் துரித வளர்ச்சி கல்வித் துறையில் மிகப் பெரும் மாற்றத்தை ஏற்படுத்தி உள்ளது. பாலர் பள்ளி முதல் உயர்கல்வி கழகங்கள் வரை கணினி மற்றும் இணையத்தின் ஆதிக்கமே கையோங்கி இருக்கிறது. அறிவுப் பெருக்கத்திற்குத் துணைபுரியும் இணையம் சார்ந்த மென்பொருள்களும் நவீன ஊடகக் கருவிகளின் பயன்பாடும் அதிகரித்துவிட்டன. நாளொரு மேனியும் பொழுதொரு வண்ணமும் புதுப்புது கருவிகளும் அவற்றில் இயங்கும் செயலிகளும் வந்தவாறு உள்ளன. இத்தகைய சூழலில் வகுப்பறைக் கற்றல் கற்பித்தலிலும் தகவல் தொழில் நுட்பத்தின் பயன்பாடு இன்றியமையாததாகிவிட்டது.

சிங்கப்பூரில் தமிழ்க் கற்றல் கற்பித்தல்

சிங்கப்பூர் பள்ளிகளில் மொழியின் அடிப்படைத் திறன்களான கேட்டல், பேசுதல், படித்தல், எழுதுதல் ஆகியவற்றோடு மாணவர்களிடையே இருவழிக் கருத்துப் பரிமாற்றத் திறன்களான பேச்சுவழிக் கருத்துப்பரிமாற்றமும் எழுத்துவழிக் கருத்துப்பரிமாற்றமும் வளர்க்கும் கற்பித்தல் முறைகள் பின்பற்றப்பட்டு வருகின்றன. மெவாழ்க்கைச் சவால்களைச் சந்திப்பதற்கும் எதிர்கொள்வதற்கும் 21-ஆம் நூற்றாண்டுத் திறன்களுக்கும் முக்கியத்துவம் அளிக்கப்படுகின்றன. இத்திறன்களைத் தனித்துக் கற்பிக்காமல் மொழிப் பாட நடவடிக்கைகளோடு இணைத்துக் கற்பிப்பதற்குத் தகவல் தொடர்பு தொழில்நுட்பம் துணைபுரிகிறது.

"தமிழ் வழங்கும் வேறு எந்த நாட்டையும் விடச் சிறந்த துணைக் கருவிகளையும், நவீன சாதனங்களையும் பயன்படுத்தித் தமிழ் கற்பிக்கக்கூடிய வாய்ப்பு வசதிகள் சிங்கப்பூரில் மிகுதியாக உள்ளன" (கா. இராணமயா, ப. 101) சிங்கப்பூர்ச் சூழலுக்கும் வளர்ந்துவரும் தொழில்நுட்பத்திற்கும் ஏற்ப மாணவர்களை ஆக்கபூர்வமான மொழி நடவடிக்கைகளில் ஈடுபடுத்தும் வகையில் தொழில்நுட்ப வசதிகள் கைக்கொடுக்கின்றன. அந்த வகையில், ஒவ்வொரு வகுப்பறையிலும் நவீன தொழில்நுட்ப கருவிகளும் பல்லூடக வளங்களும் இணைக்கப்பட்டு அவற்றின்வழி சிங்கப்பூரில் தமிழ்மொழி கற்றல் கற்பித்தல் நிகழ்கிறது.

Quizziz website –Implementation

The integration and exploitation of technology in in recent years takes place through pleasant environments of learning, collaboration and authenticity. Such an environment is the Quizizz. It requires the student's active engagement and the emergence of a learning experience that no longer relies on the sterile knowledge of the content.

Quizziz website for Virtual Google Classroom teaching & learning of Tamil Language - தமிழ்மொழி கற்றல் கற்பித்தலில் 'குவிசிஸ் இணையத்தளம் – மெய்நிகர் கூகல் வகுப்பறைக் கற்றல் கற்பித்தல்

Quizziz என்னும் இணையத்தளம் மிகவும் எளிமையாகப் பயன்படுத்தக்கூடிய விளையாட்டுத் தளமாக அமைகிறது. மாணவர்கள் தங்களின் சுயக் கற்றலுக்கும் தங்களின் கற்றலை மதிப்பீடு செய்வதற்கும் இக்கற்றல் தளம் பெரிதும் துணைப்புரிகறது. இத்தளத்தில் மாணவர்கள் விளையாட்டு முறையில் தங்களின் கற்றலுக்குப் பொறுப்பு ஏற்கின்றனர். 'குவிஸ்' (Quiz) அதாவது புதிர் முறையில் மாணவர்கள் கேள்விகளுக்குப் பதில் வழங்கும்போது தங்களின் கற்றல் அடைவு நிலைகளை விளையாட்டு முறையில் கண்டறியலாம். தாங்கள் செய்யும் பிழைகளைத் திருத்திக்கொளவதற்கும் கேள்விகளை மீள்பார்வை செய்வதற்கும் Quizziz வழிசெய்கின்றது, இத்தளம் விளையாட்டு முறையில் அமைக்கப்பெற்றதால் மாணவர்கள் தாங்கள் புதிரின் இறுதியில் பெறக்கூடிய புள்ளி விவரங்களை (Leader board) வாயிலாகவும் பார்க்கலாம். இது இந்த விளையாட்டு இணையத்தளத்தின் ஒரு முக்கிய கூறு ஆகும்.

Host Game Feature (நேரடி இணையத்தள விளையாட்டு)

அவர்கள் தங்களுக்கு வழங்கப்பட்டிருக்கும் புதிர் விளையாட்டைத் தங்களின் சக மாணவர்களுடன் ஒரே நேரத்தில் நேரடி இணையத்தள விளையாட்டாக (Live Host Game) விளையாடும் வாய்ப்பு குவிசிஸ்யில் உள்ளதால் மாணவர்கள் பெரிதும் கற்றல் கற்பித்தலில் ஈர்க்கப்படுகின்றனர்.

பல வசதிகளைக் கொண்ட Quizziz இணையத்தளம்

குவிசிஸ் இணையப்பக்கம் Smart (திறன்) சாதனங்களிலும் கணினிகளிலும் (Microsoft & iMac) இயங்கவல்லது. தேவையான வளங்களை மிக இலகுவாகவும் குவிசிஸ்-இல் உருவாக்கி நிர்வகிக்க முடியும். வளங்கள் என்று கூறும்போது, புதிருக்கான கேள்விகளைத் தயாரிப்பது ஆகும். குவிசிஸ் இணையப்பக்கம் ஆங்கிலம் மற்றும் தமிழ் மொழிகளை ஆதரிக்கும் வசதியைக் கொண்டது. (multi lingual function) Windows, Mac, iPad, iPhone, Android போன்ற தொழில்நுட்ப சாதனங்களின் வழியாகவும் ஆசிரியரால் தயாரித்து வெளியீடு செய்யப்பட்ட விளையாட்டுப் புதிர்கள் மாணவர்களுக்கு ஒப்படைப்பாக வழங்கலாம். குவிசிஸ்-இல் பெறக்கூடிய சில சிறப்பு வசதிகளை இப்போது காணலாம்.

1. *எளிமை சார்ந்த விளையாட்டுத் தளத்தில் கற்றல் மதிப்பீடு* பாடங்களை மீள்பார்வை செய்வதற்குப் பயன்படுதல், விடுமுறை வீட்டுப்பாடமாகவும் வகுப்பறை கற்றல் கற்பித்தலுக்கும் பயன்படுத்தல்.

2.மற்ற ஆசிரியர்கள் தயாரித்த புதிர்களைப் பார்வையிடுதல்,வளங்களின் பெருக்கம்

எளிதாக ஆசிரியர்கள் தங்களுக்கென்று ஒரு வினா வங்கியைத் தொகுக்க முடியும். இது நிச்சயமாக வளங்களை அதிகரிக்கின்றது, தமிழ் ஆசிரியர்களும் இத்தளத்தைப்

பயன்படுத்தி வருவது குறிப்பிடத்தக்கது. இந்த வசதி, நிச்சயமாக நேரத்தை மிச்சப்படுத்துவதற்குக் கைக்கொடுக்கின்றது.

3.மாணவர்கள் Leader board வசதி மற்றும் விடைகளைக் காண்பித்தல் விளையாட்டு என்று வந்தாலே மாணவர்களுக்குக் கற்றல் நடவடிக்கையில் ஒரு தனி ஆர்வம் கூடுகின்றது. விளையாட்டில் மற்ற மாணவர்களின் செயல்பாடு இந்த Leader board வசதி வாயிலாக அறியலாம். இந்த வசதியில் ஒவ்வொரு மாணவரும் விளையாட்டு என்ற அமைப்பு முறையில் தங்களுக்கென்று ஒரு தன்த்துவம் வாய்ந்த பெயர் ஒன்றைப் பதிவு செய்வதோடு தங்கள் கற்றல் கற்பித்தலை மதிப்பீடு செய்வதற்கு post game வழி விடைகளை மீள் பார்வை செய்யலாம்.

மேலும், குவிசிஸ் gamification அமைப்பு முறையில் இயங்குகிறது. புதிருக்கான கேள்விகளைத் தயாரிக்கும் எண்ணிக்கைக்கு அளவில்லை.

4.Memes பயன்பாடு –Trendy feature for students to give feedback after each question

Quizziz -இல் பல்வேறு கோப்பு வடிவங்களை இணைக்கும் வசதி உண்டு.குறிப்பாக மாணவர்களின் செயல்பாட்டுக்கு மதிப்பீடு வழங்குவதற்கு Memes பயன்படுத்தலாம். இந்த வசதி மாணவர்களைப் பெரிதும் கவரக்கூடிய வசதியாக அமைகின்றது. மாணவர்கள் சிறப்பாகச் செய்தால் அதனை Memes-இல் தயார் செய்து மாணவர்களுக்கு வழங்கலாம்.

5.Quizziz செயலியாகத் திறன் பேசிகளில் இயங்கும் வசதி

(Quizziz online & mobile app)

Quizziz-ஐ எங்கு வேண்டுமானாலும் பயன்படுத்தலாம். கணினிகளில் மட்டுமல்லாது திறன் பேசிகளிலும் இணையத்திலும் Quizziz இயங்கும்.

என்ற இணையப்பக்கத்தின்வழி நாம் எளிதாகக் விளையாட்டுப் புதிர்களைத் தயார் செய்து மாணவர்களின் செயற்பாட்டை Reports (தரவுகள்) வழியாகப் பார்வைஇட முடியும். திறன் பேசியிலும் இலவசமாகக் குவிசிஸ் செயலியைத் தரவிறக்கம் செய்து பயன்படுத்தலாம். குவிசிஸ் இணையப்பக்கம் மற்றம் செயலியை எல்லா துறைகளுக்கும் (எ-டு விளையாட்டு, தொழில், கல்வி) எளிமையாகப் படுத்தலாம்.

நோக்கங்கள்

இதுவரை குவிசிஸ்-இல் உள்ள சிறப்பு அம்சங்களைப் பார்த்தோம். மாணவர்கள் ஆக்கபூர்வமான மொழி நடவடிக்கைகளில் ஈடுபட்டு மொழியைத் திறம்படக் கற்க வேண்டும். அவர்கள் சுயமுனைப்புடனும் உடனிணைந்தும் செயல்பட்டு மொழியை ஆர்வத்துடன் கற்க வேண்டும். தொழில்நுட்பம் கற்பித்தலுக்குச் சுமையாக இல்லாமல் எளிதாகப் பெற்றுப் பயன்படுத்தக்கூடிய ஒன்றாக இருக்க வேண்டும். இந்த நோக்கங்கங்களைக் கருத்தில் கொண்டு டிஜூடல் குறிப்பேடாக விளங்கும் குவிசிஸ்-ஐக் கற்றல் கற்பித்தலுக்கு எவ்வாறு பயன்படுத்தலாம் என்பதைப்பற்றி சிந்தித்தோம்.

குவிசிஸ்-இல் உள்ள வசதிகளைப் பயன்படுத்தி மாணவர்களுக்கு ஒரு முழுமையான பாடத் தொகுப்பை (lesson package) வழங்கும் சிறு முயற்சியில் ஈடுபட்டோம். பாடத்திற்குரிய நோக்கங்கள், விளக்கங்கள், துணை வளங்கள், திட்ட வேலைகள், வீட்டுபாடங்கள் ஆகியவை தொகுக்கப்பட்டு ஒரு பாடமாகக் குவிசிஸ்-இல் தயாரிக்கப்பட்டது. மதிப்பீட்டிற்கு வழிசெய்யும்

வகையிலும் பாடம் அமைக்கப்பட்டது. மாணவர்கள் விடுமுறை வீட்டுப்பாடமாகத் தங்களின் சுய கற்றல் நடவடிக்கையை குவிசிஸ்-இல் புதிர் நடவடிக்கையாக மேற்கொண்டனர். இப்புதிர் மாணவர்களின் கற்றல் நடவடிக்கையில் ஒரு மதிப்பீடாக இடம்பெற்றது. புதிர் தெரிவு விடைகளின் அமைப்பு முறையில் வழங்கப்பட்டது. ஒரு குறிப்பிட்ட பாடத் தலைப்பு (எ-டு மரபுத்தொடர்கள்) ஒட்டி புதிர் தயாரிக்கப்பட்டு மாணவர்களுக்கு assign செய்யப்பட்டது. புதிரை வழங்கும்போது காலவரையறையையும் நிர்ணயம் செய்யலாம்.

Google classroom –Newly added feature

Google Classroom

Quizziz game platform was assigned as holiday homework to students using the new feature known as Google classroom , where students will be assigned to the game using their google accounts. The accounts will be used to create a class. This google class feature is a newly added feature, whereby the students gmail accounts will be imported in the excel sheet to easily create a class. Hence, students do not need to use the game pin which is being generated and can automatically view the quiz upon the link which is being sent to their gmail account. This feature defintely helps teachers easily create a virtual learning platform. Teachers are also able to create more than one class and add in parents gmail accounts as well, if necessary.

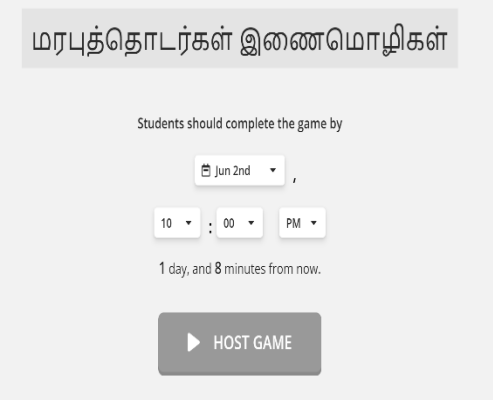


Figure 1 (Setting of deadline)

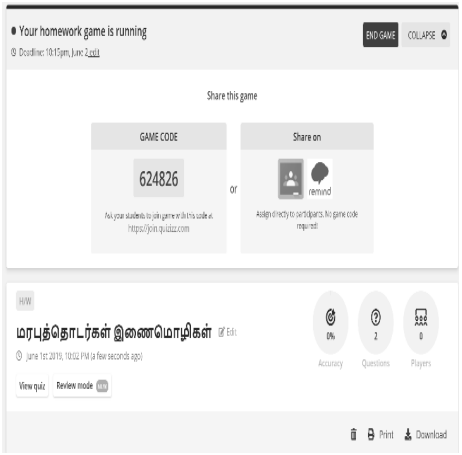
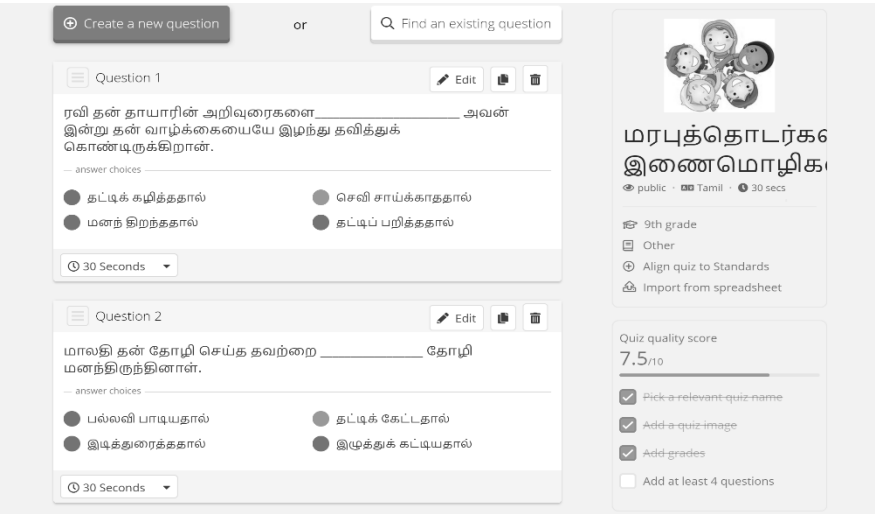


Figure 2 (Assign game using Google classroom)

கற்றல் கற்பித்தலில் குவிசிஸ் பயன்பாடு

கற்றல் நடவடிக்கைகளை குவிசிஸ்-இல் உருவாக்குவதற்கு முன் நன்கு திட்டமிட வேண்டும். கருப்பொருள், துணைக்கருப்பொருள், மாணவர்களிடையே வளர்க்க விரும்பும் மொழித் திறன்கள், பாட நோக்கங்கள் போன்ற சில அடிப்படை கூறுகளை முதலில் முடிவு செய்ய வேண்டும். பின் கேள்விகளைப் புதிர் விளையாட்டு அமைப்பு முறையில் தயாரிக்க வேண்டும். மற்றவர்கள் இந்தத் தளத்தில் வளங்களாகப் பகிர்ந்துகொண்ட கேள்விகளைப் பார்வையிட்டும் பிரதி எடுத்தும் கூட புதிரைத் தயாரிக்கலாம்.

எடுத்துக்காட்டு



இவ்வாறு குவிசிஸ்-இல் பல விளையாட்டுப் புதிர்களைச் சுலபமான முறையில் தயாரிக்கலாம்.

குவிசிஸ்-இல் தயாரித்த புதிரைப் பகிர்ந்து கொள்ளுதல்

ஆசிரியர் உருவாக்கிய புதிர் கேள்விகளை அடுத்தப் படிநிலையாக மாணவர்களுடன் பகிர்ந்து கொள்ளுதல் அவசியமாகும். இதனை மூன்று வழிகளில் மேற்கொள்ளலாம். முதலில் Live Game , இரண்டாவதாக Homework (வீட்டுப்பாடம்),மூன்றாவதாக (Solo Game) தனி நிலையில் பகிர்ந்து கொள்ளலாம்.

1. மாணவர்கள் account உருவாக்கத் தேவையில்லை join.quizziz.com என்ற இணையப்பக்கத்திற்குச் சென்று நேரடியாகவே புதிரை Game pin என்ற எண்ணைக் கொண்டு புதிர் விளையாட்டை நேரடியாக வகுப்பறையில் மேற்கொள்ளலாம்.

2. Homework, அதாவது வீட்டுப்பாட வசதிக்கு ஆசிரியர் virtual classroom ஒன்றினை அமைத்தல் அவசியம். மாணவர்கள் விளையாட்டுப் புதிரை வீட்டுப்பாடமாக மேற்கொள்வதற்கு மாணவர்களின் தொகுக்கப்பட்ட மின்னஞ்சல் முகவரிகள் தேவைப்படும். மாணவர்களின் google gmail account வைத்து Google classroom அல்லது google அல்லாத மற்ற மின்னஞ்சல் முகவரியை வைத்து new classroom

ஒன்றினையும் ஆசிரியர அமைக்கலாம். பின் உருவாக்கப்பட்ட வகுப்பறை வாயிலாக மாணவர்களுக்கு எளிதாகப் புதிர் விளையாட்டை assign செய்யலாம்.

மாணவர்கள் வீட்டில் இருந்தபடியே வசதியாகத் தங்களின் கணினி, மடிக் கணினி அல்லது திறன்பேசியின் வழி ஆசிரியர் பகிர்ந்துகொண்ட விளையாட்டைப் பார்வையிட்டுப் புதிர் விளையாட்டை நிறைவு செய்யலாம் ஆசிரியர் மாணவரின் செயற்பாட்டை Report,அதாவது தரவுகள் வாயிலாகக் கண்காணிக்கலாம்.இந்தத் தரவுகளை ஆசிரியரோடு பெற்றோரும் கண்காணித்து தங்களின் பிள்ளைகளின் கற்றல் அடைவுகளைப் பார்வையிட்டுக் கருத்து தெரிவிக்கலாம்.

சவால்களும் தீர்வுகளும்

இம்மென்பொருளைப் பயன்படுத்துவது மிகவும் எளிது. ஆதலால் எதிர்கொண்ட பிரச்சினைகள் வெகுசிலவே. அனைத்து மாணவர்களிடமும் திறன் பேசிகள் உள்ளதால் மாணவர்கள் வகுப்பில் இவ்விளையாட்டை மேற்கொள்ளும்போது தங்களின் திறன் பேசிகளைப் பயன்படுத்தினர். விளையாட்டு முறையில் கேள்விகளுக்கான பதில்களைத் தெரிவுகள் அமைப்பு முறையில் கண்டறிந்தனர். விளையாட்டின் முடிவில் மாணவர்கள் தங்கள் சக மாணவர்களுடன் சேர்ந்து பெற்ற புள்ளிகளை ஒப்பிட முடிந்தது. இது நிச்சயமாக மகிழ்வூட்டும் கற்றல் கற்பித்தலுக்கு வழிசெய்கின்றது. இருப்பினும் மாணவர்களுக்காக இவ்விளையாட்டு இணையத்தளத்தில் கேள்விகளைத் தெரிவுகள் வடிவமைப்பில் மட்டுமே தயாரிக்க முடியும். மாணவர்கள் தங்களின் விடைகளைத் தட்டச்சு செய்ய இயலாது.

இவ்வாறாக, விளையாட்டு வழியில், கணினி மற்றும் திறன் பேசி வாயிலாக எங்கள் பள்ளியில் உள்ள தமிழ் ஆசிரியர்கள் மட்டுமல்லாது மற்ற மொழி ஆசிரியர்களும் குவிசிஸ் இணையப் பக்கத்தினைக் கற்றல் கற்பித்தலுக்கும் மதிப்பீட்டிற்கும் பயன்படுத்தித் தங்களது கற்றல் கற்பித்தல் நோக்கங்களைப் படிப்படியாக நிறைவேற்றி வருகிறார்கள்.

முடிவுரை

தாய்மொழி வாழும் மொழியாகத் திகழ மாணவர்கள் மொழியைத் திறம்பட பயன்படுத்த வேண்டும். இது மாணவர்கள் தமிழ் மொழியை ஆர்வத்துடன் கற்றாலொழிய சாத்தியமில்லை. மொழியின் மீது ஆர்வத்தை வளர்க்க குவிசிஸ் இணையப் பக்கத்தினைப் கணினித் துணைக்கருவியாகப் பயன்படுத்துகின்றனர், அந்த வகையில் குவிசிஸ் ஓர் அருமையான தகவல் தொழில்நுட்ப துணைக்கருவியாக அமைகிறது எனில் அது வெள்ளிடைமலை. மாணவர்களின் கற்றல் அடைவு நிலைகளைக் கண்டறிந்து மதிப்பிடு செய்வதற்கும், கற்றல் நோக்கங்களை மீள்பார்வை செய்வதற்கும், கூடிக்கற்பதற்கும் சுயமாகக் கற்பதற்குமான வாய்ப்புகளை உருவாக்கித் தரும் குவிசிஸ் இணையத்தளம், தமிழ் கற்றல் கற்பித்தலில் ஒரு மைல்கல் என்பது திண்ணம். குவிசிஸ் இணையத்தளம் ஆகமொத்தத்தில், குவிசிஸ் இணையத்தளத்தின் பயன்பாட்டை ஆசிரியர் மற்றும் மாணவர் நிலையில் மதிப்பீடு செய்தால், 10-த்திற்கு 8 நட்சத்திரங்கள் வழங்கலாம்.

Reference

1. http://educraft.tech/index.php/quizizz
2. https://quizizz.zendesk.com/hc/en-us/articles/115000338045-Getting-Started-with-Quizizz

முறைசாரா மதிப்பீட்டைத் தகவல் தொழில்நுட்பத்துடன் ஒருங்கிணைத்துத் தமிழ்க் கற்றல், கற்பித்தலுக்குப் பயன்படுத்துதல்

செ மதிவாணன் – விக்டோரியா பள்ளி,

மா அர்ச்சுனன் – செயிண்ட் பேட்ரிக்ஸ் பள்ளி, சிங்கப்பூர்

Abstract. சிங்கப்பூர்ப் பள்ளிகளில் தமிழ்மொழி இரண்டாம் மொழியாகக் கற்பிக்கப்படுகிறது. பல்லின சமூகப் பின்புலத்திலிருந்து வரும் மாணவர்கள் தமிழ்மொழியை வகுப்பறைகளில் மட்டுமே பயன்படுத்துகின்றனர். பெரும்பாலான மாணவர்கள் வீட்டுச் சூழலில் தங்கள் தாய்மொழியான தமிழைப் பேசுவதில்லை. இதற்குப் பல்வேறு சூழல்கள் காரணங்களாக இருக்கின்றன. இம்மாணவர்களுக்கு மொழிப்பாடத்தைக் கற்பிக்கும் தமிழாசிரியர்கள் அவர்களின் மொழித் திறன்களை (கேட்டல், பேசுதல், வாசித்தல், எழுதுதல்) மேம்படுத்துவதுடன் தமிழ்மொழியின்பால் ஆர்வத்தை ஏற்படுத்தும் வகையில் கற்பித்தல் நடவடிக்கைகளை மேற்கொண்டு வருகிறார்கள். ஆசிரியர்கள் இத்தகையக் கற்றல் கற்பித்தல் நடவடிக்கைகளை மேற்கொள்வதற்குத் தகவல் தொழில்நுட்ப வளங்கள் பேருதவி புரிகின்றன. அத்தகைய வளங்களுள் ஒன்றான காஹூட் (Kahoot – Game based learning tool) என்ற விளையாட்டு அடிப்படையிலான கற்றல் தளம் முறைசாரா மதிப்பீட்டு முறைகளை செயல்படுத்தும் வண்ணம் வடிவமைக்கப்பட்டுள்ளது. இதனை எங்கள் பள்ளிகளில் தமிழ்மொழிக் கற்றல் கற்பித்தலுக்காக எப்படிப் பயன்படுத்தினோம் என்பதைக் குறித்த பகிர்வாக இக்கட்டுரை அமைகிறது.

முறைசாரா மதிப்பீடு

“முறைசார்ந்த மதிப்பீடு (Summative assessment) என்பது கற்றலை மதிப்பிடுதலாகும். முறைசாரா மதிப்பீடு (Formative assessment) என்பது கற்றலுக்கான மதிப்பீடாகும். இவ்விரண்டு மதிப்பீட்டு முறைகளையும் சமச்சீராகப் பயன்படுத்தும்போது மதிப்பீடு அர்த்தமுள்ளதாக அமைகிறது. சிங்கப்பூர் உயர்நிலைப்பள்ளிகளில் முறைசாரா, முறைசார்ந்த மதிப்பீட்டுகளின் வழி வாசித்தல், கேட்டல், பேசுதல், எழுதுதல் திறன்களை மேம்படுத்த உதவும் வகையில் மதிப்பீடு அமைகிறது”[1]. “கற்றல் கற்பித்தலில் ஒரு முக்கிய கூறாக விளங்குவது மதிப்பீடு ஆகும். அதில் முறைசாரா மதிப்பீட்டின் பங்கு முக்கிய இடத்தைப் பெறுகிறது. முறைசாரா மதிப்பீடு வகுப்பறையில் எல்லா நேரங்களிலும் நடைபெறுவதோடு, கற்றல் கற்பித்தலோடும் ஒன்றிப் போய்விடுகிறது. முறைசாரா மதிப்பீடு, கற்றுலுக்குத் துணைபுரிவதோடு மாணவர்களின் கற்றல் வளர்ச்சி, கற்றலைப் பற்றிய விழுமியங்கள், சாதகமான மனப்போக்கு முதலியவற்றிற்கும் வித்திடுகிறது”[2]. சிங்கப்பூர் மாணவர்களிடையே தமிழ்மொழிக் கற்றல் கற்பித்தலில் சாதகமான மனப்போக்கை உருவாக்குவதற்கு இந்த முறைசாரா மதிப்பீடு பெரிதும் உதவுகிறது.

காஹூட் - Kahoot (https://kahoot.com)

“தொழில்நுட்பத்தின் உதவியோடு விளையாட்டு அடிப்படையிலான கற்றல், கற்பித்தல் முறைகளைப் பயன்படுத்துவது தொடர்பான ஆர்வநிலை இன்று கல்வியாளர்களிடையே மிகவும் செல்வாக்கு

பெற்று வருகிறது. கல்வி சார்ந்த தொழில்நுட்ப விளையாட்டுகள் வகுப்பறைப் பாடங்களில் மாணவர்களின் ஆர்வநிலையை அதிகரிப்பதோடு அதனைத் தக்கவைக்கவும் உதவுகிறது. இணையத்தில் இன்று கல்வியாளர்களுக்காகவும் மாணவர்களுக்காகவும் பலதரப்பட்ட விளையாட்டு வழிக் கற்றல் தளங்கள் உள்ளன. அவற்றுள் ஒன்றான காஹூட் (Kahoot), வகுப்பறைக் கற்றல் கற்பித்தலில் பொருத்தமாகப் பயன்படுத்தப்படும்போது அது மாணவர்களின் ஆர்வநிலையை அதிகரிப்பதோடு அவர்களது கற்றலையும் மேம்படுத்துகிறது"[3]. இதிலுள்ள கற்றல் விளையாட்டுகள் 'காஹூட்ஸ்' (Kahoots) எனப்படுகின்றன. காஹூட் தளத்தின் வடிவமைப்பு முறைசாரா மதிப்பீட்டு முறைகளைப் பயன்படுத்துவதற்கு ஏதுவாக அமைக்கப்பட்டிருப்பதால் அது மாணவர்களின் கற்றலைச் சோதிப்பதற்குப் பெரிதும் உதவுகின்றது. மாணவர்கள், மரபார்ந்த வகுப்பறைக் கற்றல் நடவடிக்கைகளிலிருந்து விடுபட்டு உற்சாகத்துடன் கற்றலில் ஈடுபட இத்தளம் வழிவகுக்கிறது.

"காஹூட் தளத்தின் வடிவமைப்பு ஒரு சமூகமாக இணைந்து கற்பதற்குத் துணைபுரிவதாக இருக்கிறது. அதாவது, மாணவர்கள் ஒரு குழுவாக ஒன்றிணைந்து கற்பதற்கும் தனிநிலையில் கற்பதற்கும் ஏற்றதாக இதன் வடிவமைப்பு அமைந்துள்ளது. மேலும், இதனுடைய விளையாட்டு அமைப்புமுறை ஆர்வமூட்டும் வகையிலும் பயனீட்டாளர் எளிதாகப் பயன்படுத்தும் வகையிலும் (User friendly) உருவாக்கப்பட்டிருக்கிறது. இந்தக் கற்றல் தளத்தைப் பயன்படுத்துவோர் திரையில் தெரியும் Game Pin மூலம் ஆசிரியர் சோதிக்க விரும்பும் பகுதிக்கானப் பக்கத்தில் ஒன்றிணைவர். ஆசிரியர், தான் கற்பித்த பாடம் தொடர்பாக ஏற்கனவே உருவாக்கி வைத்திருக்கும் வினாக்களைக் காஹூட் தளத்தில் உள்ள Quiz அமைப்பிலோ அல்லது Jumbled அமைப்பிலோ உள்ளீடு செய்து மாணவர்களைச் சோதிக்கலாம். மாணவர்கள் தங்கள் கைத்தொலைபேசிகளின் வழியாக அதற்கான விடைகளை உள்ளீடு செய்வர். சரியான பதில்களுக்கு உடனுக்குடன் புள்ளிகள் வழங்கப்படும். மேலும் மாணவர்கள் தங்கள் விடைகள் சரியானவைதானா என்பதை ஒவ்வொரு வினாவின் முடிவிலும் கொடுக்கப்படும் சரியான விடைகளைக் கொண்டும் அவர்களுக்குக் கிடைக்கும் புள்ளிகள் மூலமும் உடனுக்குடன் அறிவர். ஒவ்வொரு வினாவின் முடிவிலும் மாணவர்கள் எந்த நிலையில் உள்ளனர் என்ற தரவரிசையையும் திரையில் காண்பர்"[4].

முறைசாரா மதிப்பீடும் காஹூட்டும்

"முறைசாரா மதிப்பீடு – கற்றலுக்கான மதிப்பீடு, மாணவர்களின் கற்றலுக்குத் துணைபுரிய வல்லது. இது மாணவர்கள் கற்றல் குறிக்கோள்களை அடைய உதவுகிறது. கற்பித்தலோடு பின்னிப் பிணைந்த ஒரு செயற்பாடாகவும் எல்லா நேரங்களிலும் பயன்படுத்தக் கூடியதாகவும் உள்ளது. இவ்வகை மதிப்பீடு நிகழ்வதற்குக் கற்பவரை மையமாகக் கொண்ட அணுகுமுறை தேவைப்படுகிறது"[1]. காஹூட் – விளையாட்டுவழிக் கற்றல் தளம், இவ்வகை அணுகுமுறையைச் செயல்படுத்துவதற்கு ஏற்ற ஒரு தளமாக விளங்குகிறது.

முறைசாரா மதிப்பீட்டு முறை மாணவர்களுக்கு அச்சத்தை ஏற்படுத்தாது என்பதால் மொழிப்பாடம் கற்றலில் முறைசாரா மதிப்பீட்டு முறைகளை எங்கள் தமிழ் வகுப்புகளில் பின்பற்றுகிறோம். இன்றைய மாணவர்கள் தகவல்தொழில்நுட்ப வளங்களுடன் தொடர்புபடுத்தப்பட்ட பாடங்களை ஆர்வத்துடன் கற்பதால் அவர்களுக்குத் தெரிந்த வழியில் சென்று தெரியாததைக் கற்றுக்கொடுக்க எடுக்கும் முயற்சியாகவே இந்த முறையை நாங்கள் அணுகினோம். இதற்கு முறைசாரா மதிப்பீட்டு முறையை உள்ளடக்கிய விளையாட்டு வழிக் கற்றல் தளமான காஹூட் எங்களுக்குப் பெரிதும் கைகொடுத்தது.

அகன்ற வாசிப்பு (Extensive Reading) – மாணவர்களை ஊக்கப்படுத்த உதவிய காஹூட்:

இன்றைய நவீன தொழில்நுட்பம் சார்ந்த உலகில் வாசிப்புப்பழக்கம் பொதுவாக அனைத்துத் தரப்பினரிடமும் குறைந்து வருகிறது என்பது வருத்தத்திற்குரிய ஒரு செய்தியாகும். இப்படியிருக்க பன்மொழிச்சூழலில் தமிழை இரண்டாம் மொழியாகப் பயிலும் மாணவர்களது வாசிப்புப்பழக்கம் பற்றி நாங்கள் கூறவேண்டிய அவசியம் இல்லை. வாசிப்புப் பழக்கம் மிகவும் முக்கியம்; எனவே, மாணவர்களுக்கு வாசிப்பில் ஆர்வத்தை ஏற்படுத்த முதலில் அவர்களுக்கு ஊக்குவிப்பு மிகவும் அவசியமாக இருந்தது. "ஒரு பாடத்தை விளையாட்டுவழி கற்கும்போது அப்பாடத்தின் மீதான மாணவர்களது ஊக்கநிலை அதிகரிக்கரிக்கிறது"[5]. மேலும், "விளையாட்டுவழிக் கற்றல் ஆசிரியரை மையமாகக் கொண்ட மரபார்ந்த வகுப்பறைச் சூழலில் இருந்து மாணவர்களை மையமாகக் கொண்ட வகுப்பறைச் சூழலாக மாறுகிறது"[6]. இதனைக் கருத்திற்கொண்டு காஹூட் – விளையாட்டுவழிக் கற்றல் தளத்தின்வழி நாங்கள் மாணவர்களிடையே வாசிப்புப்பழக்கத்தை ஊக்குவிப்பதற்கு மேற்கொண்ட முயற்சியினை விளக்குகிறோம்.

சிங்கப்பூர்ப் பள்ளிகளில் உயர்நிலை மாணவர்களுக்கு ஒவ்வொரு பாடத்தின் முடிவிலும் கதைப்பகுதி ஒன்று அகன்றவாசிப்புப் பகுதியாகப் பாடத்திட்டத்தில் கொடுக்கப்பட்டுள்ளது. மாணவர்கள் கதைகளை அவர்களாக வாசிப்பதில்லை; ஆசிரியர்களின் வற்புறுத்தலினாலும் பயிற்சிநூலில் உள்ள பயிற்சியைச் செய்வதற்காகவும் மட்டுமே வாசிப்பர். இப்படி வாசிப்பது அவர்களது ஆர்வநிலையை எவ்விதத்திலும் தூண்டவில்லை என்பதை உணர்ந்தோம். இதை வேறு எப்படிச் செய்யலாம் என்று சிந்தித்தபோது வாசித்தப் பகுதிகளை முறைசாரா மதிப்பீட்டுக்கு உட்படுத்துவது என்று முடிவெடுக்கப்பட்டது. அம்மதிப்பீடு அவர்களது ஆர்வத்தைத் தூண்டும் வண்ணம் அமைவது முக்கியம் என்பதும் கருத்தில் கொள்ளப்பட்டது.

உயர்நிலை 1 மற்றும் 2 ஆகிய வகுப்புகளில் உயர்தமிழ் பயிலும் மாணவர்களை இத்திட்டத்தில் ஈடுபடுத்தினோம். ஜூன் மாத விடுமுறையில் மாணவர்கள் "வாழும் மொழி வாழும் மரபு" என்ற நூலைப் படித்துவருமாறு அறிவுறுத்தினோம். பள்ளி திறந்ததும் அதில் வினாடிவினா நடத்தப்படும் என்றும் முதல் மூன்று நிலையில் வருபவர்களுக்குப் பரிசுகள் வழங்கப்படும் என்று மாணவர்களை ஊக்குவிக்கும் விதமாக அறிவித்தோம். இவ்வினாடிவினாவை விளையாட்டுவழிக் கற்றல் தளமாக காஹூட் வழி நடத்துவது என தீர்மானிக்கப்பட்டு, அதற்கான வினாக்களை உருவாக்கினோம்.

ஜூன் மாத விடுமுறை முடிந்து வந்த மாணவர்கள் இந்த முறைசாரா மதிப்பீடு – வினாடி வினா அங்கத்தில் மிகுந்த ஆர்வத்துடன் பங்கெடுத்தனர். முதல் மூன்று நிலையில் யார் வருவது என்பதில் மிகுந்த போட்டி இருந்ததை உணர முடிந்தது. மாணவர்களிடையே இந்த ஆர்வத்தை ஏற்படுத்துவதற்கென நாங்கள் தேர்ந்தெடுத்த காஹூட் வினாடி வினா அங்கம் மாணவர்களிடையே பெரும் வரவேற்பைப் பெற்றது.

இதனைத் தொடர்ந்து பாடத்திட்டத்தில் கொடுக்கப்பட்டுள்ள அகன்ற வாசிப்புப் பகுதியை இம்முறையில் மாணவர்களை வாசிக்கச் செய்து வருகிறோம். இதிலும் மாணவர்களுக்குப் புறம்சார்ந்த

ஊக்குவிப்பு தேவைப்பட்டது. ஒரு பயிற்சி நூலில் 5 முதல் 6 கதைகள் இருக்கும். ஒவ்வொரு வாரமும் ஒரு கதையை மாணவர்கள் படித்து வருவர். அவ்வாரத்தில் வரும் முதல் தமிழ் வகுப்பில் ஆசிரியர் அவ்வாரக் கதைக்கான வினாடிவினாவை காஹூட் வழி நடத்தி முதல் மூன்று நிலையில் வரும் மாணவர்களின் பெயர்களைக் குறித்துக்கொள்வார். இப்படியே அப்பயிற்சிநூலில் உள்ள கதைகள் அனைத்தையும் மாணவர்கள் படித்து முடிப்பர்; அக்கதைகளை மாணவர்கள் புரிந்துகொண்டு வாசித்தார்களா என்பதை அறிய காஹூட் வழி நடத்தப்படும் வினாடிவினாவிலும் ஈடுபடுவர். எல்லாக் கதைகளிலும் சிறப்பாகச் செய்த மாணவர்களின் பெயர்களை ஆசிரியர்கள் குறித்து வைத்திருப்பார். ஒவ்வொரு பருவத்தின் இறுதியிலும் முதல் மூன்று நிலையில் வந்த மாணவர்களுக்குப் பரிசுகள் வழங்கப்பட்டு ஊக்குவிக்கப்பட்டனர்.

இவ்வாறு மாணவர்கள் அகன்ற வாசிப்புப் பகுதியில் வாசித்த கதைகளை முறைசாரா மதிப்பீட்டுக்கு உட்படுத்தி அவற்றைத் தகவல் தொழில்நுட்ப கருவிகளின் வாயிலாகச் சோதித்தால் மாணவர்களிடம் முதலில் புறம்சார்ந்த ஊக்குவிப்பையும் (Extrinsic motivation) பிறகு அகம்சார்ந்த ஊக்குவிப்பையும் (Intrinsic motivation) ஏற்படுத்த முடியும் என்ற நோக்கத்துடன் நாங்கள் எங்கள் பள்ளிகளில் இம்முயற்சியைச் செய்து வருகிறோம். எங்களின் அடுத்த கட்ட முயற்சியாக பலதரப்பட்ட வாசிப்புகளை மாணவர்களிடையே அறிமுகப்படுத்தவிருக்கிறோம். தமிழில் வாசிப்புப் பழக்கத்தை எங்கள் மாணவர்கள் கைக்கொண்டு பயன்பெறவேண்டும். அதற்காக நாங்கள் தொடங்கிய புறம்சார்ந்த ஊக்குவிப்பு தொடரும். அது அவர்களிடையே அகம்சார்ந்த ஊக்குவிப்பாக மாறும் என்ற நம்பிக்கையில் செயல்பட்டு வருகிறோம்.

மரபுத்தொடர்கள் – மகிழ்வூட்டும் கற்றலுக்கு (Joyful learning) உதவிய காஹூட்

சிங்கப்பூர்த் தாய்மொழிப் பாடத்திட்ட அமைப்பில் உள்ள கருப்பொருள்களுள் ஒன்று மரபும் பண்பாடும் என்ற கூறாகும். பன்மொழிச்சூழல், பல இனமக்கள் வாழும் சிங்கப்பூரில் மாணவர்கள் தாய்மொழியைக் கற்கும் அதே வேளையில் தத்தம் மரபுகளை அறிந்திருக்கவேண்டும்; தம் மரபின் பெருமைகளையும் பண்பாட்டுக் கூறுகளையும் கற்றுக்கொள்ள வேண்டும் என்ற நோக்கத்தில் இக்கருப்பொருள் பாடத்திட்டத்தில் இடம்பெற்றுள்ளது. இதனைக் கருத்தில் கொண்டே தமிழ்மொழிப் பாடத்திட்டத்தில் மரபுத்தொடர்கள், இணைமொழிகள் இணைக்கப்பட்டுள்ளன.

தாய்மொழிப் புழக்கமே இல்லாத சூழ்நிலையில் இருந்து வரும் மாணவர்களுக்கு மொழியைக் கற்றுக்கொடுப்பதே சவலான காரியம்; அதிலும், மரபுத்தொடர்களைக் கற்றுக்கொடுத்து அவர்களுக்குப் புரிய வைப்பது என்பது இன்னும் சவாலான காரியம் ஆகும். இதைக் கருத்தில் கொண்டு மரபுத்தொடர்களைக் கற்றுக்கொடுக்க ஆர்வமூட்டும் வகையில் நடவடிக்கைகளை அமைப்பது அவசியமாகப்பட்டது.

இங்குள்ள மாணவர்கள் தமிழ்மொழியைப் பேசுவதில் தயக்கம் காட்டினாலும் தமிழ்த்திரைப்படங்களில் வரும் நகைச்சுவைக் காட்சிகளை விரும்பிப் பார்க்கின்றனர். இதை உணர்ந்த நாங்கள் மரபுத்தொடர்களைக் கற்றுக்கொடுக்க நகைச்சுவைக் காட்சிகளைப் பயன்படுத்தலாம் என முடிவெடுத்தோம். அதனைக் காஹூட் – விளையாட்டு வழிக் கற்றல் தளத்தின் துணைகொண்டும் முறைசாரா மதிப்பீட்டையும் ஒருங்கிணைத்துக் கற்பித்தால் அது மாணவர்களுக்கு மகிழ்வூட்டும் கற்றலாக அமையும் என்பதால் இம்முயற்சியை மேற்கொண்டோம்.

காஹூட்டின் வழி மரபுத்தொடர்களைக் கற்பிக்க உயர்நிலை ஒன்றாம் வகுப்பில் படிக்கும் மாணவர்கள் தேர்வு செய்யப்பட்டனர். இதற்கு ஒரு காரணம் இருக்கிறது. தொடக்கப்பள்ளியிருந்து உயர்நிலைப்பள்ளிக்கு வரும் மாணவர்கள் மரபுத்தொடர்களை முதன்முதலாகப் படிக்கத் தொடங்குகின்றனர். உயர்நிலைப் பள்ளியில் தொடர்ந்து நான்காண்டுகள் மாணவர்கள் வெவ்வேறு மரபுத்தொடர்களைப் படிக்கவேண்டும். ஒவ்வோராண்டும் குறைந்தது பதினைந்து மரபுத்தொடர்களை மாணவர்கள் கற்றுக்கொள்வர். மேலும், மரபுத்தொடர்கள் பள்ளி மற்றும் தேசியநிலைத் தேர்வுகளில் சோதிக்கப்படுகின்றன. அதனால், உயர்நிலை ஒன்றாம் வகுப்பிலேயே மரபுத்தொடர்களைக் கற்றுக்கொள்வதில் அவர்களுக்குள் ஆர்வத்தை ஏற்படுத்திவிட்டால் நான்காண்டுகளும் அவர்கள் மரபுத்தொடர்களை விரும்பிக் கற்றுக்கொள்வர். மேலும், மரபுத்தொடர்களை நம் முன்னோர்கள் பயன்படுத்திய விதத்தையும் அதற்கானப் பொருளையும் எளிதாகவும் அறிந்துகொள்வர் என்ற நோக்கத்துடன் இதனை அணுகினோம்.

மாணவர்களுக்குத் தமிழ்த்திரைப்படங்களில் வரும் நகைச்சுவைக் காட்சிகளுக்குப் பொருத்தமான மரபுத்தொடரையும் அந்த மரபுத்தொடரைச் சரியாகப் புரிந்துகொண்டு தேர்ந்தெடுக்க மேலும் மூன்று தெரிவுகளும் காஹூட் விளையாட்டின் மூலம் கொடுக்கப்பட்டது. மாணவர்கள் அந்தக் காட்சியைத் திரையில் பார்த்து அங்குக் கொடுக்கப்பட்டிருக்கும் தெரிவுகளில் சரியான விடையைத் தங்கள் கைத்தொலைபேசி மூலம் உள்ளீடு செய்தனர். இதற்குக் கால அளவு உண்டு. அதற்குத் தகுந்தாற்போல் அவர்களுக்குப் புள்ளிகள் வழங்கப்பட்டது. வாரத்திற்கு ஒருமுறை மரபுத்தொடர்களைக் கற்பிப்பதற்கென இந்த காஹூட் வகுப்பு நடத்தப்பட்டது. மாணவர்கள் ஆர்வத்துடனும் அதே சமயம் இணையத்தின் வழியாக மிகுந்த ஊக்கமுடனும் இப்பயிற்சிகளைச் செய்தனர். ஒவ்வொரு பாடவேளை இறுதியிலும் மாணவர்கள் குறிப்பிட்ட சில மரபுத்தொடர்களைக் கற்றுக்கொண்டனர்.

கற்பித்தல் நிகழ்ந்தபோது மாணவர்கள் மரபுத்தொடரைக் கற்றுக்கொண்டதோடு சிந்தித்தல் திறன்களிலும் மேம்பட்டு விளங்கினர். 'முதலில் நான் செய்கிறேன், அடுத்து நீ செய், பிறகு நாம் சேர்ந்து செய்வோம்' என்ற உத்தியின் வழி மாணவர்கள் சரியாகச் சிந்திக்கக் கற்றுக்கொள்வதுடன் சரியான விடையையும் கண்டுபிடித்தனர். மேலும் ஒவ்வொரு மரபுத்தொடர்களுக்கும் ஆசிரியர் கொடுக்கும் விளக்கங்களை வைத்து மாணவர்கள் எளிதாக தங்கள் பாரம்பரிய பழக்கவழக்கங்களையும் பண்பாட்டுக் கூறுகளையும் கற்றுக்கொண்டனர். கற்பித்தல் நோக்கம் எளிதாக நிறைவேறுவதற்குக் காஹூட் விளையாட்டின்வழிக் கற்றல் மிக எளிதாகக் கைகொடுத்ததுடன் மாணவர்களின் மகிழ்வூட்டும் கற்றலுக்கும் வழிவகை செய்தது.

இன்று மாணவர்கள் காஹூட் வகுப்பை வாரத்திற்கு இருமுறை நடத்தச் சொல்லி வற்புறுத்துகின்றனர். மேலும் தமிழ் வகுப்பிற்கு மிக ஆர்வமாக மாணவர்கள் வருவதைப் பார்க்க முடிகிறது. பாடத்திட்டத்தில் உள்ள இன்னும் வேறுசில கூறுகளைக் காஹூட்டின்வழி மாணவர்களுக்கு ஆர்வமூட்டும் வண்ணம் கற்பிக்க இயலும். அது தொடர்பான முயற்சிகளில் நாங்கள் ஈடுபட்டு வருகிறோம்.

பரிந்துரைகள்

•தமிழை இரண்டாம் மொழியாக படிக்கும் மாணவர்களுக்கு மொழியின்மேல் ஆர்வத்தை ஊட்ட முறைசாரா மதிப்பீட்டு முறைகளை – கற்றலுக்கான மதிப்பீட்டு முறைகளை, இணைய தொழில்நுட்ப வளங்களுடன் ஒருங்கிணைத்துக் கற்பிக்கலாம்.

- தமிழாரிசியர்கள் காஹூட் போன்று இணையத்தில் இலவசமாகக் கிடைக்கும் விளையாட்டுவழிக் கற்றல் தளங்களைக் தங்கள் கற்றல், கற்பித்தலுக்குப் பயன்படுத்திக்கொள்ளலாம்.

- உலகெங்கும் உள்ள தமிழாசிரியர்கள் காஹூட் – விளையாட்டுவழிக் கற்றல் தளத்தை பயன்படுத்தி தமிழ் கற்றல், கற்பித்தலுக்கு என தரமான வளங்களை உருவாக்கி அனைவருக்கும் பகிரலாம்.

கட்டுரைக்கு உதவிய வளங்கள்:

1. தமிழ்மொழிப் பாடத்திட்டம் 2011 (உயர்நிலை), பாடத்திட்ட வரைவு மற்றும் மேம்பாட்டு பிரிவு, கல்வி அமைச்சு, சிங்கப்பூர்.
2. தமிழ்மொழி மதிப்பீட்டு வழிகாட்டி 2012 (உயர்நிலை), பாடத்திட்ட வரைவு மற்றும் மேம்பாட்டு பிரிவு, கல்வி அமைச்சு, சிங்கப்பூர்.
3. Bawa, P. (2019). Using Kahoot to Inspire. Journal of Educational Technology Systems, 47(3), 373-390.
4. https://en.wikipedia.org/wiki/Kahoot!
5. Zarzycka-Piskorz, E. (2016). Kahoot it or not? Can games be motivating in learning grammar? Teaching English with Technology, 16(3), 17-36.
6. Wichadee, S., & Pattanapichet, F. (2018). Enhancement of performance and motivation through application of digital games in an English language class. Teaching English with Technology, 18(1), 77-92.

Building Divergence Index for Telugu-Tamil Machine Translation: Quantification of Case Divergence

Parameswari, K.
Centre for Applied Linguistics and Translation Studies,
University of Hyderabad
parameshkrishnaa@gmail.com

Abstract. The major difficulty in building a machine translation system (MT) emerges due to translation divergence, a consequence of cross-linguistic differences between languages. A systematic translation mapping of source language syntactic structures to a target language is one of the main goals of any MT systems. This paper deals with divergences that arise in the context of case marking (overt/ covert) between Telugu and Tamil. In this paper, divergence Index (DI) is used as a method to compute linguistic divergence which aims at quantifying the number of parametric variations found between a pair of languages and facilitates MT in proposing where to put efforts for the given language pair to attain a better and faster result. It attempts to quantify case divergences and proposes a solution to handle them in transfer-based MT. This paper also demonstrates how DI is built to compute linguistic divergence between them in the context of building MT.

1. Introduction

Tamil and Telugu, being Dravidian languages share a certain number of syntactic similarities while in a number of cases exhibit syntactic divergences with each other at various levels. It can be anticipated that these syntactic divergences often create catastrophic effects on MT output. In certain cases, languages show a great mismatch in the patterning of the morphological case against the patterning of the syntactic case. It arises due to the distinction of a case (form) and its grammatical role (function) that it expresses. Each case marker has a number of functions and it is obvious that they lead to case mismatches in MT and precludes the straightforward mapping of it in the target language. Hence, any effort in the direct mapping of a case marker often results in a non-standard or ungrammatical construction in the target language. To unravel all these difficulties, it is necessary to study different functions of case markers which may facilitate the source language disambiguation and target language generation. In MT, mapping case markers with their suitable functions between languages are significant to obtain better results.

2. Case marking in Telugu and Tamil

Case marking is one of the morpho-syntactic functions expressed in terms of inflectional properties of nouns (including pronouns and number words) in Tamil and Telugu. It is exhibited as suffixes to express different functions. The inflection of case is overtly marked except in the case of nominative which is null (∅) in both the languages as in Table (1).

Case	**Telugu**	**Tamil**
Nominative	*∅*	*∅*
Accusative	*-ni/-nu*	*-ai*
Dative	*-ki/-ku*	*-(u)kku/-ku*
Instrumental	*-tō*	*-āl*
Associative	*-tō(pāṭu)*	*-ōṭu/uṭaṉ*
Locative	*-lō/daggara*	*-il/-iṭam*
Ablative	*-nuMḍi/-nuMci*	*-il iruntu/- iṭam iruntu*
Genitive	*∅/-yokka*	*-iṉ/-uṭaiya/-atu*

Table (1): Case Markers in Telugu and Tamil

3. Case mismatches in Telugu and Tamil

The major reason for case mismatches between Telugu and Tamil is due to case syncretism. Case syncretism occurs when a single inflected form corresponds to two or more case functions (Comrie 1991: 44-47). Distinct case values are determined on a language-specific basis so that the case syncretism by the definition involves an observable asymmetry between paradigms within a language (Baerman, 2009:219). It is possible that a case marker with multiple functions in a language may correspond to a single function in another language. This situation definitely seeks an intense study in disambiguating case marker in the source language from its multiple functions and leading to an appropriate substitution by a corresponding one in the target language. In this section, a few examples of case divergence between Telugu and Tamil are explicated.

The nominative case marked noun is used as a subject in both Telugu and Tamil. When a subject is in the nominative case, it controls verb agreement as in the example (1).

(1)	Te.	*rāmuḍu$_i$*	*ikkaḍi- ki*	*vacc-*	*ā-*	*ḍu$_i$.*
		Ram.NOM	here- DAT	come-	PST-	3.SG.M.
	Ta.	*rāmaṉ$_i$*	*iṅkē*	*va-*	*nt-*	*āṉ$_i$.*
		Ram.NOM	here	come-	PST-	3.SG.M.
		`Ram came here'				

However, when a predicate indicates the capabilitative mood, the subject is marked with the instrumental case marker in Tamil, whereas the subject in Telugu is in the nominative (cf. Krishnamurti and Gwynn, 1985 for Telugu, Lehmann, 1993 for Tamil). . In such cases, the verb gets default agreement in Tamil. Example:

(2)	Te.	*nēnu*	*ī*	*pani*	*ceyy-a*	*gala-nu*
		I.NOM	this	work	do-INF	can-1.SG.
	Ta.	*eṉṉ-āl*	*inta*	*vēlai-ai*	*ceyy-a*	*muṭiy-um*
		I-INS	this	work-ACC	do-INF	can-3.SG.N. (default)
		`I can do this work'				

A causative verb as a predicate may support up to three arguments in the form of noun phrases, viz., causer agent, actor agent, and object. The second agent is considered as the real actor, whereas the first agent causes the second agent to act. Here the second agent phrase is marked for the postposition *cēta* `by means of' in Telugu and by the accusative case marker in Tamil as shown in (3).

(3)	Te.	*nēnu*	*vāḍi-*	*cēta*	*iMṭi*	*panu-*	*lu*	*cēy-*	*iMc-*	*ā-*	*nu.*
		I.NOM	he-	by	house.OBL	task-	PL	do-	CAUS-	PST-	1.SG.
	Ta.	*nāṉ*	*avaṉ-*	*ai*	*vīṭṭu*	*vēlai.(y-* | *ai)*	*ceyy-*	*a-*	*vai-*	*tt-*	*ēṉ.*
		I.NOM	he-	ACC	house.OBL	task-	ACC	do-	INF-	CAUS-	PST-	1.SG.
		`I made him to do the household tasks.'										

An object-complement double accusative is a construction in which one accusative is the direct object of the verb and the other accusative (either noun, adjective or participle) complements the object in that it predicates something about it (Wallace, 1985:93). In Telugu, the accusative case marker is assigned to the object irrespective of the animacy and definiteness features. The grammatical case of the object is also percolated/copied to its complement in Telugu as in (4). Whereas in Tamil, the object complement does not take the accusative and thus realized in its nominative form.

(4)	Te.	*nēnu*	*vāḍi-*	*ni*	*[rāyi-*	*ni/*	*maniṣi-*	*ni]*	*cēs-*	*ā-*	*nu.*
		I	he-	ACC	[stone-	ACC/	human-	ACC	do-	PST-	1.SG.
	Ta.	*nāṉ*	*avaṉ-*	*ai*	*[kal/*	*maṉitaṉ]*	*ākk-*	*iṉ-*	*ēṉ*		
		I	he-	ACC	stone.NOM/	human.NOM	make-	PST-	1.SG.		
		`I made him into a stone/a human.'									

In Telugu, the verb *ceyyi* `do' functions as a periphrastic causative verb and allows two accusative case marked noun phrases in a row. It is illustrated as below:
Te. NP(causer) NP(causee)-ACC NP(theme)-ACC VGF <root= "*ceyyi*" 'do'>
Ta. NP(causer) NP(causee)-ACC NP(theme)-NOM VGF<root= *"ākku"* 'make'>

The dative subject construction is the most widespread in Dravidian (Subbarao, 2012:134). The dative marked subject acts as an experiencer subject and the verb agrees with the object. In Tamil, stative predicates expressing the notion of mental, emotional and physical experience require the case-marking pattern of DAT-ACC (Lehmann, 1993:180).

(5) Te. *nā-ku* *vāḍu* *telusu*
I-DAT he-NOM know
Ta. *eṉa-kku* *avaṉ-ai.t* *teriyum*
I-DAT he-ACC know
'I know him'

Nouns inflected for the dative have a locative function in Telugu (Ramarao, 1975). To express a possessive relationship between two inanimate nouns, one of the nouns of inanimate category inflected for the dative in Telugu expresses the locative function as seen in (6). On the contrary, a locative marker is used in Tamil. Here, the dative case marker relates two noun phrases which have holonymic-meronymic relationship. The word which is a holonym takes the dative case marker in Telugu and locative in Tamil. In the following example, a wall which is the holonym of a window is marked for the dative in Telugu and the locative in Tamil to exhibit the location by means of possessive association between them.

(6) Te. *gōḍa-ku* *kiṭiki* *uMdi*
wall-DAT window.NOM be
Ta. *cuvaṟṟ-il* *jaṉṉal* *uḷḷatu*
wall-LOC window.NOM be
`The wall has a window'

In Telugu, a noun (NN) followed by a directional noun (which functions as a postposition (PSP)) inflects for the dative, whereas in Tamil the complement noun receives the dative. The swapping of the dative is noticed between the languages. We use dative swapping to refer to the exchange of the dative between the PSP and its complement noun in the translated structure. It is realized as a kind of configurational difference between Telugu and Tamil. It can be illustrated as in the following:
Te. (([NN]) ([PSP]+[DAT])) ⇔ Ta. (([NN]+[DAT/GEN]) ([PSP]))

(7) Te. *kumāru* *iMṭi-* *lōpali-ki* *veḷḷ-* *ā-* *ḍu.*
Kumar house- Inside-DAT go- PST- 3.SG.M.
Ta. *kumār* *vīṭṭ-ukku* *uḷḷē* *pō-* *ṉ-* *āṉ.*
Kumar horse-DAT inside go- PST- 3.SG.M.
`Kumar went inside the house'

Nouns[+animate] marked for the instrumental (alternatively dative) case act as recipients/ hearers when they occur as arguments of the verb[+communication] in Telugu. Whereas in Tamil, such nouns carry the locative case marker to infer similar function. Example:

(8) Te. *nēnu* *vāḍi-* *tō/ -ki* *ā* *viṣayaM* *cepp-* *ā-* *nu.*
I he- INS/ DAT that matter tell- PST- 1.SG.
Ta. *nāṉ* *avaṉ-* *iṭam* *anta* *viṣayatt-ai.c* *co-* *ṉṉ-* *ēṉ*
I he- LOC that matter-ACC tell- PST- 1.SG.
'I told him the matter'

Nouns[+place, +generic] such as *akkaḍa* 'there', *ikkaḍa* 'here' etc., in Telugu, occur adnominally and modify a noun phrase. Whereas in Tamil, they cannot occur in adnominal position and thus the adjectival participle form of existential verb *uḷḷa* 'to be' is inserted between nouns as shown in (9).

(9) Te. *akkaḍi* *prajalu*
there.OBL people
Ta. *aṅk-uḷḷa* *makkaḷ*
there-be people
`The people over there …'

As seen from examples (2-9), Telugu and Tamil show divergence in case marking which precludes straightforward mapping while building machine translation systems involving these languages.

4. Quantification of Case Divergence

The divergence in the usage of the case maker is quantified based on their different domains of occurrence. The following situations are counted for divergence.

Let languages be A and B.

1. If languages A and B use the similar case marker invariably, then no divergence
2. If anyone of the languages i.e. A or B differ in their usage, then counted as divergence
3. If anyone of the languages i.e. A or B 'optionally' realized with the different case marker, then counted as divergence

Divergence index (DI) represents a measure of the differences that occur between languages. The variations in linguistic features can be seen in any levels (L) (surface, shallow, intermediate, deep and deeper levels). These levels are identified as L1 (surface level), L2 (shallow level), L3 (intermediate level), L4 (deep level) and L5 (deeper level) according to its depth of variation. Table (2) lists the quantification of similar and divergent case functions between Telugu and Tamil. Divergent functions are also with different levels where they are identified. The percentage of divergence is calculated as divergent functions over total functions.

Case Marker	**Similar Functions**	**Divergent functions**	**Divergent Levels**					**Percentage of Divergence**
			L1	**L2**	**L3**	**L4**	**L5**	
Nominative	3	6	0	6	0	0	0	66.7%
Accusative	11	13	0	4	6	2	1	54.1%
Dative	32	36	6	9	18	1	2	52.9%
Instrumental	7	6	2	2	2	0	0	46.1%
Associative	10	7	1	5	1	0	0	41.2%
Locative	18	8	2	6	0	0	0	30.8%
Ablative	7	4	2	2	0	0	0	36.4%
Genitive	7	3	3	0	0	0	0	30%
Total	95	83	16	34	27	3	3	46.62%

Table (2): Quantification of Case Divergence between Telugu and Tamil

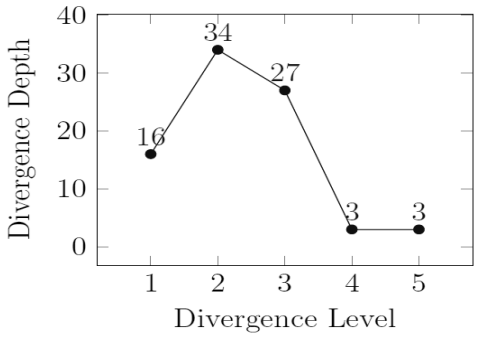


Fig.1 Case divergence between Telugu and Tamil

As seen in Table (2), each case marker show similar and divergent functions between Telugu and Tamil. The overall case divergence is identified as 46.62% between Telugu and Tamil. It is also noted as in Fig(1) that divergences fall mostly on L1, L2 and L3.

5. Handling Strategies in Machine Translation

A number of NLP modules are used in handling divergences in transfer-based MT. In the disambiguation process, the ontology of lexical items is looked in and accordingly the target language case marker is substituted. To handle the case marking, modules such as dependency parser, transfer grammars (TG) and morphological generators (MG) are used (Parameswari, 2014). The dependency parser provides the relationship between nouns and a verb in a sentence. It disambiguates the functions of case marker by providing relationship tags. TG is equipped with performing certain tasks such as insertion, deletion, modification and re-ordering of words and chunks. It also has the ability to handle files where it is possible to operate a single rule over a list of items. MG generates well-formed wordforms of target language based on the grammatical feature.

6. Conclusion

Case marking in Telugu and Tamil show a considerable amount of divergence and hence, the straightforward mapping between these languages are not possible in MT. NLP modules such as parsers, TG and MG are useful in handling these divergences. Quantification of divergence facilitates MT in proposing where to put efforts for the given language pair to attain a better result. A systematic study of divergences and proper handling of them are indispensable to get comprehensive output in MT.

Acknowledgment: I would like to acknowledge members of Indian Language to Indian Language (IL-IL) MT project and Prof. G. Uma Maheshwar Rao, the chief investigator of Telugu-Tamil and Telugu-Tamil bidirectional MT systems for giving me an opportunity in building Telugu-Tamil MT systems.

Reference

1. Baerman, Matthew. 2009. Case Syncretism. Oxford: Oxford University Press.
2. Comrie, Bernard. 1991. `Form and Function in Identifying Cases'. In Frans lank (ed.),Paradigms: The Economy of Inection, 41{56. Berlin: Walter de Gruyter.
3. Krishnamurti, Bh. & J. P. L. Gwynn. (1985). *A Grammar of Modern Telugu*. Delhi: Oxford University Press.
4. Lehmann, Thomas. (1993). *A Grammar of Modern Tamil*. Pondicherry: Pondicherry Institute of Linguistics and Culture.
5. Parameswari, K. (2014*). Development of Telugu-Tamil Machine Translation: With Special Reference to Divergenc*e. Unpublished Ph.D. Thesis. Hyderabad: University of Hyderabad.
6. Ramarao, Chekuri. (1975). *Telugu vākyam* [`The Telugu Sentence'- in Telugu]. Hyderabad: AP Sahitya Academy.
7. Subbarao, K. V. (2012*). South Asian Languages: A Syntactic Typology*. Cambridge:Cambridge University Press.

8. Wallace, Daniel B. (1985). `The Semantics and Exegetical Significance of the Object-Complement Construction in the New Testament'. *Grace Theological Journal* 6(1). 113-160.

Handwritten Tamil OCR by Using the Statistical and Structural Theory

M. Antony Robert Raj[1*], S. Abirami[1], S. M Shyni[2], S. Murugappan[3],

[1]Department of Information Science and Technology, Anna University, Chennai–25

[2] Department of Electrical and Electronics Engineering, Sathyabama Institute of Science and Technology, Chennai,

antorobert, abirami.murugappan, shynima, drmryes{@gmail.com }

Abstract. Handwritten character recognition is a challenging and interesting research field when the intended use is to characterize the level of sophistication and intricacy. This paper deals with the distinctive and effective feature extraction techniques such as KD-tree and Directional pixel location representation for taking out the feature values from the character image. Final node values of the tree and the weightage of the directional points are employed to train the classifier, Support Vector Machine (SVM). For training and testing purposes, the samples are collected and clustered into two groups based on their level of complexity. The classification process occurs distinctively on the features and the performance of the same is noted for each classified group. The final outcome shows that these features extraction techniques can confront the valuable percentage of difficulties.

Keywords: KD tree, Directional pixel point, Quad and Support Vector Machine.

1 Introduction

The enormous need for computerizing the Tamil language prompts in order to involve Tamil handwritten character recognition is the central focus of this research. Applications are exploited in the field of banking for the purpose of the processing of checks, postal address verification, form verification and recognition of any communication devices through the Tamil language, there is a plethora of historical documents, poems, and palm scripts available for recognizing. Attainment of this is not an easy task; there is a multitude factors that need to be considered [29]. Tamil characters have similar and cursive shapes by default and these sets show a high-end variation in shape, location, style variation of shape, broken structure, and excessive loops and portions in handwritten Tamil characters.

The Tamil language contains a large character set (247 Characters), wherein 12 vowels, 18 consonants, 216 combination characters, and one special character are available. Generally, the shape of those characters are cursive (Vattezhuth). Recognition of this cursive letter can be possible in two modes, namely structural and statistical [30]. In a structural way, shape or direction of the character portion is taken into account directly. Alternatively, in the statistical approach, the pixel position or variation is considered for extracting the feature values. Several experiments conducted prior and in current research provide solutions for the same.

In a statistical way, an additional two modes of feature extraction techniques are possible, they are pixel-based and location-based. In previous works, the features such as pixel variation, density, different direction based pixel position and axis based location are implemented in the pixel-based feature extraction [4]. For identifying the location of the features, zone and quad tree procedure is incorporated [11, 28]. The two-way structural theory was also identified during the experimental analysis. It consisted of a directional procedure and a shape based notation. In the directional procedure, sub-line direction [5], shape's direction and chain code [28, 31] based directional feature extraction techniques were tested. Strip tree [27] and prism tree [32] shape features were tested for identifying the shape based features. SVM [5,28, and 32] and decision tree [27] classifiers were processed for classifying the correct Tamil characters. A maximum of 30 characters was tested in each way of the process.

In the related works, significant data was available in the field of offline Tamil handwritten character recognition. Jagadeesh Kumar R et al. [26] gathered the features such as normalized coordinates, moralized derivatives, curvature, aspect curliness and lineness for Time-domain features and the sliding window, stroke, discrete cosine transform and the window sequence for the frequency domain. The classifier Hidden Markov Model was implemented and achieved 93% to 98% results. The zoning procedure experimented in the works of Rajashekararadhya et al. [6, 25], wherein centroid-based features are collected in various aspects and considered for the purpose of classification. By applying the neural network and nearest neighbor [25] about 96% accuracy is achieved within the results. The performance rate was 94.9 and 93.9 when applying the nearest neighbor and SVM. Totally, 10 characters (Tamil numerals) were considered for their works. Shanthi et al. [8] also used the zoning procedure for extracting the pixel density values as features. With the implementation of SVM about 82.04% accuracy is achieved for the 34 Selected Tamil characters. The Tamil characters possess characteristic properties such as definitive width, height, circle, vertical and horizontal lines, dot, and the curve were identified and taken as features. With the help of a neural network classifier 97% accurate results were achieved for the characters containing minor variations. Sigappi et al. [12] experimented a profiling procedure for extracting the features and implemented a hidden Markov model for classifying the characters. 90% of the accuracy rate achieved for 40 different Tamil words. Scale Invariant Feature Transform (SIFT) [24] was extracted from the important points of the characters. These points were grouped as codebook by using the k-mean procedure. The classifier K-nearest neighbor applied to achieve 87% results.

In the process of feature analysis, the study has been made on the various works of handwritten recognition of different languages. Subhadip Basu, Nibaran Das et al. [2] used a quad-tree based image partitioning technique and longest-run features for recognizing multi-script (Latin, Devanagari, Bangla, and Urdu) postal code. The same features were used by Nibaran Das, Ram Sarkar et. al. [3] and Ritesh Sarkhel, Nibaran Das [10] for recognizing Bangla characters. XU Xiaobing, WU Xiaoxu et.al [23] were extracted the directional features from horizontal, vertical, left and right-diagonal stroke of the Chinese characters. In the work of Ming-Ke Zhou, Xu-Yao Zhang et. al [14] and Javad Sadri, Mohammad Reza Yeganehzad et. al [15], the original gradient direction features were extracted and used to classify the Chinese characters [14] and digits [15]. Abdelhak Boukharouba, Abdelhak Bennia [9] used Freeman chain code procedure to extract directional features from the vertical and horizontal directions given an image for classifying the digits. Jija Dasgupta, Kallol Bhattacharya et. al [16] extracted the features directional and stroke orientation distribution for classifying the English characters. With this consideration, this paper deals with the shape and direction in a statistical approach. Two unique methods KD-tree and directional pixel location are employed on the character for extracting the feature in tree representation and the directional weight factor. Lastly, the features are gathered into the SVM as training and testing sets for predicting the right character.

2. General Phases

Selectively, essential image processing techniques are used for this process. Techniques such as Image pre-processing, feature selection, Feature extraction, and Classification. The selected algorithm in each phase is detailed in the following figure (figure 1).

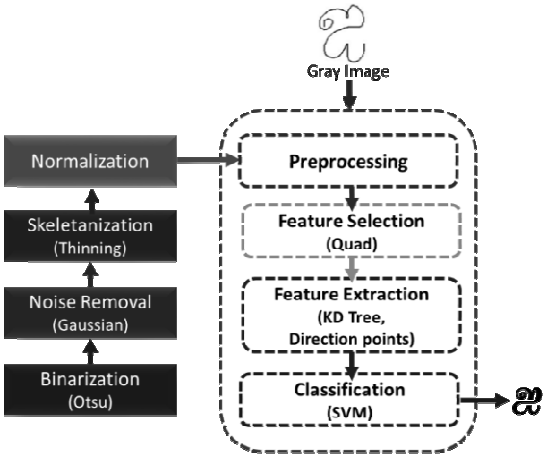


Figure 1. Phases of Tamil handwritten character recognition.

In the pre-processing techniques, Otsu's procedure, Thinning algorithm, and the Gaussian process are used for getting the binary, skeletonized and noiseless image for this experimental procedure respectively. The default algorithms which was used in the previous works are used for this experiment as well [4, 5 11, 27-32]. All character samples are collected from HP-India Lab Datasets [1]. Each character structures are stored in the character image in an isolated form. The Image contains a low noise ratio since the image is taken into the default Gaussian process to remove the unwanted pixels present in the character image. The image is standardized into 200X200 pixel size. The segmentation process is not used for this work since all character images are collected individually. The image is divided by the quad and KD-tree and the directional points are used to extract the features from the character portion of the image selected by quad. Finally, the supervised classification algorithm SVM is used for classifying the characters.

3. Feature Selection and Extraction

The noiseless character image is taken into special treatment in order to extract the better features. Each character portion of the image is separated by the quad procedure and the following process is employed to extract the features from the character parts.

3.1 KD Tree

The KD tree process [13] is normally used for object detection or for object representation. The KD tree method is exploited in this experiment owing to its efficient and effective ability to isolate and point out a portion of a character explicitly. In this experiment, the character portions are in the form of continuous pixels, keeping this in hindsight the standard KD tree model has modified accordingly, whilst keeping the general rules intact. The following procedure elaborates the working of the KD tree process. It describes the technique of feature extraction from the character portion.

3.2 Feature Selection

Let, the character image be denoted as IM,

Divided (IM) → IM{Qr_1, Qr_2, Qr_3, Qr_4}, where Qr_1, Qr_2, Qr_3, Qr_4 are the quad portions of the image IM.

In the tree formation, Root ← (IM) and Leaf (IM) ← {Qr_1, Qr_2, Qr_3, Qr_4}.

Represented the image (IM) as the root (IM) and the sub-quads (Qr_1, Qr_2, Qr_3, Qr_4) as a leaf node of the tree as shown in the following figure (figure 2)

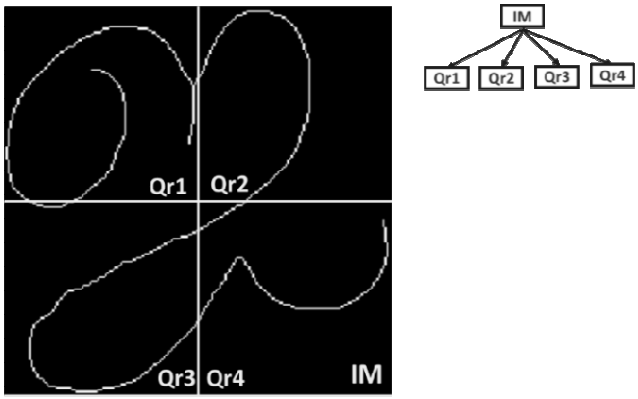


Figure. 2. *Quad representation in an image*

Feature Representation:

Let, quadrant Qr_1 → Sub-root of the root IM in the tree. In the quadrant Qr_1, each pixel location has been analyzed to identify any white pixels (pixel value is one in the binary image) are available. Let values be represented such that location (Loc) of point (i, j) with respect to the pixel (P) of the quadrant Qr_1. If the white pixels are found in the quadrant Qr_1, then the condition (equation 1) must be 1.

$$\mathrm{Loc}_{i,j}^{P}(\mathrm{Qr1}) = 1, \quad \text{where,} \begin{cases} 1 & Pixel \in \mathrm{Qr1}, \\ 0 & Pixel \notin \mathrm{Qr1}, \end{cases} \qquad \text{- (1)}$$

If pixels are found in the particular quadrant, then there are four conditions possible to draw the KD tree by using this.

Condition1: Valid pixel point with one neighbor.

Let's take, counter=1 and the counter has been incremented (counter++) if the following condition (equation 2) is satisfied.

$$\sum_{i,j=1}^{N} neig(\mathrm{Loc}_{i,j}^{P}(\mathrm{Qr1})) = 1 \qquad \text{- (2)}$$

where,

$$\mathrm{Loc}_{i,j}^{P}(\mathrm{Qr1}) \neq RM_{i,j}, \mathrm{Loc}_{i,j}^{P}(\mathrm{Qr1}) \neq LM_{i,j},$$

$$\mathrm{Loc}_{i,j}^{P}(\mathrm{Qr1}) \neq TM_{i,j}, \mathrm{Loc}_{i,j}^{P}(\mathrm{Qr1}) \neq BM_{i,j}$$

Here, N= 8, denoting the eight neighbors of the particular valid pixel and $RM_{i,j}$, $LM_{i,j}$, $TM_{i,j}$ and $BM_{i,j}$ are the location point (i, j) on the right margin, left a margin, top margin and bottom margin respectively. If the sum of all eight directional neighbor pixels (neig) of particular valid pixel location of the quadrant Qr_1 is 1, signifies that there is a pixel found with one neighbor. If the counter shows more than one, then there are a number of pixels available with one neighbor, ie, $Loc_{i,j}$, $Loc_{i+1,j+1}$, $Loc_{i+2,j+2}$, ..., where $Loc_{i,j}$ is pointing the location of the first pixel with one neighbor and the rest of them are pointing the locations of the

next pixels with one neighbor. Assume, if there are two-pixel locations of the valid pixels with one neighbor ($Loc_{i,j}$, $Loc_{i+1,j+1}$), then two points are noted in the right margin ($RM_{i,j}$, $RM_{i+1,j+1}$) which are perpendicular to the point ($Loc_{i,j}$, $Loc_{i+1,j+1}$). Further, the distance between those locations and marginal points are calculated by using equation 3 and 4.

$$\frac{Loc_i+RM_i}{2} + \frac{Loc_j+RM_j}{2} = X_1 \qquad -(3)$$

$$\frac{Loc_{i+1}+RM_{i+1}}{2} + \frac{Loc_{j+1}+RM_{j+1}}{2} = X_2 \qquad -(4)$$

If $X_1 > X_1$, then a perpendicular line has been drawn between the left margin and the right margin of the quadrant (Qr_1) towards the point $Loc_{i,j}$. Else, the line has been drawn towards the point $Loc_{i+1,j+1}$. This line divides the quadrant into two rectangular parts (P1, P2) as shown in figure 3. This has been taken as the leaf node of the root Qr_1.

If counter=1, then there is only one valid pixel point found with one neighbor ($Loc_{i,j}$). So, a perpendicular line has been drawn from left margin to right margin towards the point $Loc_{i,j}$and take out two separated rectangular parts (P1, P2) as discussed earlier and the same has been represented as a leaf node of the root Qr_1 in the tree.

Condition2: Junction point

There are no valid pixel points with one neighbor, then experiment continues on the way to find the junction points, where, getting the pixel point which contains three or four neighbors. The counter has been declared (counter=1) and incremented if the following condition (equation 5) is satisfied.

$$\sum_{i,j=1}^{N} neig(\mathrm{Loc}_{i,j}^{P}(\mathrm{Qr1})) = 3 \; or \; 4, \qquad -(5)$$

$$\text{where, } \begin{cases} value & Pixel \in \mathrm{Qr1}, \\ 0 & Pixel \notin \mathrm{Qr1}, \end{cases}$$

where N denotes the eight directional pixels of the valid pixel point. If the counter>2, then there are three and above valid pixels found with three or four neighbors, the locational pointer of that valid point are denoted as $Loc_{i,j}$, $Loc_{i+1,j+1}$, $Loc_{i+2,j+2}$, ..., the same procedures that are discussed in condition1 are used here to continue the experiment for separating the rectangular parts (P1 and P2). If the counter==1, then only one valid pixel pointer found with three or four neighbors. The separation of the (P1 and P2) on the quadrant has been performed on the point by using the perpendicular line. The same has been marked in the tree as the leaf node of the Qr_1 as shown in figure 3.

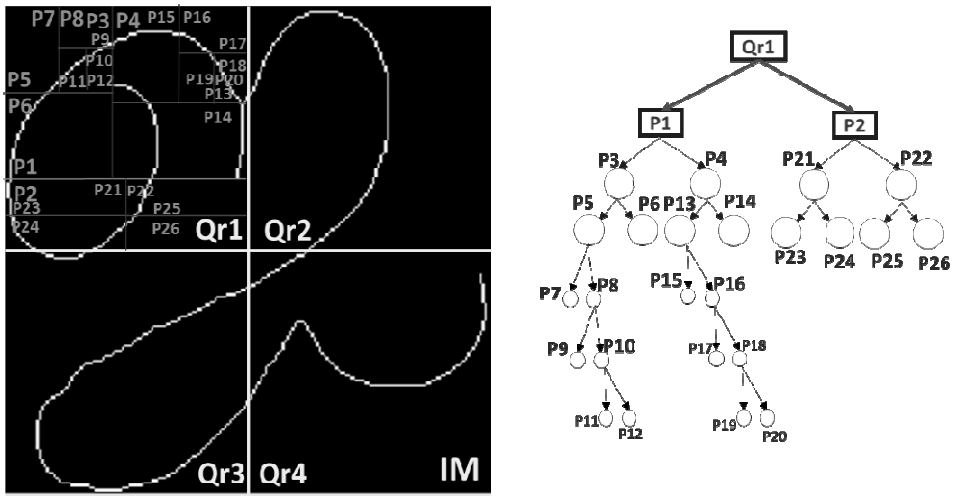


Figure. *3*. *KD Tree representation for Quad 1(Qr1).*

Condition 3: Two neighbor pixels.

If both conditions fail, the following condition (equation 6) is experimented without initializing the counter,

$$\sum_{i,j=1}^{N} neig(\mathrm{Loc}_{i,j}^{P}(\mathrm{Qr1})) = 2 \qquad \text{- (6)}$$

where,

$$\mathrm{Loc}_{i,j}^{P}(\mathrm{Qr1}) \neq RM_{i,j}, \mathrm{Loc}_{i,j}^{P}(\mathrm{Qr1}) \neq LM_{i,j},$$

$$\mathrm{Loc}_{i,j}^{P}(\mathrm{Qr1}) \neq TM_{i,j}, \mathrm{Loc}_{i,j}^{P}(\mathrm{Qr1}) \neq BM_{i,j}$$

This testing, analyses whether any valid pixel locations are available with two neighbors except on each border on the quadrant Qr_1. If the condition (equation 6) is true, then the two endpoints of the right margin are noted ($T_{i,j,}$ and $B_{i,j}$). The mid-point of the margin has been calculated by using those points in equation 7.

$$\frac{T_i + B_i}{2} + \frac{T_j + B_j}{2} = X_3 \qquad \text{– (7)}$$

A perpendicular line has been drawn from the midpoint (X_3) of the right margin to left margin. If this line meets the valid pixel then the division (P1 and P2) is possible and the tree can also be extended to the leaf node, otherwise, the division is impossible in this quadrant and the tree cannot be extended from the root node.

Condition4. Dot representation

In rare cases (maybe in subparts of KD division), only one pixel found in the quadrant (any parts), ie, valid pixel point with no neighbor.

$$\sum_{i,j=1}^{N} neig(\mathrm{Loc}_{i,j}^{P}(\mathrm{Qr1})) = 0 \qquad \text{- (8)}$$

where,

$$\mathrm{Loc}^{P}_{i,j}(\mathrm{Qr1}) \neq RM_{i,j}, \mathrm{Loc}^{P}_{i,j}(\mathrm{Qr1}) \neq LM_{i,j},$$
$$\mathrm{Loc}^{P}_{i,j}(\mathrm{Qr1}) \neq TM_{i,j}, \mathrm{Loc}^{P}_{i,j}(\mathrm{Qr1}) \neq BM_{i,j}$$

This point should not be from the margin because if the point is found in the margin it will be continued as a pixel point of the next quadrant. The location of this point has been noted as $Loc_{i,j}$. The perpendicular line is drawn from left margin to right margin to separate the quadrant (P1 and P2) and the same has been noted in the tree. The division cannot be continued further from those subnodes.

Moreover, the next level of iteration, the horizontal line which separated the quadrant into two parts ($Qr_1 \rightarrow$ P1 and P2) has been taken into consideration for continuing the experiment. The same condition rules are applied for P1 and P2 also, where the mid margin of P1 and P2 are considered for the next level of pixel-based subdivision. If the conditions (as discussed earlier) are satisfied, a vertical line will be drawn from mid to top or bottom margin to go to next level division P3, P4 and so on. The same procedure has been applied to each of the separated blocks, where if the mid-margin is a horizontal line, then the vertical line is used for separation, otherwise, a horizontal line is used for separating the blocks. The iteration has been continued until locating all character structure found in the quadrant (Qr_1) and the same has been represented in the tree as shown in figure 3.

Furthermore, the same procedure is applied to the next quadrant (Qr_2) also, where the separation has taken place from the center margin to the right margin (left to right). The horizontal perpendicular line is used for the first level of separation. The following figure (figure 4) shows the representation of the KD-Tree for the character part present in the Qr_2. In the Qr_3, if the above-said conditions (condition 1 to 4) are satisfied then the initial separation has been taken from the bottom mid to left margin. As shown in figure 5, the point separation and KD tree constructions have been established.

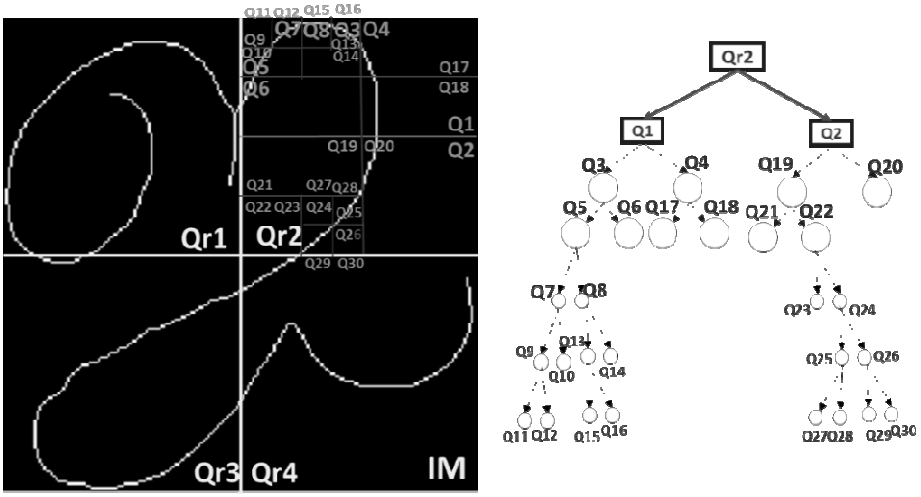


Figure *4.* *KD Tree representation for Quad 2(Qr2)*

In the final Quad (Qr_4), the valid pixel point identification starts from the bottom midpoint to the right margin. The valid pixel points are identified and the KD tree has been established as discussed in the first quad conditions. Finally, all formations which are extracted from each quad are merged and final formation has been found as shown in figure 5. The final tree has been formed by merging all trees which are extracted from each quad. The final tree formation is shown in figure 6.

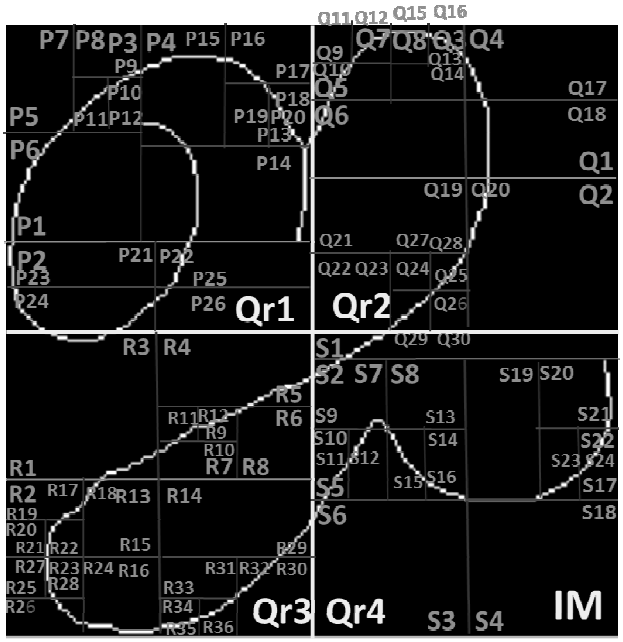


Figure. *5.* *KD Tree representation for all Quad.*

Finally, the feature values are extracted from the tree by using the depth-first search method, where tree traversal starts from the left node to root and root to the right node. The feature points are noted and used as features for classifying the correct characters. As per this experiment, some additional features are also essential to test the pixel availability. The feature experiment is continued for finding directional pixel points.

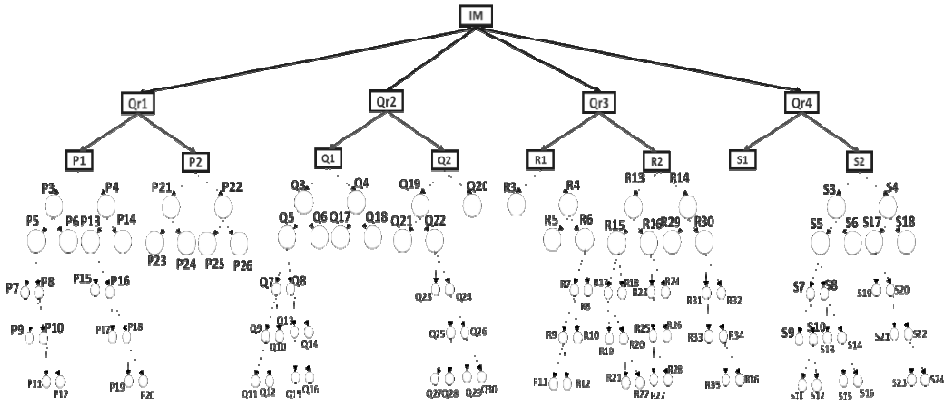


Figure. *6.* *KD Tree – extracted from the image IM.*

3.2 Directional Pixel Location

The character image (IM) subdivided into four equal quads (Qr_1, Qr_2, Qr_3, and Qr_4) as discussed in the previous section. The following condition (equation 9) is used for finding the midpoint of the image to do the same.

$$IM\left(\frac{Max(R)}{2}, \frac{\max(C)}{2}\right) \quad - (9)$$

where R represents the row and C represent the column of the image.

Initially, each row is tested to find the validation pixel points in each sub-quad. Here, two possibilities are identified and noted for the feature representation. They are the valid pixel point with one neighbor (equation 2) named as 'e' series and the junction point (valid pixel point with three or more junction points) (equation 5) tags as 'j' series. Also, the direction based location of the each point has been identified by the pointing the location (between (O_1_R + 0, O_1_C + 1) and (O_1_R – 1, O_1_C + 1), (O_1_R – 1, O_1_C + 1) and (O_1_R – 1, O_1_C + 0), (O_1_R – 1, O_1_C + 0) and (O_1_R – 1, O_1_C – 1), (O_1_R – 1, O_1_C – 1) and (O_1_R – 0, O_1_C – 1), (O_1_R – 0, O_1_C – 1) and (O_1_R + 1, O_1_C – 1), (O_1_R + 1, O_1_C – 1) and (O_1_R + 1, O_1_C + 0), (O_1_R + 1, O_1_C + 0) and (O_1_R + 1, O_1_C + 1 in Qr_1). The same procedure is continued in all quads (Qr_2, Qr_3, and Qr_4) as well. Additionally, to locate the valid edge pixel on the border of the quadrant, the testing has been done from the midpoint (O) of the image towards the direction (O_R + 0, O_C + 1), (O_R – 1, O_C + 0), (O_R – 0, O_C – 1) and (O_R + 1, O_C + 0), where O denotes the center point of the IM and R, C represents the row and column. Located points are named as a continuation of 'e' series.

Further, the center points of each quad have identified by the following conditions (equation 10 to 13).

$$IM\left(\frac{Max(R)}{4}, \frac{\max(C)}{4}\right) \rightarrow O_1 \quad \text{- (10)}$$

$$IM\left(\frac{Max(R)}{4}, \frac{\max(C)*3}{4}\right) \rightarrow O_2 \quad \text{- (11)}$$

$$IM\left(\frac{Max(R)*3}{4}, \frac{\max(C)}{4}\right) \rightarrow O_3 \quad \text{- (12)}$$

$$IM\left(\frac{Max(R)*3}{4}, \frac{\max(C)*3}{4}\right) \rightarrow O_4 \quad \text{- (13)}$$

From each centre point of the quad (O_1, O_2, O_3, O_4), eight directions (In quad O_1, (O_1_R + 0, O_1_C + 1), (O_1_R – 1, O_1_C + 1), (O_1_R – 1, O_1_C + 0), (O_1_R – 1, O_1_C – 1), (O_1_R – 0, O_1_C – 1), (O_1_R + 1, O_1_C – 1), (O_1_R + 1, O_1_C + 0) and (O_1_R + 1, O_1_C + 1)) travel has been established for finding the directional points. Each directional travel towards the end point of the quad, if it meets any valid pixel points (identified by the formula 1) then the same has been tagged ('t' series as shown in figure 7) and the same is stored in an array. The direction points which are located from each quad are represented in figure 8.

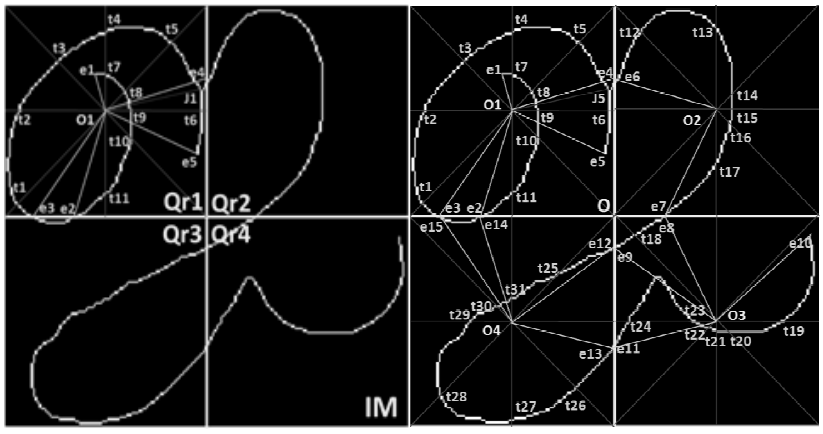


Figure. 7. *Locating direction pixel from the quad Qr1*. **Figure. 8.** *Locating direction pixel from all quads*.

A unique weightage for each point extracted from each direction, junction point and the endpoints and its directional locations is assigned and considered the same as representing the features (Ψ).

4. Feature Formations

In the KD Tree Process, 2^3 division process has been established which included the root process. If the feature detection process may or may not establish all division since it has been depending on the characters structures which visited in each quad. Hence the input feature division for KD Tree has been limited to 380 nodes and one root node (381 features). With this, additionally, (8 directions × 4 quads →32) directional pixel locations which included weightage of one neighbor pixel (1), junction Points (1) and border conditions (1) (total 32 + 3 → 35 features) are taken into the input of the classifiers.

5. Classification

A successful classifier in the pattern recognition environment, SVM has been used here for the purpose of classification of the correct characters. It is an effective and productive method when applying multi features [18]. It is a fruitful algorithm while applying directional features [16] [17] [19] for classifying the unique character sets. SVM supports high classification of characters by using the features as curvatures[33], different levels of granularity [20], quad-tree based image partitioning technique and longest-run features [2] [3] [10] [21], directional and transition features in the vertical and horizontal directions [9], and Rectangle Histogram Oriented Gradient [22]. Also, it has provided good outcome when considering certain different features, for example, octant centroid features, modified shadow features, number of loops and longest run features [21].

In order to use SVM with Radial Basis Function, we assume that, if there are two classes, the training data (X) which belongs to both classes are represented in the hyperplane. The classification has been done by using the $W^t X + b$, where b is the bias and W is a weight factor. The decision must be in the condition of $W^t X + b$, which will lead this into either positive form or negative form. For multiclass [7], the same experiment has been implemented in a hierarchical approach. This experiment has implemented in One Vs rest approach. (N-1) possible terms are established for this experiment. Here, the marginal division for the upper and lower bound is $2/\|w\| = 2/\sqrt{W^t W}$.

So, maximize the marginal division $2/\|w\|$ is equal to minimizing $\|w\|^2$

Based on the Karush Kuhm Tucker Theory, the optimization of SVM is

$$Y_i = \sum_{i=1}^{w} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N} \alpha_i \alpha_k t_i t_k \, k(x_i, x_k) - (14)$$

Where, $k(x_i,x_k) = x_i(k) = \exp(-\|x_i-x_k\|^2 / 2\sigma_k^2)$,

(x_i,t_i) are training data, i = 1,2, …, N

α_i is Lagrange multiplayer

and, (x_i, x_k) is Radial basis function $x_i(k)$

With these key terms, the multiclass SVM with Radial basis function has been implemented to classify the various classes of Tamil language

6. Data Collection and Analysis

The character samples are collected from the HP-India [1] and handwritten characters pof people from Tamilnadu, India. There are no words used for data collection, each character was collected individually. For these experiments, 10 characters from 12 vowels, 18 **க** series, three set of combination characters (க, ங, ச, ஞ, ட, ண, த, ந, ப, ம, ய, ர, ல, வ, ழ, ள, ற, ன, கி, ஙி, சி, ஞி, டி, ணி, தி, நி, பி, மி, யி, ரி, லி, வி, ழி, ளி, றி, னி, கீ, ஙீ, சீ, ஞீ, டீ, ணீ, தீ, நீ, பீ, மீ, யீ, ரீ, லீ, வீ, ழீ, ளீ, றீ, னீ, கு, ங, சு, ஞு, டு, ணு, து, நு, பு, மு, யு, ரு, லு, வு, ழு, ளு, று, னு அ, ஆ, இ, ஈ, உ, எ, ஏ, ஐ, ஒ, ஓ) are chosen (total 82 characters). Three hundred samples are collected for each character, totally 24600 samples are gathered. The collected samples are equally divided into two groups based on their complexity levels. The lower complexity samples formed the group (G1) and higher complexity characters are clustered in the group (G2). Each group is involved in this experiment separately and combined manner. Initially, in each group, 100 samples of each character are taken, for testing and remaining samples are taken into a training phase. For these experiments, three hundred samples are collected from each character, so this process has to be continued for all other samples as well. Initial cycle, 100 samples from single characters are chosen for testing while the remaining acts as training samples. In the next cycle, another 100 will be chosen for testing and the rest of them are considered as training samples. Finally, the last 100 samples are selected for testing and remaining for the training phase. The same process continues for the G2 group too. In the end, all samples are mixed where 600 samples are found for each character. As discussed procedure three cycles followed for these experiments, 200 samples are collected for testing and the rest of them for training in each cycle of the process. The results achieved for each process is discussed in the following section.

7. Result and Discussion

As discussed, the experiment was continued in three cycles for each group, the success rate of each cycle for the group G1 has been listed in the following table (table1). The handwritten complexities of this group's characters are very low, so the recognition result of the same had crossed above 90%. Comparatively, the success rates produced by the vowels are relatively lower. The main reason behind this is because two similar shape characters are found in the vowels group.

The positive possibilities for this experiment are fairly high. Most of the samples are recognizable by this algorithmic procedure. The failure rate of each group has been just less than 7% in each group. The failure of the characters in vowels is ஒ and ஓ, where numerous numbers of writers are missing the circle on the bottom part character ஓ. This algorithm is sensitive to locate this, but sometimes it failed. The same issue occurs among characters எ and ஏ. In க series and combinational characters, the failures are as a result of personalized writing styles of individual writers. The issues continued highly on the characters ல and வ, the main reason behind this is the handwriting style of வ looks like ல. The graph (figure 9) represents the resulting discussion of the group G1

Table 1. Success rate analysis of group1

Group	Character set	Testing Samples	Success Rate 1	Success Rate 2	Success Rate 3
G1	Vowels	100	90.6	92.7	94
	க series	100	92.05	94.3	95.1

	கி to னி	100	92.22	94.67	93.5
	கீ to னீ	100	93.94	94.78	93.89
	கு to னு	100	94.55	94.67	94.83

82 to 86% of success rate has been achieved for the characters in the group2 as discussed in table 2. This is a valuable result for this group of characters. The unique character gets success here; especially the characters 'ட', 'ப', 'ழ', and the same combination with other groups also are providing a success rate well above 95%. The characters ண, ணி, ணீ, and ணு are successfully recognized here since all samples of these characters are independent. Occasionally this series is misclassified with ன since writers write the character ண like ன.

Table 2. Success rate analysis of group2

Group	Character set	Testing Samples	Success Rate 1	Success Rate 2	Success Rate 3
	Vowels	100	83.3	82.7	86
	க series	100	85.2	85.67	85.83
G2	கி to னி	100	84.89	84.17	85.06
	கீ to னீ	100	84.06	84.78	84.72
	கு to னு	100	85.3	85	86.39

Due to the writer's mistake, the confusion among the various characters is increased. Even confusion is found among the characters தீ, நீ, றீ, தி, நி, றி, because when writers are writing நீ like றீ and நி like றி. A minor shape's location varies on the right top portion of the ற may be confused with ந. Some writers write the character தீ like நீ as they missed to write a curve in the left bottom of the character. The same issues can happen in other characters also.

In the combined group, all samples are used for classification purpose. As discussed earlier, 600 samples of each character are tested in three groups. The success rate of this is higher than the G2 group. The following table (table 3) shows the resulting rate achieved for all samples. Confusion among the characters is reduced here.

Table 3. Success rate analysis of group1 and group2

Group	Character set	Testing Samples	Success Rate 1	Success Rate 2	Success Rate 3
G1 and G2	Vowels	200	89.45	88.1	87.85
	க series	200	89.72	89.44	88.78
	கி to னி	200	88.28	88.25	87.97

கீ to னீ	200	88.5	89.39	89.42
கு to னு	200	88.89	89.5	89.22

As per the analysis, the success rate depends on the real shape of the characters as well. That is, the handwritten characters which contain real shape also must be included in the training samples to predict the unshaped characters. The following table (table 4) shows the comparative study of the success rate of each group. Additionally, the success rate also depends on the unique feature values of each character set. This algorithmic combination has successfully faced the challenges in deciphering the writer's complexity. Table 5 describes the results achieved from variouse works and the proposed works.

The success depends on the following factors

- KD tree locates the location of each characters sample
- Direction point also considers the location and variation in the location of the particular character portion
- Both algorithms consider the location, direction based on the structure of the shape, where the location of each shape is considered highly which leads to positive results

Table 4. Success rate Comparisons

Group	Success Rate 1	Success Rate 2	Success Rate 3
G1	92.88	94.38	94.29
G2	84.68	84.63	85.56
G1 and G2	88.92	89.02	88.73

Table 5. Results comparison among the various Tamil handwritech characters recognition works

Paper Ref	Algorithm Used	Characters Count	Results
Ref –[25] (Rajashekararadhya et al.)	Zoning, Neural network classifier	10 Nos	96%
Ref –[24] (Subashini et al.)	SAIF, K-MEAN	-	87%
Ref-[10] Ritesh Sarkhel et al.	CNN	10 Nos	99.7%
Ref - [26] (Shanthi et al.)	Zoning, SVM	34 Nos	82.4%
Ref - [12] (Sigappi et al.)	Vertical and word profile, HMM	Words-40 Nos	90%
Proposed	**KD-Tree Drectioanl, SVM**	**82 Nos**	**88- 89%**

7.1. Challenging Factors

Several challenging factors were confronted by this experimental analysis.

- The character confusion as a result of personalized writing styles by individual writers among many characters.
- Location mismatch may create confusion among the characters
- Minor variation in similar shape characters is misclassified as another character altogether
- Missing some portion of unique characters also leads to misclassification
- Adding unnecessary portion such as in the case of italic writing also predicts the negative results

8. Conclusion

With a concentration in location, KD-tree and directional pixel location algorithmic procedure are experimented to extract features from the character image. The features are trained and tested in the SVM classifier for predicting the correct characters. 88 to 89% accuracy has been achieved by this experimental analysis. This process was experimentally successful for a specified set of characters and its samples. This experiment has been established to test the algorithmic procedure which is highly focused on location. This can fit into a specific group of character with a variation. As per the analysis, a shape based algorithmic procedures are also needed to handle the challenges towards similar shapes.

References

1. HP-India Tamil handwritten characters Data set, http//lipitk.sourceforge.net/hpl-datasets.htm, Accessed, 10, June 10 2014.
2. Subhadip Basu, Nibaran Das, Ram Sarkar, Mahantapas Kundu, Mita Nasipuri, Dipak Kumar Basu. A Novel Framework for Automatic Sorting of Postal Documents With Multi-Script Address Blocks, Elsevier, Pattern Recognition, 2010, 43, pp. 3507–3521
3. Nibaran Das, Ram Sarkar, Subhadip Basu, Punam K, Saha, Mahantapas Kundu, Mita Nasipuri. Handwritten Bangla Character Recognition Using a Soft Computing Paradigm Embedded in Two Pass Approach, Elsevier, Pattern Recognition, 2015, 48, pp. 2054–2071
4. Antony Robert Raj, M, Abirami, S. Offline Tamil Handwritten Character Recognition Using Statistical Features, AENSI Journals, Advances in Natural and Applied Sciences, 2015, Special 9, (6), pp. 367-374
5. Shyni, S, M, Antony Robert Raj, M, Abirami, S. Offline Tamil Handwritten Character Recognition Using Sub Line Direction and Bounding Box Techniques Indian Journal of Science and Technology, 2015, 8 (7), pp. 110–116
6. Rajashekararadhya, S, V, Vanaja Ranjan, P. Efficient Zone based Feature Extraction Algorithm for Handwritten Numeral Recognition of Four Popular South Indian Scripts, Int. Journal of Theoretical and Applied Information Technology, 2008, pp. 1171 – 1181
7. Chih-Wei Hsu, Chih-Jen Lin. A comparison of methods for multiclass support vector machines, IEEE Transactions on Neural Networks, 2002, 13, 2, pp. 415 – 425
8. Shanthi, N, Duraiswami, K. A Novel SVM -based Handwritten Tamil character recognition system, Springer, Pattern Analysis & Application, 2010, 13, 2, pp.173-180
9. Abdelhak Boukharouba, Abdelhak Bennia. Novel Feature Extraction Technique for the Recognition of Handwritten Digits, Elsevier, Applied Computing and Informatics, 2017, 13, pp. 19–26
10. Ritesh Sarkhel, Nibaran Das, Amit K, Saha, Mita Nasipuri. A Multi-Objective Approach Towards Cost Effective Isolated Handwritten Bangla Character and Digit Recognition, Elsevier, Pattern Recognition, 2016, 58, pp. 172–189
11. Antony Robert Raj, M, Abirami, S. Offline Tamil Handwritten Character Recognition Using Statistical Based Quad Tree, Australian Journal of Basic and Applied Sciences 2016, Special 10, (2), pp. 103-109
12. Sigappi, A.N, Palanivel, S, Ramalingam, V. Handwritten Document Retrieval System For Tamil Language, Int. Journal of Computer Application, 2011, pp. 31
13. Nikolett Bereczky, Amalia Duch, Krisztián Németh, Salvador Roura. Quad-K-D Trees, Theoretical Informatics, Springer, 2014, 616, pp. 743–754

14. Ming-Ke Zhou, Xu-Yao Zhang, Fei Yin, Cheng-Lin Liu. Discriminative Quadratic Feature Learning for Handwritten Chinese Character Recognition, Elsevier, Pattern Recognition, 2016, 49, pp. 7-18
15. Javad Sadri, Mohammad Reza Yeganehzad, Javad Saghi. A Novel Comprehensive Database for Offline Persian Handwriting Recognition, Elsevier, Pattern Recognition, 2016, 60, pp. 378–393
16. Jija Dasgupta, Kallol Bhattacharya, Bhabatosh Chanda. A Holistic Approach for Off-Line Handwritten Cursive Word Recognition Using Directional Feature Based on Arnold transform, Elsevier, Pattern Recognition Letters, 2016, 79, pp. 73-79
17. Cheng-Lin Liu, Ching Y, Suen. A new benchmark on there cognition of handwritten Bangla and Farsi numeral characters, Elsevier, Pattern Recognition, 2009, 42, pp.3287-3295
18. Partha Pratim Roy, Ayan Kumar Bhunia, Ayan Das. Prasenjit Dey, Umapada Pal, HMM-based Indic handwritten word recognition using zone segmentation, Elsevier, Pattern Recognition, 2016, 60, pp.1057-1075
19. Faouzi Zaiz, Mohamed Chaouki Babahenini, Abdelhamid Djeffal. Puzzle based system for improving Arabic handwriting recognition, Elsevier, Engineering Applications of Artificial Intelligence, 2016, 56, pp. 222–229
20. Georgios Vamvakas, Basilis Gatos, Stavros J, Perantonis. Handwritten character recognition through two-stage foreground sub-sampling, Elsevier, Pattern Recognition, 2010, 43, pp. 2807–2816
21. Suresh, R, M, Arumugam S. Fuzzy technique based recognition of handwritten characters, Elsevier, Image and Vision Computing, 2007, 25, 230–239
22. Parshuram M, Kamble, Ravinda S, Hegadi. Handwritten Marathi character recognition using R-HOG Feature, Elsevier, Procedia Computer Science, 2015, 45, 266 – 274
23. Xu Xiaobing, Wu Xiaoxu, Wang Jianping, Zhu Chenghui, Qiao Yuping. Recognition Research of Offline-Handwritten Chinese Character Based on Biomimetic Pattern, Elsevier. Procedia Engineering, 2011, 15, pp. 5116 – 5120
24. Subashini, A, Kodikara, N, D. A Novel Sife-Based Codebook Generation For Handwritten Tamil Character Recognition, 6th IEEE Int. Conf. on Industrial and Information Systems (ICIIS), 2011, 261 – 264
25. Rajashekararadhya, S, V, Vanaja Ranjan, P. Zone-Based Hybrid Feature Extraction Algorithm for Handwritten Numeral Recognition of two popular Indian Script, World Congress on Nature & Biologically Inspired Computing, 2009, pp. 526 – 530
26. Jagadeesh Kumar, R, Prabhakar, R, Suresh, R, M. Off-line Cursive Handwritten Tamil Characters Recognition, Int. Conf. on Security Technology, 2008, pp.159 – 164
27. Antony Robert Raj, M, Abirami, S. Strip Tree based offline Tamil Handwritten Character Recognition, Springer SIST, Int. Conf. on ICT for Intelligent Systems, 2015, pp. 367-374
28. Antony Robert Raj, M, Abirami, S. Hybrid Features based Offline Tamil Handwritten Character Recognition, 14th Int. Tamil Internet Conf, 2015, pp. 360-370
29. Antony Robert Raj, M, Abirami, S. **A Survey on Tamil Handwritten Character Recognition using OCR techniques,** The Second Int. Conf. on Computer Science Engineering and Applications (CCSEA), 2012, 05, pp. 115-127
30. Antony Robert Raj, M, Abirami, S. Analysis of Statistical Feature Extraction Approaches used in Tamil Handwritten *OCR, 12th Tamil Internet Conference- INFITT, 2013, pp*. 144-150
31. Antony Robert Raj, M, Abirami, S. Offline Tamil Handwritten Character Recognition using Chain Code and Zone Based Features *13th Tamil Internet Conference- INFITT, 2014, pp*. 28-34
32. Antony Robert Raj, M, Abirami, S, Murugappan, S, Baskaran, R, Prism Tree Shape Representation Based Recognition of Offline Tamil Handwritten Characters, Springer, Proceedings of the First Int. Conf. on Computational Intelligence and Informatics (ICCII), 2016, 507, pp. 457-470
33. Meenu Alexa, Smija Das. An Approach towards Malayalam Handwriting Recognition Using Dissimilar Classifiers, Elsevier, Procedia Technology, 2016, 25, pp. 224 – 231

கணினிவழித் தமிழ் கற்றல் மற்றும் கற்பித்தல் குறித்தான ஆய்வு தகவல் தொழில்நுட்பத்தின் துணையுடன் செய்தித் தயாரிப்பில் மாணவர்களை ஈடுபடுத்துவதன்வழி இருவழிக் கருத்துப்பரிமாற்றத் திறனை மேம்படுத்தல்

திருமதி பாஸ்கரன் கங்கா மூத்த ஆசிரியர், தமிழ்த்துறை, சுவா சூ காங் உயர்நிலைப் பள்ளி, சிங்கப்பூர்

திரு அந்தோணி ராஜ் ஜோசப், தமிழாசிரியர், தமிழ்த்துறை, சுவா சூ காங் உயர்நிலைப் பள்ளி

முன்னுரை

உள்ளங்கையில் உலகம் சுருங்கிவிட்டது என்று கூறும் அளவிற்குத் தகவல் தொழில்நுட்பம் நம்மை ஆளத் தொடங்கிவிட்டது மேலும்,-21 ஆம் நூற்றாண்டில் வாழும் மாணவர்களும் மின்னிலக்கவழிக் கற்போராக இருக்கின்றனர் .எனவே, தகவல் தொழில்நுட்பத்தையும் மாணவர்களையும் இணைத்துக் கல்வி கற்பிக்க வேண்டிய அவசியம் இன்றைய தாய்மொழி ஆசிரியர்களுக்கு உண்டு என்று கூறினால் அது மிகையன்று. சிங்கப்பூர்க் கல்வி அமைச்சு அனைத்துப் பாடங்களிலும் 21-ஆம் நூற்றாண்டுத் திறன்களை ஒருங்கிணைத்துக் கற்பிக்குமாறு ஆசிரியர்களை ஊக்குவிக்கிறது. மேலும் தொடக்கப்பள்ளியிலிருந்தே மாணவர்களின் எழுத்து வழி, பேச்சுவழி கருத்து பரிமாற்றத்த திறன்களை வளர்க்கும் வகையில் .முதல் பாடத்திட்டங்கள் வடிவமைக்கப்பட்டுள்ளன 2015

மாணவர்களின் தேவை, ஆர்வத்தின் அடிப்படையில் எங்கள் பள்ளியில் தாய்மொழித் துறை தகவல் தொடர்புத் தொழில்நுட்பத்தினை ஒரு முக்கிய கற்பித்தல் கருவியாகக்கொண்டு பேச்சுத்திறனை மேம்படுத்தும் முயற்சியில் இறங்கி வெற்றிக் கண்டது. கருத்துப் பரிமாற்றம், உடனிணைந்துக் செயலாற்றுதல், தகவல் திறன்கள் ஆகிய கூறுகளை உள்ளடக்கிய எங்கள் பாடங்களை வடிவமைத்தோம்.

சிக்கலும் நோக்கமும்:

நம் சிங்கையில் உயர்நிலைப் பள்ளியில் தமிழ்ப் பயிலும் மாணவர்களிடம் தமிழ்மொழி புழக்கம் குறைவாகவே உள்ளது. அதற்கு, மாணவர்கள் சிலரின் தாய்மொழித் தமிழ்மொழியாக இல்லாமல் இருப்பதும், கலப்புத் திருமணங்களின் விகிதம் அதிகரிப்பதும், வீட்டில் ஆங்கில மொழி புழக்கம் அதிகரித்து இருப்பதும் காரணங்களாக இருக்கலாம். எனவே, எங்கள் பள்ளியில் தாய்மொழியில் பேசச் சிரமப்படும் மாணவர்களின் ஆர்வத்தைத் தூண்டி தாய்மொழியில் அவர்களைச் சரளமாகப் பேச வைப்பதற்கான வழிமுறைகளை யோசித்தப்போது தகவல் தொழில் நுட்பம் அதற்கு ஒரு சிறந்த கருவியாக உதவும் என்பதை உணர்ந்தோம். தகவல் தொழில் நுட்பத்தை மொழி கற்பிக்க மட்டுமே பயன்படுத்துவதைவிட மாணவர்களை ஒரு செய்தி நிருபராக உருவாக்க இத்தகவல் தொழில் நுட்பம் உதவும் என்று எண்ணினோம். மேலும், இது மாணவர்களின் மொழிவளத்தைப் பெருக்குவதோடு மாணவர்கள் எழுத்துவழி, பேச்சுவழி தங்கள் கருத்துகளைத் தங்குதடையின்றித் தமிழில் வெளியிட துணையாக இருக்கும் என்று நம்பினோம். மாணவர்களைச் செய்தி நிருபராக மட்டுமின்றி ஒரு

செய்தியை எப்படி மக்களிடம் கொண்டு சேர்ப்பது, அதற்கான தரவுகளை எப்படி திரட்டுவது, காணொளி காட்சிகளை உருவாக்கும் வழிமுறைகள் யாவை என்பது பற்றியும் மாணவர்களுக்குக் கற்றுக் கொடுத்தோம்.

இதற்கு எங்களுக்குத் தமிழாசிரியர் பணித்திறன் மேம்பாட்டகத்தின் பேச்சுவழிக் கருத்துப்பரிமாற்றத் திறன் மேம்பாட்டிற்கான சட்டகமும் (1 பின்னிணைப்பு) கல்வி தொழில் நுட்பப் பிரிவின் கற்பித்தலியல் சார்ந்த ஆதரவு சட்டகமும் (2 பின்னிணைப்பு) பெரிதும் துணையாக அமைந்தன.

செய்தித் தயாரிப்பில் மாணவர்களை அனுபவப்பூர்வமாக ஈடுபடுத்தி அவர்களின் இருவழிக் கருத்துப்பரிமாற்றத் திறனை மேம்படுத்தல் – இதுவே எங்கள் ஆய்வின் தலைப்பு.

பிரச்சினை அடிப்படையில் நிகழும் கற்றல் அணுகுமுறையில் பாடத்தை ஆரம்பிப்பதற்குமுன் கவனத்திற்கொள்ள வேண்டியவை.

- மாணவர்களிடம் ஒரு பிரச்சினையை முன் வைத்தல்.
- பிரச்சனை, உலக நடப்புகளை ஒட்டியும் அவர்களை ஈர்ப்பதாகவும் இருக்கும் வகையில் அமைத்தல்.
- கற்றல் தேவைகளுக்கும் பண்பியல்புகளுக்கும் தொடர்புடையதாக இருத்தல்.

மாணவர்வகளிடம் ஒரு பிரச்சினையைத் தருவதுடன் அவர்களுக்கு உதவ மொழி ஆதரவு**,** பாடப்பொருள் ஆதரவு**,** உத்திமுறை ஆதரவு போன்றவற்றை வழங்கினோம் .

கற்றலைத் தூண்டுதல்

- மாணவர்கள், தங்கள் கூகுள் வகுப்பில் சில கருப்பொருள்களைக் கொடுத்து அவற்றுடன் தொடர்புடைய பிரச்சினைகள் குறித்து மாணவர்களைக் குழுநிலையில் கலந்துரையாடச் செய்தோம். கருப்பொருளையொட்டிச் சில தலைப்புகளைத் தேர்ந்தெடுத்து அவற்றைக் குறித்த தரவுகளைச் சேகரிக்க மாணவர்களை ஊக்கப்படுத்தினோம்.

சிந்தித்தலையும் கலந்துரையாடலையும் ஊக்குவித்து வளர்த்தல்

- மாணவர்கள் கருப்பொருள் அடிப்படையில் சில தலைப்புகளைத் தேர்ந்தெடுத்தனர், பின்னர் அத்தலைப்புக் குறித்த பிரச்சினைகள், அப்பிரச்சினை எதனால் உருவாகின்றது போன்ற விவரங்களை இணையப் பக்கத்திற்குச் சென்று சேகரித்தனர்தங்கள் கூகுள் .பின்னர் . வகுப்பில் தங்கள் குழுவினருடன் பகிர்ந்துகொண்டனர். தங்கள் கருத்துகளையொட்டி விவாதித்து மதிப்பீடுகளையும் வழங்கினர். .1 தலைப்புகள்) கைத்தொலைபேசியில் அதிக நேரம் ஈடுபடுவதால் விளையும் தீமைகள், .2 இணைய மிரட்டல், .3 உடல் எடையை அளவுக்கதிக்கமாகக் குறைப்பது,4 . பிள்ளைகளிடம் கல்வி முன்னேற்றம் குறித்த பெற்றோர்களின் அதிக எதிர்பார்ப்பு, .5 வீட்டுப்பாடங்கள் அதிகம் எண்ணி மன உளைச்சல் அடைதல்)

- ஆசிரியர்களும் மாணவர்களின் கருத்துகளையொட்டி தங்கள் கருத்துகளையும் மதிப்பீடுகளையும் கூகுள் வகுப்பறையில் பகிர்ந்துகொண்டனர். இதன்மூலம் மாணவர்கள் பிரச்சனையை ஆராய்ந்து அதற்கான தீர்வுகளையும் அறிய வாய்ப்பு அளிக்கப்பட்டன .

கற்றலின் வெளிப்பாட்டினை வழிநடத்துதல்

- மாணவர்களிடம் நிருபர்களின் பணியைக் குறித்த தகவல்களைப் பகிர்ந்துகொண்டோம். நிருபர் என்பவர் யார், அவர் எப்போதும் விழிப்பு நிலையில் இருக்க வேண்டியதன் அவசியம் என்ன, செய்திகளின் நம்பகத்தன்மையை எவ்வாறு அறிவது, செய்தியை மக்களிடம் கொண்டு சேர்க்கும்போது எவற்றை கவனத்தில் கொள்வது என மாணவர்களுக்கு வகுப்பில் விளக்கினோம்.
- மாணவர்கள் தாங்கள் தொகுத்த கருத்துகளின் அடிப்படையில் அவற்றை எவ்வாறு செய்தியாக வழங்குவது என்று திட்டமிட்டதுடன் தாங்கள் பாகமேற்றுள்ள பாத்திரங்களுக்கேற்ற வசனங்களைக் கூகுள் வகுப்பறையில் தட்டச்சுச் செய்தனர்குழு . உறுப்பினர்கள் தங்கள் ஆலோசனைகளை அங்கு வழங்கியதுடன் திருத்தங்களையும் செய்தனர்.
- அத்துடன் செய்திக்கு வலுச் சேர்க்கும் வகையில் எப்படி ஒளிக்காட்சியை அமைப்பது, எந்தக் கோணங்களில் இருந்து காட்சிகளை ஒளிப்பதிவு செய்வது, ஒளிக்காட்சியின்போது அதில் இடம்பெற வேண்டிய முக்கிய கூறுகள் ஆகியவற்றையும் நாங்கள் கற்றுக் கொடுத்தோம்.
- மாணவர்களைக் கணினிக்கூடத்திற்கு அழைத்துச் சென்று ஒளிப்பதிவு செய்த பிறகு அக்காட்சிகளை எப்படி வெட்டுவது, பெயர்களை இணைப்பது, பதிவு செய்த குரல்களைச் சேர்ப்பது போன்ற தொழில் நுட்பக்கூறுகளையும் கற்றுக் கொடுத்தோம். இவை அனைத்தும் ஆரம்ப நிலையிலேயே கற்றுக் கொடுப்பது அவசியமாகும். அப்போதுதான் மாணவர்கள் தாங்கள் செய்யவிருக்கும் திட்டவேலை குறித்த முழுமையான அறிவைப் பெறுவர்.
- இறுதியில் குழுவில் இருந்து ஒருவர் செய்தி நிருபராகவும் மற்றொருவர் அச்செய்தியுடன் தொடர்புடைய மாந்தராகவும், வேறொருவர் அக்காட்சிகளை ஒளிப்பதிவு செய்பவராகவும் தாங்கள் தேர்ந்தெடுத்த பாகத்திற்கேற்ப கொடுக்கப்பட்ட ஒப்படைப்பை வெற்றிக்கரமாகச் செய்து முடித்தனர்.

கண்காணிப்பும் கருத்துரைப்பும்

- மாணவர்கள் வகுப்பறையில் தங்களுடைய காணொளி காட்சியைப் ஒளிப்பரப்பியதுடன் இந்தத் திட்டவேலை தொடங்கியது முதல் இறுதி வரை தங்களுக்கு ஏற்பட்ட அனுபவங்களை சக மாணவர்களுடன் பகிர்ந்துகொண்டனர்.
- மாணவர்கள் தகவல் தொடர்புத் தொழில்நுட்பத்தைப் பயன்படுத்திய முறை, அதன்வழி கருத்துகளைப் பரிமாறிக்கொண்ட வழிமுறைகள், பேசும் திறன், தனி நிலை, இணை நிலையில் அவரகளின் பங்களிப்பு ஆகியவற்றின் அடிப்படையில் ஆசிரியர்கள், மதிப்பெண்கள் வழங்கினார்கள்.

திட்டவேலையில் பங்கு பெற்ற மாணவர்கள் பற்றிய விவரங்கள்

- உயர்நிலை இரண்டில் பயின்ற மாணவர்கள்
- பதிமூன்று வயது நிரம்பிய மாணவர்கள்
- உயர்தமிழ், விரைவு, வழக்க நிலையில் பயின்ற மாணவர்கள் 25

திட்டவேலைக்கான கால வரம்பு

- தவணை இரண்டு – பத்து வாரங்கள்
- வாரம் இரண்டு வகுப்புகள் – இரண்டு மணி நேரம்

மாணவர்கள் எடுத்துக்கொண்ட தலைப்புகள்

இளையர்கள் எதிர்நோக்கும் பிரச்சினைகள்

- கைத்தொலைப்பேசியை அதிகம் பயன்படுத்துதல்
- மன அழுத்தம்
- வன்முறை
- உடல் பருமன்
- நண்பர்களின் பேச்சைக் கேட்பது
- குழந்தை வன்முறை
- பிள்ளைகளிடம் பெற்றோர்களின் அளவுகடந்த எதிர்பார்ப்பு
- கல்வி

மாணவர்கள் தங்கள் திட்டவேலைக்காக கூகுள் இணையப் பக்கம், வலைப்பூ, யூடியுப் ஆகிய பக்கங்களில் கருத்துகளைத் திரட்டினர்ஆங்கிலம் ., தமிழ் என இருமொழிகளிலும் அவர்கள் இணையத்தில் தங்கள் திட்டவேலைக்கான கருத்துகளைத் தேடித் தொகுத்தனர்.

மாணவர்களின் கருத்துகள்

1.1 திட்ட வேலை குறித்து என்ன நினைக்கிறீர்கள்? திட்டவேலையில் உங்களுக்குப் பிடித்த கூறுகள் குறித்து உங்கள் கருத்துகளைக் கூறவும்.

1) குழு நிலையில் இணைந்து செயலாற்றுவது, வகுப்பு மாணவர்களுடன் கலந்துரையாடுவது பயன்மிக்கதாக இருந்தது.
2) பிரச்சினைக்குரிய தலைப்பைத் தேர்ந்தெடுக்கும் சுதந்திரம் தந்தது மகிழ்ச்சியைத் தந்தது.
3) கொடுக்கப்பட்ட தகுதிநிலை விளக்கக் குறிப்புகள் திட்டவேலையை எளிதாகச் செய்ய உதவின.
4) ஒளிக்காட்சியைப் பதிவு செய்வது, தேவையில்லா காட்சிகளை நீக்குவது, ஒலிப்பதிவு செய்த குரலை இணைப்பது போன்ற நடவடிக்கைகள் புத்துணர்ச்சி ஊட்டின.
5) ஒவ்வொருவரின் கற்கும் முறை, தனித்துச் செயல்படும் திறன் பற்றி அறியமுடிந்தது.
6) தன்னம்பிக்கையுடன் சரளமாகப் பேச உதவியதுடன் ஏதோ சாதித்த உணர்வை வழங்கியது.
7) திட்ட வேலையைக் கொடுத்த காலக்கெடுவிற்குள் முடிப்பது சாத்தியமா என்ற வினா எழுந்தாலும் முயன்றால் முடியும் என்ற உறுதியை உண்டாக்கியது.

8) ஒளிப்பதிவு செய்வது, நடிப்பது, காணொளி தயாரிப்பது என அனைவருக்கும் திட்டவேலையில் பங்களிக்க வாய்ப்பளிக்கப்பட்டது,

.திட்டவேலையில் உங்களுக்குப் பிடிக்காத கூறுகள் குறித்து உங்கள் கருத்துகளைக் கூறவும் 1.2

1) செய்தியை ஒளிப்பதிவு செய்வது போல் இல்லாமல் செய்தி வாசிப்பது போல் அமைந்திருக்கலாம் .

.2.1இந்த அணுகுமுறை பிடித்ததற்கான காரணங்களைக் கூறவும்

1) இந்தத் திட்டவேலை வித்தியாசமான கற்றல் அணுகுமுறையைக் கொண்டிருந்தது.
2) வழக்கமான திட்டவேலையிலிருந்து மாறுபட்டு இருந்ததால் எங்களை மிகவும் கவர்ந்தது.
3) திட்டவேலை முதலில் கடினமாக இருந்தாலும் ஆசிரியர்கள் நன்கு விளக்கியதால் சிறப்பாக முடித்தோம்.
4) வசனங்கள் எழுத, ஒளிக்காட்சித் தயாரிக்கக் கற்றுக்கொண்டோம்.
5) ஒரு பிரச்சினையை ஒரே கோணத்தில் பார்க்காமல் பல்வேறு கோணத்தில் பார்க்கக் கற்றுக் கொண்டோம்.
6) குழுக்களாகச் செயல்பட்டதால் மொழி வளம், சொல் வளம் பெருகியது.
7) சுய முனைப்புடன் கற்கும் திறன் பெருகியது.

2.2 இந்த அணுகுமுறை பிடிக்காததற்கான காரணங்களைக் கூறவும்.

1) நேரம் அதிகம் எடுத்துக் கொண்டது.

2.3இருவழி கருத்துப் பரிமாற்றத்திற்காகச் செய்தி தயாரிப்பு என்ற உத்தியைப் பயன்படுத்தியதன் நன்மைகளாக நீங்கள் கருதுபவை.

1) தமிழ்மொழியில் கருத்துகளை வெளிப்படுத்த உதவியது.
2) ஒளிப்பதிவு செய்தல், ஒளிக்காட்சி தயாரித்தல் போன்ற தொழில்நுட்பத் திறன்களைக் கற்றுக் கொண்டோம்.
3) குழுவாக இணைந்து பணியாற்றக் கற்றுக் கொண்டோம்.
4) பாத்திரங்களாகப் பாகமேற்று நடிக்கும்போது அப்பாத்திரமாகவே உணர்ந்து நடிக்கக் கற்றுக் கொண்டோம்.
5) கூடிக்கற்றல் அணுகுமுறை, சுய முனைப்புடன் கற்றல் ஆகியவற்றின் அவசியத்தை உணர்ந்தோம்.
6) மற்றவர்களின் கருத்துக்கும் முக்கியத்துவம் தர வேண்டும் என்று கற்றுக்கொண்டோம்.
7) ஒரு செய்தி நிருபராக முதன்முதலில் பணியாற்றும் அனுபவம் பெற்றோம்.

2.4 இந்தத் திட்டவேலையை மேற்கொள்ளும்போது எதிர்கொண்ட சிரமங்களைக் கூறவும்.

1) தொழில்நுட்பத்திறன் அதிகம் இல்லாததால் ஒளிப்பதிவு செய்வதில் சிரமம் உண்டானது.

2) நேர நிர்வாகத்தில் சிரமம் ஏற்பட்டது.
3) குழுவில் சில உறுப்பினர்கள் அதிகம் பங்களிக்காதது சிரமத்தை உண்டாக்கியது.
4) ஒளிப்பதிவு செய்யும்போது தடங்களும் தடுமாற்றமும் ஏற்பட்டன.
5) குழுவினரிடையே புரிந்துணர்வு ஏற்பட தாமதமானது.

மாணவர்களிடம் ஆசிரியர்கள் கண்ட மேம்பாடு

மேற்குறிப்பிட்ட நடவடிக்கைகளின் மூலம் மாணவர்கள்...

- பல புதிய தகவல்களைப் பெற்றனர்.
- தங்கள் சொல்வளத்தைப் பெருக்கிக்கொண்டனர்.
- தன்னம்பிக்கையுடன் பேசினர்.
- பிரச்சனைகள் குறித்து நுட்பமாக ஆராய்ந்து அதற்கான தீர்வு காணும் திறன் பெற்றனர்.
- குறும்படங்கள் தயாரிக்கும் திறன் பெற்றனர்.
- மாணவர்கள் வாய்மொழித் தேர்வைத் திறம்பட அணுகினர்.
- கருத்து வளம் பெருகியதால் மாணவர்கள் கட்டுரைப் பாடத்தின்போது கருத்துகளைத் திறம்படத் தொகுத்து எழுதினர்.

பிரச்சினைகளை களையும் முறை

1) மாணவர்கள் திட்டவேலையின் தொடக்கத்திலேயே தங்கள் பிரச்சினைகள் குறித்து ஆசிரியருடன் பகிர்ந்துகொள்ளலாம்.
2) மாணவர்கள் அவரவர் திறன் அறிந்து தங்கள் பணிகளைப் பிரித்துக் கொள்ளலாம்.
3) மாணவர்கள் ஒருவரையொருவர் புரிந்துகொண்டு மற்றவர்கள் தங்கள் பணிகளைச் செய்ய சிரமப்படும்போது தாங்களாகவே முன்வந்து உதவலாம் .

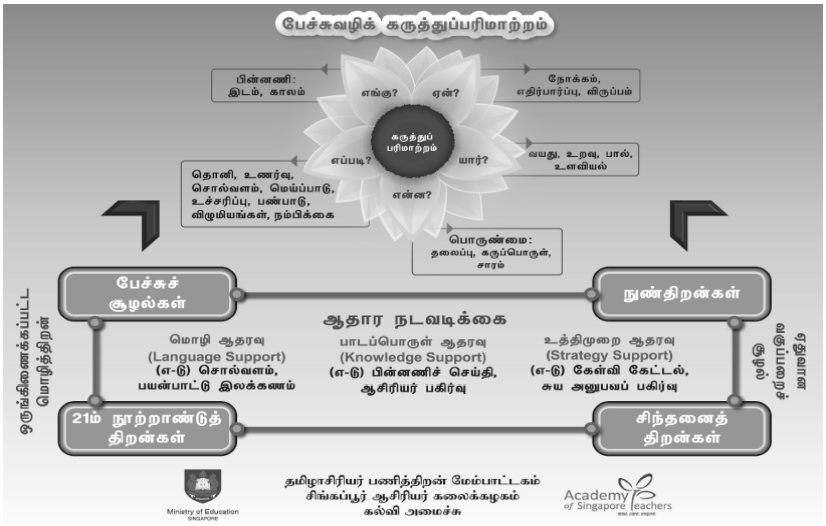
பரிந்துரைகள்

1) மாணவர்களை சமூக நிகழ்வுகளுக்கு அழைத்துச் சென்று நேரடியாக நிகழ்ச்சிகளை ஒளிப்பதிவு செய்து அது குறித்த ஒளிக்காட்சிகளைத் தயாரிக்க ஊக்கப்படுத்தலாம்.
2) செய்தி நிருபர்களை வரவழைத்து அவர்களின் அனுபவங்களைப் பகிர்ந்துகொள்ள செய்யலாம். இதன்மூலம் மாணவர்கள் செய்தி நிருபர்களின் பணி குறித்த அதிகப்படியான தகவல்களை அறிந்துகொள்வர்.
3) தொலைக்காட்சி அலைவரிசை நிறுவனத் தலைவருடன் கலந்தாலோசித்து மாணவர்களுக்குச் செய்தி நிருபருடன் இணைந்து பணியாற்ற வாய்ப்பு நல்கலாம்.

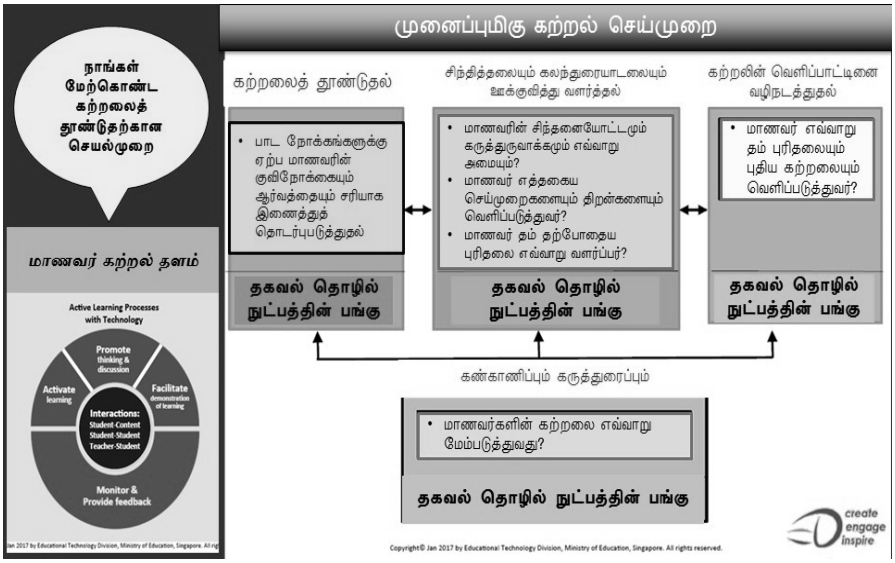
முடிவுரை

மக்களுடன் நேர்காணல் நிகழ்த்துவது மாணவர்கள் பேசும் திறனை வகுப்பில் மட்டுமல்லாது வெளிப்புறத்திலும் நடத்துவதற்கான வாய்ப்பை வழங்குகிறது, மேலும் அவர்களைச் சமூகமயமாக்கலில் உதவுகிறது .நேர்காணல்களுக்குப் பிறகு, ஒவ்வொரு மாணவரும் மிகவும் சுவாரஸ்யமான செய்திகளைக் கண்டறிந்து தங்கள் நண்பர்களுக்குத் தெரிவிக்க முடியும் என்ற கருத்தும் எங்கள் ஆய்வுக்கு வலு சேர்த்தது) .கேய், 2006). இதற்குத் திட்ட வேலையின் ஆர்வத்தோடு ஈடுபட்ட எங்கள் மாணவர்களின் பங்களிப்பே சான்றாகும்எனவே ., மாணவர்கள் தமிழ்மொழி கற்றல் கற்பித்தலில் ஆர்வத்துடன் ஈடுபட தகவல் தொடர்புத் தொழில்நுட்பம் பெரிதும் துணைபுரிகிறது என்பது எங்கள் ஆய்வின் கண்கூடான முடிவாகும்.

பின்னிணைப்பு 1



பின்னிணைப்பு 2



துணை நூல்கள்

1. தமிழ் மொழிப் பாடத்திட்ட வழிக்காட்டி 2011, உயர்நிலைப் பள்ளி, தொடக்கநிலைப் பள்ளி, சிங்கப்பூர் கல்வி அமைச்சு.
2. தாய்மொழிகள் மறுஆய்வுக் குழு அறிக்கை 2010, சிங்கப்பூர் கல்வி அமைச்சு.
3. தமிழாசிரியர் பணித்திறன் மேம்பாட்டகம், சிங்கப்பூர் ஆசிரியர் கலைக்கழகம், கல்வி அமைச்சு.

4. Educational Technology Division 2017, Ministry of Singapore.
5. Teaching Communication Matters: Importance, Methods and Challenges of Teaching Communication Skills Dr Linda M. Perry Communications and New Media Programme Faculty of Arts and Social Sciences Centre for Development of Teaching and Learning CDTL Brief / October/November 2010, Page 7
6. Staab, C. 1992. Oral language for today's classroom. Markham, ON: Pippin Publishing.
7. Teaching Speaking: Activities to Promote Speaking in a Second Language Hayriye Kayi

நன்றி

• முதல்வர், துணை முதல்வர்கள், தாய்மொழித் துறை ஆசிரியர்கள், மாணவர்கள், பெற்றோர்கள் - சுவா சூ காங் உயர்நிலைப் பள்ளி. சிங்கப்பூர்

• திருமதி மஞ்சுளா, முதன்மை ஆசிரியர், சிங்கப்பூர் ஆசிரியர் கலைக்கழகம், கல்வி அமைச்சு.

பட்டுப்புழு வளர்ப்பு விவசாயிகளின் வருமானத்தைப் பெருக்க தமிழ் வழி இணையதள பல்லூடக மின்கற்றல் கட்டமைப்பு

சரவணன் ரூபதர்ஷினி[1], நடராஜன் ராஜா[2] அழகேசன் தங்கமலர்[3] மற்றும்
செந்தா வெங்கடாச்சாரி கிருஷ்ணமூர்த்தி[4]
1-4ம் ஆண்டு இளம்அறிவியல் (பட்டுப்புழுவியல்) மாணவி,
2-துணைப் பேராசிரியர் (கணிப்பொறி அறிவியல்),
3-உதவியாசிரியர் (பட்டுப்புழுவியல்),
4-பேராசிரியர் மற்றும் தலைவர், பட்டுப்புழுவியல் துறை
வனக்கல்லூரி மற்றும் ஆராய்ச்சி நிலையம்,
தமிழ்நாடு வேளாண்மை பல்கலைக்கழகம், மேட்டுப்பாளையம்.

ஆய்வுச்சுருக்கம்

இன்றைய சமூகம் அடிப்படை கற்றல் முறையிலிருந்து மாறி இணைய வழி கற்றலில் பரிணமிக்கிறது. மேற்படி கற்றல் முறையில் பள்ளிப்பாடம் முதல் அறிவுசார் ஆராய்ச்சி கற்றல் வரை இணையத்தின் பயன்பாடு பெருகி வருகிறது. பெரும்பாலான விவசாயிகள் இணையத்தள மற்றும் கைபேசி சார்ந்த செயலிகள் வாயிலாக வேளாண் தொழில் நுட்பங்களை புகைப்படம் மற்றும் காணொளிக்காட்சிகள் மூலம் சுயமாக கற்று நடைமுறைப்படுத்தி வேளாண் மற்றும் வேளாண் சார்ந்த தொழில்களில் இலாபம் ஈட்ட முடியும். இவ்விணையச் செயலித் தொகுப்பு தமிழ் வழியில் பட்டுப்புழு வளர்ப்பு சார்ந்த குறுந்தகவல்கள், புகைப்படம் மற்றும் காணொளிக்காட்சிகளை உள்ளடக்கி கட்டமைக்கப்பட்டுள்ளதால் எளிதில் விவசாயிகள் மற்றும் பட்டு உற்பத்தி தொழில் முனைவோருக்கு செயல்முறைப்படுத்தும் வண்ணம் அமைகிறது. இம்மின்-கற்றல் தொகுப்பில் பட்டுப்புழு வளர்ப்பின் சாராம்சங்கள் முழுமையாக தொகுக்கப்பட்டுள்ளன. தரமான வெண்பட்டு உற்பத்தியில் தமிழ்நாடு முதன்மை பெற்று விளங்குகிறது. தமிழ்நாடு பட்டுப்புழு வளர்ப்பு விவசாயிகள் அதிக அளவில் இரட்டை கலப்பினமான கிருஷ்ணராஜா (சி.எஸ்ஆர்2 X சிஎஸ்ஆர்.27) X (சிஎஸ்.ஆர்.6 X சிஎஸ்.ஆர்.26) இரகத்தினை வளர்த்து வருகின்றனர். பட்டுப்புழு வளர்ப்பினை, இளம்புழு வளர்ப்பு மற்றும் வளர்ந்த புழு வளர்ப்பு என்று இரண்டு நிலைகளாக உற்பத்தி மேற்கொண்டு வருகின்றனர். அவற்றில் முதல் இரண்டு புழு பருவங்களை (10 நாட்கள்) வளர்ப்பு மேற்கொள்வது "இளம்புழு வளர்ப்பு" எனப்படுகிறது. இதனை மேற்கொள்ளும் விவசாயிகள் இளம்புழு வளர்ப்பு விவசாயிகள் என அழைக்கப்படுகின்றனர். இதற்கு 13 அடி உயரமுள்ள 1500 சதுரடி அளவுள்ள முழு வளர்ப்பு அறை தேவைப்படுகிறது. மேலும் 1-1.25 ஏக்கர் அளவுள்ள முசுக்கொட்டை பயிர் தோட்டத்திலிருந்து 5000 நோயற்ற முட்டைத் தொகுதியாக வளர்க்கின்றனர். இதனை மாதத்திற்கு 3 தவணைகளாக வருடத்திற்கு மொத்தம் 36 தவணைகளாக இளம்புழு வளர்க்கப்பட்டு முதிர்ந்த புழு வளர்ப்பு விவசாயிகளுக்கு விநியோகிக்கப்பட்டு வருகிறது. இவ்வாறு வருடத்திற்கு 60,000 முட்டைத் தொகுதி இளம்புழுக்கள் 9000 விவசாயிகளுக்கு வழங்கப்படுகின்றன. பட்டுப்புழு வளர்ப்பில் வருமானமாக ரூ.50,000 ஈட்டுகிறார்கள். மேலும் அவருடைய வரவு செலவினை விகிதம் முறையே 1:1.8 என்ற அளவில் கிடைக்கிறது. மேலும் வருடத்திற்கு 420

நபர்களுக்கு வேலைவாய்ப்பினை அளிக்கிறது. இரண்டாவது நிலையாக மூன்றாம் புழு பருவத்திலிருந்து பட்டுக்கூடு உற்பத்தி மேற்கொள்வதனை வளர்ந்த புழு வளர்ப்பு எனப்படுகிறது. இதனை மேற்கொள்ளும் விவசாயிகளை வளர்ந்த புழு வளர்ப்பு விவசாயிகள் என அழைக்கப்படுக்கின்றனர். இதற்கு 13 அடி உயரமுள்ள 1000 சதுரடி அளவுள்ள வளர்ப்பு அறை தேவைப்படுகிறது. பட்டுப்புழு வளர்ப்பில் ஈடுபாடுடைய மூன்று குடும்ப உறுப்பினர்கள் கொண்ட விவசாயி மாத வருமானமாக ரூ.30,000 ஈட்டுகின்றனர். மேலும் அவருடைய வரவு செலவு விகிதம் முறையே 1:3 முறை அளவில் கிடைக்கிறது. இதன் மூலம் வருடத்திற்கு 360 நபர்களுக்கு வேலை வாய்ப்பினை அளிக்கிறது. இதனோடு இளம்புழு மற்றும் வளர்ந்த புழு வளர்ப்பு விவசாயிகள் பட்டுப்புழுக் கழிவு, மண்புழு உரம், கோழி, ஆடு மற்றும் மாடு வளர்ப்புகளை மேற்கொள்வதன் மூலம் பட்டுப்புழு வளர்ப்பு கழிவுகளை திறம்பட சுழற்சி முறையில் மேலாண்மை செய்து செலவுகளை குறைத்து 10-35% அதிகமாக இலாபம் ஈட்டுகின்றனர்.

சுருக்கச்சொற்கள்: *முசுக்கொட்டை பயிர், பட்டுப்புழு வளர்ப்பு, பல்லூடகம், இணையதள மென்பொருள் கட்டமைப்பு, தமிழ்வழி மின்-கற்றல், மற்றும் கூகுள் இணையதளம்.*

1.அறிமுகம்

வேளாண்மை, அறிவியல், கல்வி, பொருளாதாரம் ,மருத்துவம், விளையாட்டு போன்ற பல்வேறு சமூக மேம்பாட்டு துறைகளின் வளர்ச்சிக்கு தமிழ்வழி இணையதள பல்லூடக மின்கற்றல் கட்டமைப்பு கற்றல் முறை மிகவும் உறுதுணையாகவும், மலிவாகவும், எளிதில் நம் கையடக்கத்தில் பெற்றுக் கொள்ள ஏதுவாகவும் விளங்குகிறது .

அந்த வகையில் பட்டுப்புழு வளர்ப்பு சார்ந்த சிறந்த வேளாண் தொழில்நுட்பங்களை புகைப்படங்கள், காணொளிக்காட்சிகள், தகவல்கள், விளக்கங்கள் மற்றும் தகவல் பரிமாற்றங்கள் வாயிலாக விவசாயிகளுக்கு மிகவும் எளிதாகவும், துரிதமாகவும் கற்று புரிந்து கொள்ளும் வகையிலும் தமிழ் வழி இணையதள பல்லூடக மின்கற்றல் கட்டமைப்பு முறை அமைக்கப்பட்டுள்ளது.

மேற்படி இந்த கூகுள் தமிழ் வழி இணையதள பல்லூடக மின்கற்றல் கட்டமைப்பு இணையதள புதிய பரிமாண வசதியின் முதன்மை பக்கம் முசுக்கொட்டை பயிரிடுதல், இளம்புழு வளர்ப்பு, மற்றும் முதிர்புழு வளர்ப்பு, ஆகிய வலைப்பக்கங்களை கொண்டுள்ளது. பயனாளியின் தகவல்களை பதிவு செய்ய கூகுள் மின் படிவம் பயன்படுத்தப்பட்டுள்ளது. மேலும் முதன்மை பக்கத்திலிருந்து எந்தவொரு வலைப்பக்கத்தையும் சொடுக்கி உள்ளே செல்லமுடியும். அவ்வாறு உள்சென்று மேற்படி பட்டுப்புழுவியல் தலைப்பு சார்ந்த தகவல்களைப் பெறமுடியும்.

பட்டு உற்பத்தி மற்றும் பயன்பாடு பல நூற்றாண்டுகளுக்கு முன்பிருந்தே நமது கலாச்சாரத்துடன் ஒன்றிணைந்து உள்ளது. அதன் தேவை அதிகமாக இருந்தாலும் உற்பத்தி குறைவாகவே உள்ளது . உற்பத்தியை அதிகரிக்க பல முறைகள் இருந்தாலும் அவை விவசாயிகளை சரியான நேரத்தில் அடைவதில்லை, மின் கற்றல் மூலம் பட்டுப்புழு வளர்ப்பு பற்றிய தெளிவான தகவல்கள், குறிப்புகள் செயல் முறைகள், பரிந்துரைகள் ஆகியவை விவசாயிகளுக்கு உடனுக்குடன் கிடைக்கிறது . அது மட்டுமில்லாது கற்போருக்கு கற்றலை

எளிதாக்குவதோடு ஆர்வத்தையும் தூண்டுகிறது, விவசாயிகள் மற்றும் தொழில் முனைவோர்களுக்கும் கற்றலை எளிதாக்குவதோடு புது தொழிற் நுட்பங்களை தெரிந்து கொண்டு ஆராயவும் பயன்படுகிறது.

2. இலக்கிய கணக்கெடுப்பு

அரேபியாவில் கல்லூரி பயிலும் மாணவர்கள் பல்கலைக்கழங்களில் சேர்ந்து பயிலும் வாய்ப்பினை இணைய தளம் வாயிலாக உயர்கல்வி கற்கும் வாய்ப்பினை பெற முடிகிறது[1]. மேலும் ரசியாவில் அறிவியல் ஒலிம்பியாட் பயிற்சி[2], நீரழிவு நோய் பற்றிய விழிப்பு உணர்வு கல்வி [3] மற்றும் சுற்றுசூழல் பற்றிய அறிவியல் [4] போன்ற பல்வேறு தகவல்களை இணைய தள வாயிலாக கற்று வருகின்றனர் . இந்தியாவில் இணைய பயனாளர்களின் எண்ணிக்கை 500 மில்லியனிலிருந்து 2019-ல் 627 மில்லியனாக வேகமாக அதிகரித்துள்ளது.

3. தற்போதுள்ள அடிப்படை கற்றல் அமைப்பின் சவால்கள்

அடிப்படை கற்றல் முறையானது குறிப்பிட்ட புவியியல் மற்றும் புலம் சார்ந்த கற்றல் ஆர்வலர்களை மட்டுமே கற்பதற்கு வழி கோலுகின்றது. மேலும் பாடத்திட்டம் கட்டமைப்பு, கால மேலாண்மை, கற்கும் எளிமை மற்றும் மொழி-கலாச்சார தடைகளால் உலகளாவியளவில் கற்றலை செயல்முறைப்படுத்துவதில் சிரமங்களைக் கொண்டுள்ளது.

4. முன்மொழியப்பட்ட முறை - கூகுள் தமிழ் வழி இணையதள பல்லூடக மின்கற்றல் கட்டமைப்பு

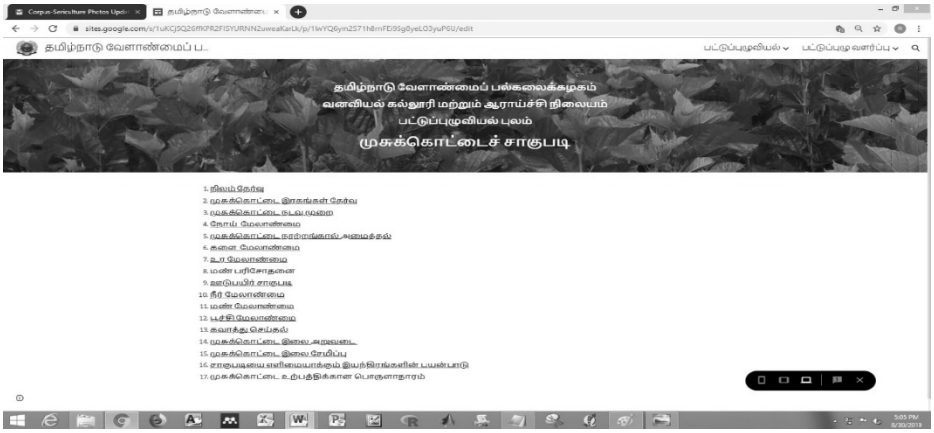
மின்-கற்றல் சுற்றுச்சூழல் அமைப்புகளின் சரியான காரணிகளையும், செயல்படுத்தலையும் ஆதரிக்க ஒரு கட்டமைப்பு முறையை முறைப்படுத்துவதாகும். இதற்கு கூகுள் இணைய தள வசதி ஒரு சிறந்த உதாரணம் ஆகும். கூகுள் இணைய தளத்தில் மேற்படி பாடத் தரவுகளான புகைப்படங்கள், காணொளிக்காட்சிகள் , தகவல்கள், விளக்கங்கள், தகவல் பரிமாற்றங்கள் மற்றும் எழுத்து வடிவமாக நிர்ணயிக்கப்பட்டு கற்போருக்கு ஏதுவாக அமைக்கப்பட்டுள்ளது .

மேலும் இந்த தமிழ் வழி இணையதள பல்லூடக மின்கற்றல் கட்டமைப்பு இணைய வசதியை https://sites.google.com/s/1uKCj5Q26ffKPR2FISYURNN2uweaKarLk/p/1lwYQ6ym2S71h8rnFEi9Sg8yeLO3yuP6U/edit என்ற இணைய முகவரியில் சொடுக்கி பயன்படுத்த முடியும்.

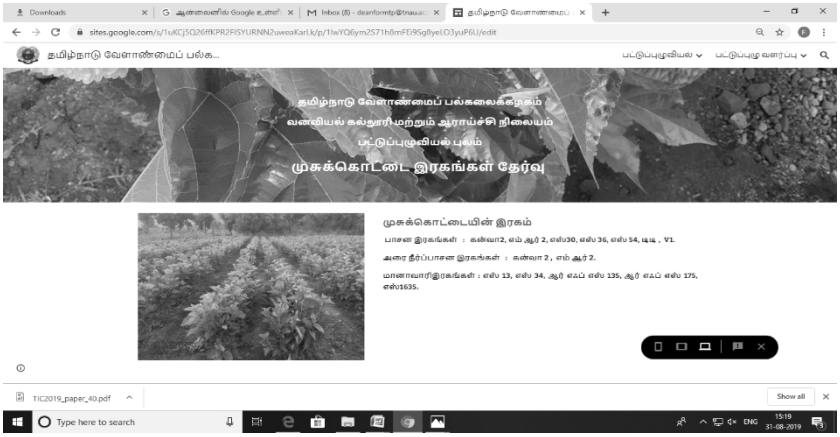
படம். 1 முகப்பு பக்கம்

5. முடிவுகளின் விவாதம்

தமிழ் வழி இணையதள பல்லூடக மின்கற்றல் கட்டமைப்பின் முதல் பக்கம் பட்டுப்புழு வளர்ப்பின் பல்வேறு கருத்து வலைப்பக்கங்களை காட்டுகிறது. அதில் முக்கியமான முசுக்கொட்டை வளர்ப்பு, பட்டுப்புழு வளர்ப்பு ஆகியவை தனித்தனியாய் தரப்பட்டுள்ளது.

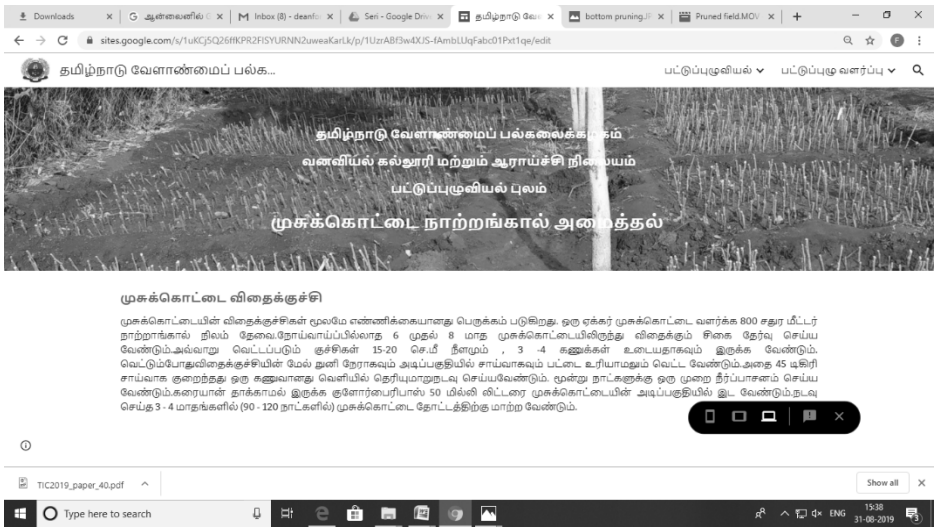


படம்.2 முசுக்கொட்டை சாகுபடி



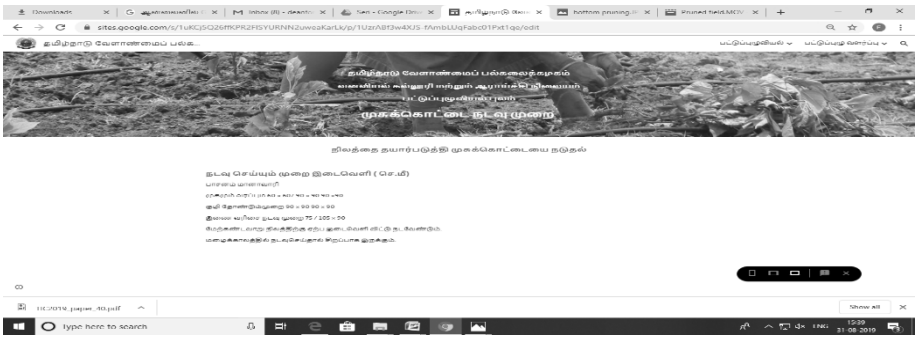
படம்.3 முசுக்கொட்டை இரகங்கள் தேர்வு

முசுக்கொட்டையில் பல்வேறு ரகங்கள் உள்ளன . அவற்றில் பல அதிக மகசூல் தரும் ரகங்களும், வெல்வேறு சூழ்நிலைக்கு தகுந்த ரகங்களும், நோய்க்கு எதிரான ரகங்களும் உள்ளன.



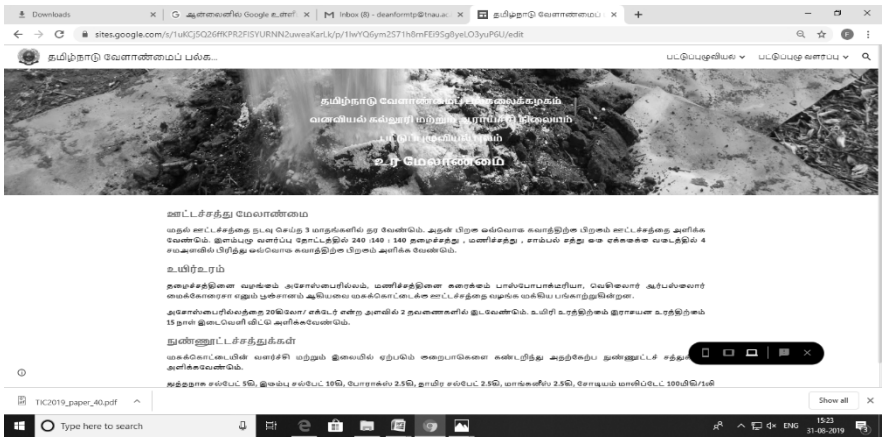
படம்.4 முசுக்கொட்டை நாற்றங்கால் அமைத்தல்

முசுக்கொட்டையானது நாற்றங்கால் மூலமே வளர்க்கப்படுகிறது . அதோடு அது விவசாயிகளுக்கு எளிதாகவும் , கால நேரத்தை குறைப்பதாகவும் உள்ளது.



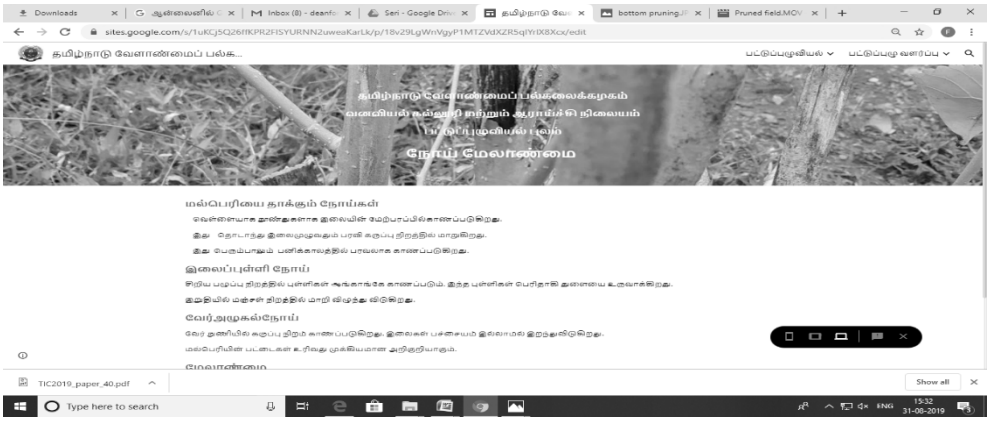
படம்.4 முசுக்கொட்டை நடவு முறை

முசுக்கொட்டையானது பல்வேறு நிலங்களில் வளர்க்கப்படுகிறது . ஆனால் சில நிலங்களில் மட்டுமே நன்றாக வளர்கிறது. அதற்கு தகுந்த நிலத்தில் பல்வேறு உத்திமுறைகள் தேவை.



படம்.5 உரமேலாண்மை

முசுக்கொட்டையின் வளர்ச்சியை அதிகரிக்க உரமானது பல்வேறு நிலைகளில் ,சரியான அளவுகளில் இட வேண்டும்



படம்.6 நோய் மேலாண்மை

அப்படி வளர்க்கும் போது முசுக்கொட்டையானது பல்வேறு நோய்களால் பாதிக்கப்படுகிறது .அவற்றை கட்டுப்படுத்த சில முறைகள் உள்ளன.

படம்.7 நீர் மேலாண்மை

தாவரத்தின் வளர்ச்சிக்கு நீர் மிகவும் முக்கியமான ஆதாரம் ஆகும். நீரை சிக்கமானமாகவும் , தாவரத்தின் வளர்ச்சியை பாதிக்காதவாறும் பயன்படுத்த வேண்டும்.

படம்.8 கவாத்து செய்தல்

தாவரத்தின் வளர்ச்சியை அதிகரிக்கவும் , பட்டுப்புழு வளர்ப்புக்கு ஏற்றவாறு அதனை மாற்றவும் கவாத்து முக்கியமானது.

இதே போன்று முசுக்கொட்டை சாகுபடிக்கான அனைத்து தொழில்நுட்பங்கள்,முசுக்கொட்டை சாகுபடிக்கான வரவு செலவுப்பட்டியல்,இளம்புழு வளர்ப்பிற்கான தொழில்நுட்பங்கள் மற்றும் அதற்கான உற்பத்திச் செலவு மற்றும் வளர்ந்த புழு வளர்ப்பிற்கான தொழில்நுட்பங்கள் மற்றும் உற்பத்திக்கான வரவு செலவுப் பட்டியல் போன்றவை இவ்விணையதளபல்லூடக மின்கற்றல் கட்டமைப்பில் பதிவேற்றப்பட்டுள்ளது.

முடிவுரை

பட்டுப்புழுவியல் சார்ந்த தமிழ் வழி இணையதள பல்லூடக மின்கற்றல் கட்டமைப்பு எளிதில் பயன்படுத்தும் வகையில் கூகுள் இணையதள வசதியில் கட்டமைக்கப்பட்டுள்ளது. மேலும் இவ்வசதியை பல்வேறுபட்ட பயனாளிகளான மாணவர்கள், விவசாயிகள், விஞ்ஞானிகள், தொழில்முனைவோர், மற்றும் ஆர்வமுள்ள பெண்கள் கற்று பயன்பெறும் வகையில் கட்டமைக்கப்பட்டு அவர்களின் வாழ்வாதாரத்தை மேம்படுத்த பயன்படும் என்று எதிர்பார்க்கப்படுகிறது. இது முடிவற்ற தொடர்ச்சியான புதுப்புதுத் தகவல்களை பதிவேற்றம் செய்யும் விதத்திலும் பயனாளிகளுக்கு தங்களைப்பற்றிய விவரங்களையும் தேவைகளையும் எதிர்பார்ப்புகளையும் நேரடியாக பதிவேற்றம் செய்து தகவல்களை எளிதில் பரிமாற்றம் செய்யும் வகையில் அமைவதோடு தேவைக்கேற்ப புதிய தொழில் நுட்பங்களை தொடர்ந்து அளிக்கக்கூடியதாகவும் இவ்விணையதளம் அமையவிருப்பதால் இது தொடர்ந்து புத்தம்புதிதாக மிலிருந்து கொண்டே இருக்கும்.

குறிப்புகள்

1. சர்கா ஹுபாகோவா, வரலாறு மற்றும் முன்னோக்குகள், பக்கங்கள் 1187-1190, தொகுதி 191,2015.
2. அப்துல்அஸிஸ் அல்தியாப், ஹருன் சவுத்ரி, அலெக்ஸ் கூட்சூகோஸ், ஃபிரோஸ் ஆலம், ஒரு விமர்சனம்,எரிசக்தி செயல்முறை, பக்கங்கள்2752-2756, தொகுதி 116, 2014,

3. முஸ்தபா ஹருன், கேன் , அறிவியல் ஒலிம்பியாட்டில் ஆசிரியரின் எலினெரிங் பயன்பாடு பற்றிய விசாரணை, பக்கங்கள் 241-249, தொகுதி 191,2015

4. அப்துல்லா ஹரிரி, ஹிச்சாம் பிஹ்ரி பாஸி,அலர்னிங் செயல்பாடுகளை வடிவமைக்க ஆவணப்படங்களைப் பயன்படுத்துதல், பக்கங்கள்2752-2756,
தொகுதி 116, 2014.

5. விக்கிபீடியா https://www.wikipedia.org/

6. google தளங்கள் https://sites.google.com/

Algorithms for certain classes of Tamil Spelling correction

Muthiah Annamalai*, T. Shrinivasan
ezhillang@gmail.com

1. Introduction

Tamil language has an agglutinative, diglossic, alpha-syllabary structure which provides a significant combinatorial explosion of morphological forms all of which are effectively used in Tamil prose, poetry from antiquity to the modern age in an unbroken chain of continuity.

However, for the language understanding, spelling correction purposes some of these present challenges as out-of-dictionary words. In this paper the authors propose algorithmic techniques to handle specific problems of conjoined-words (out-of-dictionary) e.g. தென்றல்காற்று = தென்றல் + காற்று when parts are alone present in word-list in efficient way. Morphological structure of Tamil makes it necessary to depend on synthesis-analysis approach and dictionary lists will never be sufficient to truly capture the language.

1.1 State of The Art

Many popular spell-checking applications have advanced state of the art in Tamil spelling correction but their methods are sometimes opaque. The work of Rajaraman(Vaani) [14] and Dr. Vasu Renganathan [13] remain popular in the interwebs, while Google has used hunspell/aspell variants with affix dictionary of Thamizha collaborative [15] to create spelling support of Tamil text in Google Docs product - without contributing back much. Whereas, the input method editor for Apple iOS and such from Murasu Anjal team is proprietary [16]. Our own work of solthiruthi is nascent and performs somewhat slowly due to overzealous searches, and not ready for production; however, that does not mean inferiority in class of corrections it is capable of - some of which is shared in this paper.

1.2 Overview

Authors propose some algorithms to identify and correct conjoined words; algorithms forMayangoli letter transposition/substitution error correction; algorithms are proposed for correcting typographical errors originating from keyboard layouts. Authors also propose using machine-learning / deep-neural network based techniques to (a) identify a Tamil word sequence as valid word or mis- spelled word. Tamil misspelled letters can be collected using crowd-sourced format and used as training data for a Tamil based Deep Neural Network (DNN) for the purpose of classifying correctly spelled and misspelled words. Further, Tamil verb declension problem can also perhaps be resolved using the big-data AI/ML approaches.

We also present algorithmic techniques to optimize a spell-checker implementation, to detect the presence of Tamil unicode character in unicode-point quickly, and also algorithm to detect foreign- language words in Tami and substitute with equivalent (when available, or acceptable Tamil form). While some of these techniques are common, this work shows the algorithms in Tamil language context, and their novelties are elucidated in this work.

2. Edit Distance Search Algorithm

Basic spell checker algorithm for Tamil is laid out in [1] by team of Prof. Geetha, Dhanabalan and co-workers which includes verb declension, affix removal, morpheme extraction and then applying corrections to root word and then synthesizing it.

2.1. Norving Algorithm

A simpler algorithm for correcting Tamil words is to directly apply the Edit distance algorithm - popularly referred to as the Norving-algorithm [2,4]. This algorithm essentially computes for each letter in a word the possibility that the letter at the position in word could be,

1. Deleted,
2. Substituted - with alternate letter
3. Inserted - with alternate letter

and enumerates through all the alternate forms of input word at edit distance of 2

Clearly the presence of 247 unique Tamil letters (323 including Grantha letters) including the alpha-syllabary forms explodes the search space for a N-letter word can go as, $(247)^N$ or $(323)^N$ as you may choose to implement the algorithm for an edit-distance of upto N letters in word. Clearly not all letters in a word will be mis-spelled.

Generally the algorithm is limited to searching 2 or 3 edit distance. For an N letter word, we have then ${}_NC_2$ x 323 x 323 options or ${}_NC_3$ x 323 x 323 options respectively for the chosen edit-distance. Judiciously implementing this algorithm is key to having a data-driven spell-checker perhaps by tree-pruning techniques [11].

The implementation in Open-Tamil solthiruthi module is the following which generates suggestions which are filtered against a dictionary (typically represented as Trie data structure [8]. This edit-distance search occurs in spelling correction in Mayangoli correction, and Typographical error correction as well as described in the sections below.

```
def norvig_suggestor(word:list,alphabets=tamil_letters,nedits=1,limit=float("inf")):
    # recursive method for edit distance > 1
    if nedits > 1:
        result = []
        for nAlternate in norvig_suggestor(wordL,alphabets,nedits-1,limit-len(result)):
            if len(result) > limit:
                break
            result.extend( norvig_suggestor(nAlternate,alphabets,1,limit-len(result)) )
        return set(result)

    ta_splits     = [ [u"".join(wordL[:idx-1]),u"".join(wordL[idx:])] for idx in range(len(wordL) + 1)]
    #pprint( ta_splits )
    ta_deletes    = [a + b[1:] for a, b in ta_splits if b]
    ta_transposes = [a + b[1] + b[0] + b[2:] for a, b in ta_splits if len(b)>1]
    ta_replaces   = [a + c + b[1:] for a, b in ta_splits for c in alphabets ]
    ta_replaces2  = [ c + b for a, b in ta_splits for c in alphabets ]
    ta_inserts    = [a + c + b     for a, b in ta_splits for c in alphabets]
    # TODO: add a normalizing pass word words in vowel+consonant forms to eliminate dangling ligatures
```

```
return set(ta_deletes + ta_transposes + ta_replaces + ta_replaces2 + ta_inserts )
```

2.2. Driver Algorithm for Spell Checker

சொல்திருத்தியில் கணினி நிரல் செய்யவேண்டியது இதுவே:

1. உள்ளீடு கொடுக்கபட்ட சொல் சரியானதா, அல்லது தவறானதா ?
2. தவறான சொல் என்ற பட்சத்தில் அதன் மாற்றங்கள் என்னென்ன ?

As per the above algorithm, we correct only words not in dictionary. This is non-word error correcting algorithm - i.e. it corrects only words not in dictionary. This algorithm won't correct word-sense disambiguation type errors, homonyms etc. - i.e. the in-dictionary word error which occurs out of place in text. Algorithm will iterate through the words in text and generate alternates using best effort for the wrong word and send them to user (replacing per user choice).

3. Mayangoli Letter Transposition/Substitution Error Correction

தமிழில் உள்ள மயங்கொலி எழுத்துகள் நான்கு வரிசையில் அமைக்கலாம் [9],

- ல, ழ, ள வரிசை.
- ர, ற வரிசை.
- ந, ன, ண வரிசை.
- ங, ஞ, வரிசை.

சொல்திருத்தியில் கணினி நிரல் செய்யவேண்டியது இதுவே:

1. உள்ளீடு கொடுக்கபட்ட சொல் சரியானதா, அல்லது தவறானதா?
2. தவறான சொல் என்ற பட்சத்தில் அதன் மாற்றங்கள் என்னென்ன?

முதல் படியை எளிதாக ஒரு கையகராதியை கொண்டு செயல்படுத்தலாம். இதனை ஓபன்-தமிழ் (open-tamil) solthiruthi தொகுப்பில் Tamil VU மின் அகராதியை கொண்டு செயல்படுத்தியுள்ளோம். சரியான சொற்கள், அதாவது வேர் எடுத்த, புணர்ச்சி மற்றும் சாந்தி பிரிக்கப்பட்ட சொற்கள் அனைத்தும் சராசரி மின்அகராதியில் காணலாம். இதுவே எளிதான படி.

இரண்டாவது படிதான் ஒரு சொல்திருத்தியின் சிறப்பிற்கும், தரத்திற்கும், முக்கியமானது; இந்த பதிவில் எப்படி மயங்கொலி எழுத்து பிழைகளை திருத்தலாம் என்று சில எண்ணங்களை சமர்ப்பிக்கிறேன். உதாரணம் உரையின் சொல் "**பளம்**" என்பது பிழை என்று கண்டறியப்பட்டது. இது **பள்ளம்**, அல்லது **பழம்** என்று இரு மாற்றங்களை எழுத்தாளர் நினைத்தாலும் இதனை பிழையாக உள்ளீடு செய்துள்ளார். இங்கு **ள-ல-ழ** மயக்கம் காணப்படுகிறது. இதனை கணினி "பலம்", "பழம்" என்றும் மாற்றுகளை உருவாக்கி இதில் அகராதியில் உள்ளவற்றை மட்டுமே வடிகட்டி எழுத்தாளருக்கு பரிந்துரை செய்யவேண்டும்.

இதனை கொண்டு அணைத்து மயங்கொலி பிழைகளை திருத்தும் ஒரு தன்மை கொண்ட சொல்திருத்தியை உருவாக்கலாம். உதாரணம்,

வளர்ச்சி நிலையில் உள்ள, தற்போது மென்பொருள் வடிவமைப்பில் உள்ள சொல்திருத்தி ஓபன்-தமிழ் தொகுப்பில் காணலாம்: [எச்சரிக்கை: இது இன்னும் பொது பயன்பாட்டிற்கு பொருத்தமானதல்ல]

```
:~/devel/open-tamil$ ./spell.sh -i
>> பளம்
சொல் “பளம்” மாற்றங்கள்
(0) பம், (1) பளகு, (2) உளம், (3) பள், (4) அளம், (5) ஆளம், (6) பழம்
```

இதன் செயல்முறை கீழ்கண்டவாரு பைத்தன் மொழியில் அமையும் (இடம் சுருக்கத்தின் காரணமாக constructor மற்றும் சில comment/குறிப்புகள் தரப்படவில்லை:

```
class Mayangoli:
  varisai = [[ u"ல்", u"ழ்",u"ள்"],[u"ர்", u"ற்"],[u"ந்",u"ன்",u"ண்"],[u"ங்",u"ஞ்"]]#வரிசை.
  @staticmethod
  def run(word,letters):
    obj = Mayangoli(word,letters)
    obj.find_letter_positions()
    if len(obj.matches_and_positions) == 0:
      return []
    obj.find_correspondents()
    obj.generate_word_alternates()
    return obj.alternates

  def find_letter_positions(self):
    for idx,letter in enumerate(self.letters):
      p = tamil.utf8.splitMeiUyir(letter)
      if len(p) == 1:
        continue
      mei,uyir=p
      for r in range(0,len(Mayangoli.varisai)):
        for c in range(0,len(Mayangoli.varisai[r])):
          if mei == Mayangoli.varisai[r][c]:
            self.matches_and_positions.append((idx,r,c))
    return len(self.matches_and_positions) > 0
  def find_correspondents(self):
    for pos,r,c in self.matches_and_positions:
      src_letter  = self.letters[pos]
      _,src_uyir = tamil.utf8.splitMeiUyir(src_letter)
      alt_letters = list()
```

```
            for alternate_mei in Mayangoli.varisai[r]:
                alt_letters.append( tamil.utf8.joinMeiUyir(alternate_mei,src_uyir) )
            self.pos_classes.append(alt_letters)
        return True
    def _generate_combinations(self):
        return itertools.product(*self.pos_classes)
    def generate_word_alternates(self):
         for position_sub in self._generate_combinations():
            alt_letters = copy.copy(self.letters)
            if _DEBUG: pprint.pprint(position_sub)
            idx =0
            for pos,r,c in self.matches_and_positions:
                alt_letters[pos] = position_sub[idx]
                idx += 1
            word_alt = u''.join(alt_letters)
            self.alternates.append(word_alt)
        return True
```

4. Algorithm For Conjoined Word Recognition

[தென்றல்காற்று = தென்றல் + காற்று] இரு சொற்களும் சொல் அகராதியில் இருந்தாலுன் கூட இனைந்த சொல் அகராதியில் இருக்காது. இதனை தற்காலிக சொல்திருத்திகள் பிழை என்று சொல்லும். ஆனால் இந்த செயல்முறையினால் நாம் இவற்றை பிரித்துப் பார்த்து சரியான் சொல் என்று கண்டறியலாம்.

இந்த செயல்முறை ஓப்பன் தமிழ் தொகுப்பில் இவ்வாறு:

```
class OttruSplit:
    """ யாரிகழ்ந்து = [ய் + ஆரிகழ்ந்து], [யார், இகழ்ந்து] ,[யாரிக், அழ்ந்து], [யாரிகழ்ந்த்,உ]"""
    def run(self,lexicon):
        self.generate_splits()
        return self.filter(lexicon)
    def generate_splits(self):
        """
            யாரிகழ்ந்து =
                [['ய்', 'ஆரிகழ்ந்து'],
                 ['யார்', 'இகழ்ந்து'],
                 ['யாரிக்', 'அழ்ந்து'],
                 ['யாரிகழ்ந்த்', 'உ']]
         """
        L = len(self.letters)-1
        for idx,letter in enumerate(self.letters):
```

```
        if not( letter in tamil.utf8.grantha_uyirmei_letters):
            continue
        muthal = idx == 0 and u"" or u"".join(self.letters[0:idx])
        meethi = idx == L and u"" or u"".join(self.letters[idx+1:])
        mei,uyir = tamil.utf8.splitMeiUyir(letter)
        muthal = muthal + mei
        meethi = uyir + meethi
        self.results.append([muthal,meethi])
    return len(self.results) > 0


  def filter(self,lexicon):
    self.results = list( filter(lambda x: all( map(lexicon.isWord,x) ),self.results) )
    return self.results
```

5. Typographical Error Correction In Tamil

அதாவது தமிழில் தமிழ்99 அல்லது அஞ்சல் போன்ற விசைகளின் வழி உள்ளீடு செய்கையில் எழுத்துப்பிழைகள் வந்துவிடும். இதனை எப்படி சரிபாப்பது ? சில செயல்முறைகளை இதற்காக கையாளலாம். இதன் விரிவான் கட்டுரையை இங்கு காணலாம் [10].

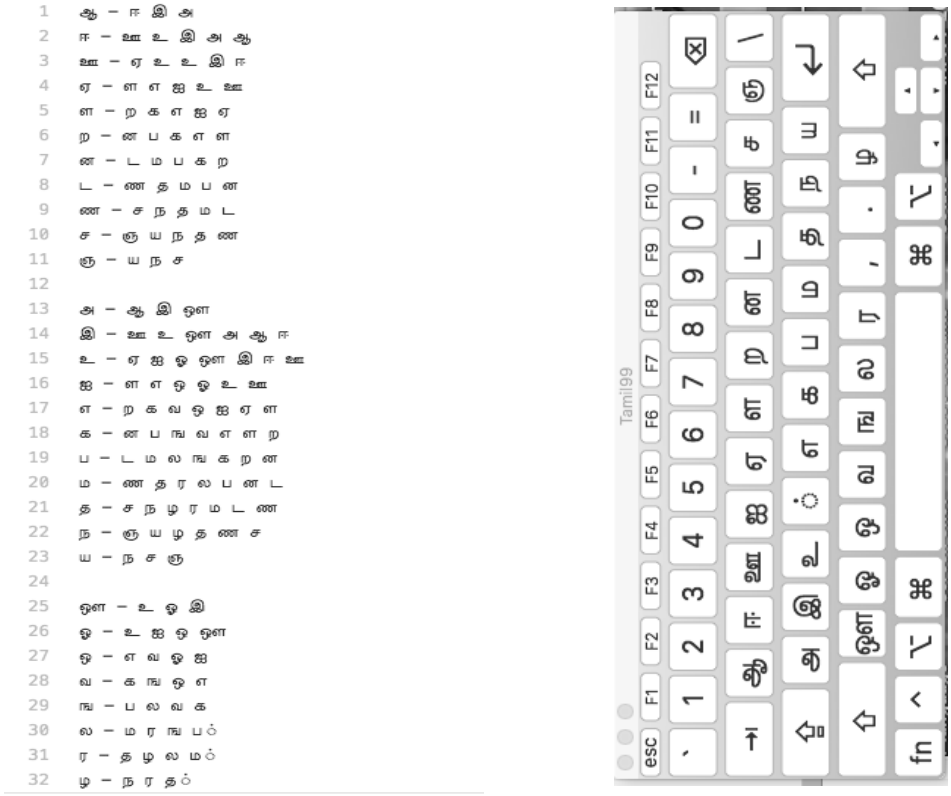


Figure: (left) confusion matrix for Tamil-99 keyboard, (right) Keyboard layout - Tamil-99.

இந்த அல்கோரிதத்தின் நிரலாக்கம் இங்கு ஓப்பன் தமிழ் திரட்டில் சேர்க்கப்பட்டது. இதனை நீங்கள் முழுதேடலில் இடம் கொடுத்தால் 2398 விடைகள் கிடைக்கும் – அதாவது முழு 4-எழுத்து சொல்லின் 4-எழுத்து தொலைவில் உள்ள திருத்தங்கள் எல்லாவற்றையும் தேடுவதால் உண்டாகும் தகவல் வெள்ளப்பெருக்கு; சாதாரணமாக 1 அல்லது 2 எழுத்துப்பிழைகள் மட்டுமே உள்ளன என்பது அறிவியலாளர்கள் கணிப்பு. இதை நாம் செயல்படுத்தும் 'tree pruning search' அல்கொரிதம் வகையினால் நாம் 56 மாற்றங்களுக்குள் மட்டுமே தேடல்களை நடத்தி இந்த தட்டச்சு கைவிரல் தவரான உள்ளீட்டிற்கு தீர்வு காணலாம்.

இதன் சிக்கல் அளவு [computational complexity] என்பது, ஒரு n-எழுத்து சொல் என்று கொண்டால், $O(k_1 \times k_2 \times k_3 \ldots k_n) = O(k^n)$ என்று அதிக பட்சமாக இருக்கலாம் என்று [ஏதோ ஒரு $k > 0$ எண்ணால்] என்று நம்மால் காட்டமுடியும். This is also a variant of edit-distance search.

```
# explore all edit distances - i.e. len(word_in) or only upto value in ed.
# we can restrict the edit distance search to any value from [1-N]
def oridam_generate_patterns(word_in,cm,ed=1,level=0,pos=0,candidates=None):
    """ ed = 1 by default, pos - internal variable for algorithm" """
    alternates = cm.get(word_in[pos],[])
    if not candidates:
        candidates = []
    assert ed <= len(word_in), 'edit distance has to be comparable to word size [ins/del not explored]'
    if (pos >len(word_in)) or ed == 0:
        return candidates
    pfx = ''
    sfx = ''
    curr_candidates = []
    for p in range(0,pos):
        pfx = pfx + word_in[p]
    for p in range(pos+1,len(word_in)):
        sfx = sfx + word_in[p]
    for alt in alternates:
        word_alt = pfx + alt + sfx
        if not (word_alt in candidates):
            candidates.append( word_alt )
            curr_candidates.append( word_alt )
    for n_pos in range(pos,len(word_in)):
        # already what we have ' candidates ' of this round are edit-distance 1
        for word in curr_candidates:
            oridam_generate_patterns(word,cm,ed-1,level+1,n_pos,candidates)
    if level == 0:
        #candidates.append(word_in)
```

```
        for n_pos in range(pos,len(word_in)):
            oridam_generate_patterns(word_in,cm,ed, level+1,n_pos,candidates)
    return candidates

def corrections(word_in,dictionary,keyboard_cm,ed=2):
    """
    @input: word_in - input word
        dictionary - dictionary/lexicon
        keyboard_cm - confusion matrix for keyboard in question
    """
    assert isinstance(dictionary,Dictionary)
    candidates = oridam_generate_patterns(word_in,keyboard_cm,ed)
    #TBD: score candidates by n-gram probability of language model occurrence
    #etc. or edit distance from source word etc.
    return list(filter(dictionary.isWord,candidates))
```

6. AI/Ml Approaches to Tamil Spelling Correction

Wide-spread success of various tasks like image recognition [6] which surpasses human level cognition in this task, and significant improvements in speech synthesis and speech recognition have demonstrated A.I. and Machine Learning methods suitable for these tasks, Sequence to sequence models including Recurrent Neural Networks (RNN), Long Short-term Memory (LSTM), and Word2Vec type representations of Neural Networks have enabled high success rate in translation, concordance tasks [7].

6.1 Anomaly Detection to Reduce Processing

Currently spelling correction is required in less than 25% of user input which remains perhaps requiring deep analysis and complex algorithms. To improve speed of spelling correction we can also take the approach of training a AI/ML based system to declare word or text as valid Tamil word - this can (in principle) automatically resolve 75% of input text and free up the time required for the complex analysis. Such heuristics can allow using anomaly detection algorithms for proper identification of the errors in the text and filter the computational load on the spell checking program.

Further, Tamil misspelled letters can be collected using crowd-sourced format and used as training data for a Tamil based Deep Neural Network (DNN) for the purpose of classifying correctly spelled and misspelled words. Further, Tamil verb declension problem (see: Rajam Krishnan) can also perhaps be resolved using the big-data AI/ML approaches.

6.2. Word Sense Disambiguation

1) அன்பே சிவம். 2) அன்பே சவம்

Both the sentences contain legal words from a Tamil Lexicon but only one makes sense [12]. The simple minded non-word error correcting spell-checker will not be able to tell them apart.The simple way to tell them apart is to use a concordance database for Tamil and find the words co-occurring successors/predecessors and offer alternate. Yet another way is to find the word-level bi-gram, tri-gram probability distribution from a language model for Tamil and use it to identify *சவம்* is not a highly probable successor to '*அன்பே*' thereby determining such an instance could trigger the suitable search and replacement.

6.3 Algorithm for Foreign-Language Word Substitution

In a previous research article we proposed a fully feed-forward ANN which was capable of identifying non-tamil words in text as well as English text (originally reported in [17]).

We propose a simple algorithm which can use the above AI classifier and extract this word and look up equivalent Tamil word for English or other language text using a parallel dictionary. Updating the word for the tense and any morphological processing we can replace the anglicized word in Tamil script with an equivalent Tamil word.

6.4 Computational Complexity

We try to answer the question of complexity of Tamil language spell-checker algorithm in various contexts. It seems the correction of Tamil words in text will be somewhere between exponential in number of letters of word - clearly the substitution cases are shown to be exponentially complex in number of letters.The complexity of affix removal is polynomial in the number of affixes.

7. Optimizing A Spell Checker Implementation

We can propose techniques for how to improve speed of spell checker.

7.1 Algorithm for Fast Unicode Letter Detection

Text containing a mix of Tamil and English (or other language scripts) can be quickly eliminated by using the check if a character falls into the Tamil Unicode block in basic block range. This check eliminates a character in analyzed text from further complex processing. In Python3, the following function performs the Unicode Tamil letter/character detection.

```
is_tamil_unicode_predicate = lambda x: x >= chr(2946) and x <= chr(3066)
```

7.2. Performance Engineering

7.2.1. Caching Results

A given text for spelling correction can be grouped into a series of words, excluding stop words. Now we can make spell checker save suggestion list for each mis-spelled word (non-word) and re-use the suggestions list from cache the second and later times when word is mis-spelled identically in the document. The same mis-spelling can, however, have different correct words in the document at each site depending on context and only be replaced upto first order approximation in “white-washed” fashion. Caching has a potential to restore performance of spell checker.

7.2.2. Multi-Threading/Multi-Processing

The non-word errors suggestion generation and morphological processing can be carried out in parallel in the whole document, perhaps by a producer-consumer model serviced by a dozen(s) of worker threads which all generate suggestions for non-word errors including caching as mentioned above. Such an architecture of spelling checker could improve speed.

7.2.3. Redis / Distributed Db

The memory performance of the spelling checker can be offloaded to a remote machine by requesting a distributed data-based to hold the dictionary or trie form of the word-list/Lexicon. This may be suitable for production environments.

8. Conclusion

In this paper we have attempted to make a summary of various known algorithms for specific classes of Tamil spelling errors. We believe this collection of suggestions to improve future spelling checkers. We also note do not cover many important techniques like affix removal and other such techniques of key importance in rule-based spell checkers.

REFERENCES

1. "Tamil spell checker," T. Dhanabalan, R Parthasarathi – Sixth Tamil Internet 2003-இல அணணா பல்களைக்கழகத்தில் வெளிவந்த கட்டுரை.
2. J. L. Peterson, "Computer programs for detecting and correcting spelling errors," CACM (1980).
3. சொல்திருத்தி – தெறிந்தவை 1, https://ezhillang.blog/2019/02/08/சொல்திருத்தி-தெறிந்தவை-1
4. மயங்கொலி சொற்களின் பிழைதிருத்தங்கள், https://ezhillang.blog/2017/11/04/மாலை-பொழுதின்-மயக்கமென்ன/
5. Peter Norvig, "How to write a spelling corrector," https://norvig.com/spell-correct.html
6. L. Fei-Fei, ImageNet: crowdsourcing, benchmarking & other cool things, CMU VASC Seminar, March, 2010.
7. Ian Goodfellow and Yoshua Bengio and Aaron Courville, 'Deep Learning,' MIT Press (2016).
8. Trie Data Structure, https://xlinux.nist.gov/dads/HTML/trie.html and https://en.wikipedia.org/wiki/Trie (accessed Aug 2019).
9. Mayangoli Sorkal List from Valaitamil.com http://www.valaitamil.com/list-of-mayankoli-sorkal_15177.html (accessed Jan 2019)
10. Typographical error correction for Tamil-99 keyboard https://ezhillang.blog/2019/02/18/சொல்திருத்தி-தெறிந்தவை-3/ (2019).
11. Peter Norvig, 'Artificial Intelligence - a modern approach,' 3rd ed. (2009).
12. தமிழில் சொல் பொருள் கொண்ட திருத்தி, https://ezhillang.blog/2018/04/07/அன்பழகன்-வாத்தியார்/
13. Tamil Spell Checker by Dr. Vasu Renganathan, http://thetamillanguage.com/spellcheck/
14. Vaani Tamil Spell Checker by Rajaraman http://vaani.neechalkaran.com/
15. Hunspell Affix dictionary by V. Ilanchezhian, Mugunth, et-al Thamizha Collaborative, https://github.com/thamizha/thamizha-solthiruthi
16. Muthu Nedumaran and Murasu Anjal-team, iOS/Mac OS (High Sierra/Mojave) support for Tamil input editing via Tamil 99 (Anjal) keyboard, and Tamil phonetic keyboard (accessed 2019).
17. Syed Abuthahir et-al, "Growth and evolution of open-tamil," INFITT, Coimbatore, India (2018).

A TOOL TO EXPLORE MORPHOLOGICAL REGULARITIES OF TAMIL LANGUAGE USING WORDEMBEEDINGS

J.Jeyanthasingam, K.Suthagar, P.Paralogarajah, N.Kavirajan, T.Uthayasanker, Department of Computer Science and Engineering,
University of Moratuwa
{asingam.14, suthagar.14, piraveena.14, nishanthini.14, rtuthaya}@cse.mrt.ac.lk

Abstract. We present an unsupervised approach for exploring morphology using word embedding. Our method builds a morphological generator based on morpho-lexical senses. We evaluate this approach using a morphologically rich language(Tamil). We show this approach is capable of analyzing a wide range of morphological regularities in embedded space for morphologically rich languages. Our approach gains importance since it requires less manual effort compared to prior approaches to build such tools.

Keywords: Neural embedding, morphology, morpho-lexical sense, morphologically rich language, morphologically poor language, regularities

1. Introduction

Morphological analysis is used for many tasks in Natural Language Processing (NLP) including text retrieval [1], speech recognition [2], automatic translation and dictionary automation [3]. Morphology is the system involved in word formation or in the branch of linguistics that deals with words, their internal structure, and how they are formed [4]. Morphological features differ among languages where some languages use the order of words to represent grammatical information while other languages embed this information within words using morphology. Morphologically Rich Languages (MRLs) are the languages in which substantial grammatical information, i.e., information concerning the arrangement of words into syntactic units or cues to syntactic relations, is expressed at word level [5]. The morpheme is the smallest grammatical part of a language that has a morpho-lexical sense (MLS). Morphemes can be used to generate different morphological forms of a root word. For example, the word “played” is created by integrating the morpheme “ed” (i.e., past tense morpheme) with the stem “play” (i.e., root verb).

Morphological analysis and generation are essential steps in any NLP tasks [6]. Morphological analyser analyses the morphemes in a word while the morphological generator generates the desired morphologically inflected form of the root. Previous approaches used rules-based, [7], POS tag based and word embedding based [8] [9] techniques to analyse or generate morphology [10]. Among these approaches, most of the existing morphological analysis and generation were carried out using rule based approach. But the difficulty in solving inflections and exceptions of the language [6], difficulty in solving ambiguity [7], and the need for human efforts are the main limitations in the rule based morphological analysis. Since word embedding use vector representations, it differs from other methods with its ability to perform arithmetic operations with words

Word Embedding represents a word as a real-valued vector, plays an important role in building word vectors based on the contexts in a corpus [11]. These vector representations help to the semantic and syntactic regularities in a language [9]. Specifically, these linguistic regularities are observed as constant vector offsets between pairs of words sharing a particular relationship, e.g. “clothes is to shirt as dish is to bowl” (clothes:shirt::dish:bowl) [9]. Current research works on linguistic regularities in word embeddings focus on

the so-called “proportional analogies” of a:b::c:d kind [12]. Morphological regularities are the morphology based linguistic regularities captured in the word embedded space. “See is to sees as return is?” will capture the present-third party-singular form of the root verb “return”. Mikolov et al. [9] and Soricut et al. [13] exploited these morphological regularities in an unsupervised learning
Pre-existing works on word embedding based morphological analysis tried to solve this issue using the morphological regularities. In this paper, we present a tool to explore morphological regularities in depth beyond simple arithmetic operations. This tool can be served as A Morpho-Lexical Sense based Tamil Morphological Generator.

2. Related Works

In this section, we have discussed the prior works in Morphological regularities in neural word embeddings. Many recent research works have been carried out in word representations via word embeddings [6] [14] [15] and we can see, there is a growing trend in utilizing neural word embeddings for NLP applications. It has been proved that word embeddings capture both syntactic and semantic properties of natural languages [9]. Later, continuous word embeddings or neural embeddings have been shown to capture morphological similarity using different models such as subword and character-based word embedding model [16], log-bilinear model [17], and techniques such as implicitly using morphological information [18].

Previous research works have explored the morphological regularities exhibited in the word embedding vector space [8] [5] [13]. Most of these research works are on affix based unsupervised methods [8] [13]. The Google analogy test [5] which contains 9 morphological categories [12], is a break-through in morphological regularities based analysis. Later Soricut and Och [13] exploited these regularities to generate affix based morphological transformation rules in an unsupervised, language-agnostic fashion. They showed that their method is capable of discovering a wide range of morphological rules, which in turn were used to build morphological analyzers and they have evaluated the method across six different languages [13]. Arihant Gupta et al. [8] improved this approach to exploit morphological regularities present in high dimensional vector spaces.

In contrast to the prior approaches, the work presented here is able to capture the morphological regularities using morpho-lexical senses rather than the affix based pattern matching approach. To the best of our knowledge, this is a novel approach to manipulate morphology using its morpho-lexical senses. Morpheme level analysis is carried out in the embedded space and subsequently, this study helped to derive a word embedding based morphological generator for Tamil.

3. Background

Morphology is a concept that deals with the systematic correspondence between the meaning and the form of words. Inflection and word-formation domains are comprised under the study of these regularities. Inflection deals with the morphosyntactic properties while word formation deals with the creation of new (complex) words by various morphological mechanisms such as affixation, truncation, and alterations. Morphology plays an important role in word formation with the help of morphemes. For example, in the word “reconsideration”, the prefix “re” and the suffix “ation” are the affixes (i.e.: word attachment that adds a different meaning to the word) with the stem “consider”. Most of the word embedding based morphological analyzers capture only the suffixes and prefixes [8] [13]. But in MRLs, more than one morphemes will be embedded into an affix. An example of this is discussed using the Tamil word “இருந்தது” (irunthathu).

இருந்தது (irunthathu) = இரு (iru) (Root Verb) + ந்தது (inthathu) (suffix)

ந்தது (inthathu) (Suffix) = ந்த் (inthth) (Past Tense Marker) + உ (u) (Verbal Participle Suffix) + அது (athu) (Genitive Case)

இருந்தது (irunthathu) = இரு (iru) (Root Verb) + ந்த் (inthth) (Past Tense Marker) + உ (u) (Verbal Participle Suffix) + அது (athu) (Genitive case)

The suffix in this example contains past tense morphemes, verbal participle morpheme and genitive case morphemes. Further, each morpheme contains an MLS that integrate a piece of information to the word. e.g.: The past tense morpheme is the reason for the tense of this word. Most of the existing models work at the suffix level and are not able to capture the morphemes within the suffix. We experiment with the ability of word embedding in manipulating words at the morpheme level.

4. A Tool for Tamil Morphological Regularities Based On Word Embeddings

We introduce a MLS based morphological generator which works with any inflected form without being restricted to the root form. In order to build an MLS based morphological generator, we have found morphemes and their MLSs from the language and mapped the MLSs with some admissible words. The Table I shows the mapping of MLSs and the admissible words for the English Language.

Morpho-lexical sense		**Admissible words**
Tense	Past Tense	Yesterday, past
	Present Tense	Today, present
	Future Tense	Tomorrow, future
Gender	Masculine form	He
	Feminine form	She
Party information	First party	I, We
	Second party	You
	Third party	They, It, He. She,

Table 1: Mapping of morpho-lexical sense with admissible words

In the table I , we have addressed three MLS such as tense, gender and party information with the sub-divisions in the MLS and corresponding admissible words of the MLSs. According to our implementation, the desired morphological forms of the source verb can be generated using word vectors as follows:
The source – (MLS of the source) + (MLS of the target) (*)

For example, by mapping the MLSs with the admissible words, the operation (“played”) - (“yesterday”) + (“today”) will generate the word “play” using the embedded space. Proposed morphologyGenerator() algorithm based on the operation (*) generates the target words using MLSs of the source and the target. This sense is mapped using admissible words. Later, the top N closest words to the resulting vector are identified based on the cosine similarity. Theoretically, the closest word should be the

target morphological form. But the model struggled to produce accurate results because of noise and low occurrences of these words. To overcome these issues, an additional rule based filter is used to retrieve the most probable word from the closest N words. This filter returns the closest word which contains the first letter and last letter the same as the expected output. This algorithm requires the characteristics (form, gender, party, tense, etc...) of the source and target as input and returns the relevant morphological form of the word.

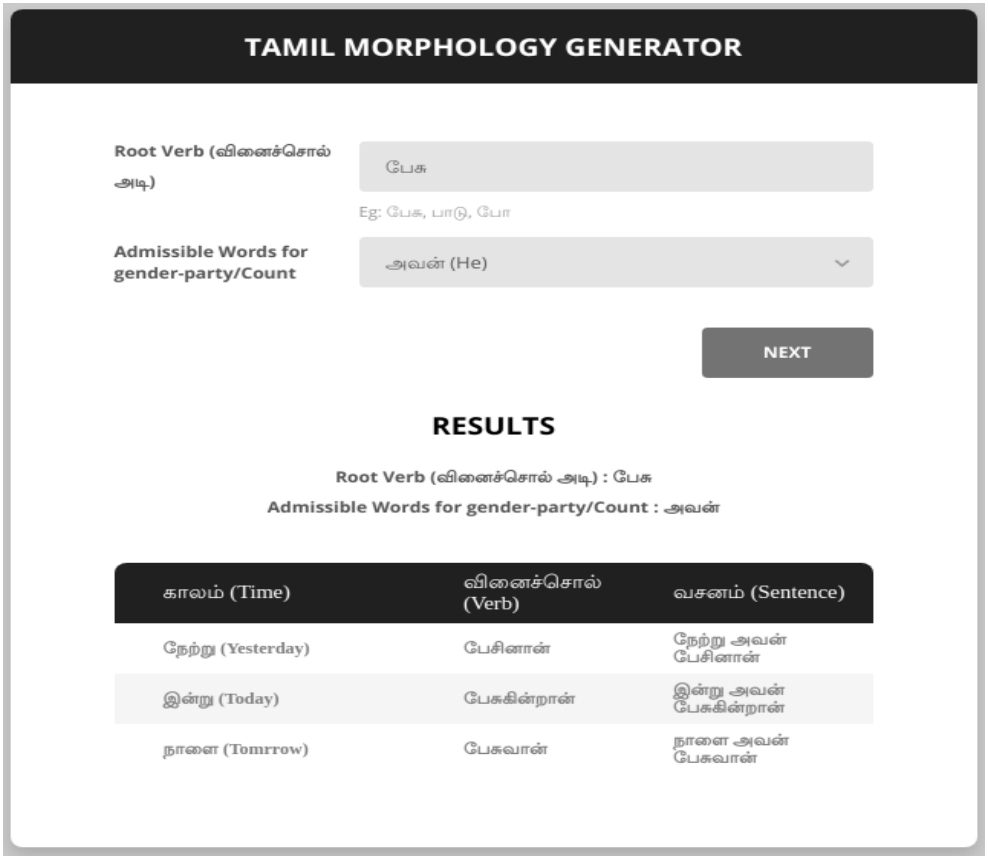


Figure 1: Graphical User Interface of our Tamil Morphological Generator Tool using the Word "பேசு " (Pesu - speak). Once the user types in the verb of interest and choose the admissable word for gender and count the tool will list all learned morphological forms in three tenses along with example sentences.

5. Experiment

We detail the experimental steps in this section. We used skip-gram word embedding technique blazingText approach to generate the word embedding model. We did an in-depth analysis of the word embedding model to find how it behaves with morphology. We implemented a morphological generator on top of this model.

An MLS based morphological generator is derived from the MLS based analysis using admissible words. In this work, we have identified 30 frequently used morphological forms with the help of language experts in Tamil. 13 morphemes that determine party, gender (masculine, feminine or gender-neutral), number and tense of a word, are identified to generate these 30 different forms. Empirically we chose the

best admissible words for each section. Finally, the 30 morphological forms are generated using our algorithm and selected admissible words (table 3).

Each of the admissible word given in the table portrays the corresponding tense, gender, party, count and human/non-viable information that embedded in the particular admissible word

Table 3: The admissible words and the respective senses for the morphemes.

Admissible words	**Sense**	**Admissible words**	**Sense**
அவன் (*avan*) (he)	Third party, Male, human	நான் (*nān*) (I)	First Party, Singular, Human
அவள் (*aval*) (she)	Third party, Female, Human	நாங்கள் (*nāngal*) (we)	First party, Plural, Human
அவர் (*avar*) (he/she)	Third party, Singular, gender neutral, human	நீ (*nī*) (you)	Second Party, Singular, Human
அவர்கள் (*avargal*) (they)	Third party, Plural, Human	நீங்கள் (*nīngal*) (you)	Second party, Plural, Human
அது (*athu*) (it)	Third Party, Singular, dead, non-viable and all living beings except human	நேற்று (*nētru*) (yesterday)	Past Tense marker
அவை (*avai*) (they)	Third Party, Plural, dead, non-viable and all living beings except human	இன்று (*inru*) (today)	Present Tense marker
		நாளை (*nālai*) (tomorrow)	Future Tense marker

This generator performs differently compared to the standard morphological generators. Rather than depending only on the root verb, this can perform with an inflected form as well. An inflected form of a word and integrated morph-lexical sense are required to generate all the target 30 forms. This generator is evaluated using the manually constructed dataset.

6. Results and Analysis

The results of the experiment is presented in this section. Most of the conceptual experimental results are evaluated subjectively while the generator is evaluated subjectively as well as objectively. Findings are analysed for these results.

A MLS based morphological generator is implemented in this research. The most frequent morphological forms of a randomly selected word is given in table 7. Table 7 shows the results from the generator for the stem "பேசு" (pesu).

Table 7: Results for the root verb "பேசு" (pesu)(speak) from our Morphological generator for all 30 forms.

Gender / Tense	**நேற்று(netru)-Yesterday**	**இன்று(intru)-Today**	**நாளை(netru)-Tomorrow**
அவன் (*avan*)(he)	பேசினான் (*pēasinan*)	பேசுகின்றான் (*pēasukinrān*)	பேசுவான் (*pēasuvān*)
அவள் (*aval*)(she)	பேசினாள் (*pēasinal*)	பேசுகின்றாள் (*pēasukinrāl*)	பேசுவாள் (*pēasuvāl*)
அவர் (*avar*)(he/she)	பேசினார் (*pēasinār*)	பேசுகின்றார் (*pēasukinrār*)	பேசுவார் (*pēasuvār*)
அவர்கள் (*avargal*) (they)	பேசினர் (*pēasinar*)	பேசுகின்றனர் (*pēasukinranar*)	பேசுவர் (*pēasuvar*)
அது (*athu*)(it)	பேசியது (*pēasiyathu*)	பேசுகின்றது (*pēasukinrathu*)	பேசும் (*pēasum*)
அவை (*avai*)(them)	-	-	-
நான் (*nān*)(I)	பேசினேன் (*pēasinen*)	பேசுகிறேன் (*pēasukiren*)	பேசுவேன் (*pēasuven*)
நாங்கள் (*nāngal*)(we)	பேசினோம் (*pēasinōm*)	பேசுகிறோம் (*pēasukirom*)	பேசுவோம் (*pēasuvōm*)
நீ (*nī*) (you - singular)	-	பேசுகிறாய் (*pēasukirāi*)	-
நீங்கள் (*nīngal*) (You - plural)	பேசினீர்கள்(*pēasinerrkal*)	பேசுகிறீர்கள் (*pēasukireerkal*)	-

In Table 7, Admissible words that represent the gender, party and number information are given in the first column of the table. Then for each tense (past, present, future), corresponding inflected forms of the root verb are generated with the addition of respective admissible words of each tense. Except for some desired forms, most of the inflected forms of the target word are generated by our morphological generator. The results for the word "பேசு" (pesu) (speak) is generated with 80% accuracy. The model struggle with unseen words

which is the main reason for the miss classifications. A detailed objective evaluation is done to analyse the characteristics of the generator.

7. Conclusion and Future Works

We presented a novel unsupervised Tamil morphological generator that can be easily adapted to any MRL. This needs only a morphologically rich corpus and no other tools nor a huge manual effort. This tool can be implemented in other languages and can be used to many applications including machine translation.

REFERENCES

[1] Hjelm, H., & Schwarz, C. (2006). LiSa–morphological analysis for information retrieval. In Proceedings of the 15th Nordic Conference of Computational Linguistics (NODALIDA 2005) (pp. 65-70).

[2] Al-Sughaiyer, I. A., & Al-Kharashi, I. A. (2004). Arabic morphological analysis techniques: A comprehensive survey. Journal of the American Society for Information Science and Technology, 55(3), 189-213.

[3] Narejo, W. A., & Mahar, J. A. (2016, April). Morphology: Sindhi morphological analysis for natural language processing applications. In Computing, Electronic and Electrical Engineering (ICE Cube), 2016 International Conference on (pp. 27-31). IEEE.

[4] Aronoff, M., & Fudeman, K. (2011). What is morphology? (Vol. 8). John Wiley & Sons.

[5] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[6] Goyal, V., & Lehal, G. S. (2008, July). Hindi morphological analyzer and generator. In First International Conference on Emerging Trends in Engineering and Technology (pp. 1156-1159). IEEE.

[7] Selvam, M., & Natarajan, A. M. (2009). Improvement of rule based morphological analysis and pos tagging in tamil language via projection and induction techniques. International journal of computers, 3(4), 357-367.

[8] Gupta, A., Akhtar, S. S., Vajpayee, A., Srivastava, A., Jhanwar, M. G., & Shrivastava, M. (2017). Exploiting Morphological Regularities in Distributional Word Representations. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (pp. 292-297).

[9] Mikolov, T., Yih, W. T., & Zweig, G. (2013). Linguistic regularities in continuous space word representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 746-751).

[10] Antony, P. J., & Soman, K. P. (2011). Parts of speech tagging for Indian languages: a literature survey. International Journal of Computer Applications (0975-8887), 34(8), 22-29.

[11] Arora, S., Li, Y., Liang, Y., Ma, T., & Risteski, A. (2018). Linear algebraic structure of word senses, with applications to polysemy. Transactions of the Association of Computational Linguistics, 6, 483-495.

[12] Gladkova, A., Drozd, A., & Matsuoka, S. (2016). Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In Proceedings of the NAACL Student Research Workshop (pp. 8-15).

[13] Soricut, R., & Och, F. (2015). Unsupervised morphology induction using word embeddings. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 1627-1637).

[14] Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. Journal of machine learning research, 3(Feb), 1137-1155.

[15] Socher, R., Karpathy, A., Le, Q. V., Manning, C. D., & Ng, A. Y. (2014). Grounded compositional semantics for finding and describing images with sentences. Transactions of the Association of Computational Linguistics, 2(1), 207-218.

[16] Takala, P. (2016). Word embeddings for morphologically rich languages. In Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.
[17] Cotterell, R., & Schütze, H. (2015). Morphological word-embeddings. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 1287-1292).
[18] Xu, Y., & Liu, J. (2017). Implicitly incorporating morphological information into word embedding. arXiv preprint arXiv:1701.02481.

A STUDY ON THE EFFECTIVENESS BEHIND CLASSICAL LANGUAGE TAMIL AS A SCIENTIFIC LANGUAGE FOR COMPUTING

[1]S.Prasanna Devi, [2]S.Manivannan
[1]Professor, Department of CSE, SRM Institute of Science & Technology, prasannasiva11@gmail.com
[2]Professor, Dr.MGR Educational and Research Institute University,

ABSTRACT

Statistics is an art basically involved in collection, interpretation and validation of ~~the~~ data and statistical data analysis focuses on quantitative research which explores to quantify the data. Quantitative data involves descriptive data such as observational data and survey data. This paper conceptualized that the length of words in Tamil is lesser than the length of English equivalent words. The Dataset was constructed with the help of Oxford - English to Tamil Dictionary and words from alphabet 'A' were taken and its lengths were calculated. For both the English and Tamil equivalents for alphabet 'A', the corresponding mean and standard deviation were calculated. Using statistical analysis, the independent sample T test applied over both the English and Tamil words and the results of level of significance obtained proves our conceptualization that computing using Tamil language results in less memory consumption than computing using English.

Keywords: Memory Consumption, Comparison of English versus Tamil equivalent words, word length.

1. INTRODUCTION

Language plays an important role in human's day to day life. It is the only means for humans through which they can communicate their thoughts, ideas and exchange opinions with others. Learning a language does not stop with communication alone. It helps to understand concepts and learn about the existing things in the vast field of education. There were roughly about 6500 languages which were spoken all over the world over a wide range of geographical area. Among those languages, there are 2000 languages which still exist in the world with less than around 1000 speakers. Out of that there are eight ancient languages that are still spoken in the world and Tamil language is one amongst in the list. Tamil language is around 5000 years old and found to be spoken by around 78 million people around the world. It is considered as an official language of Srilanka and Singapore. While English language has a total number of 26 alphabets, Tamil language has 247 alphabets totally and is famous for its rich vocabulary. [1]Apart from its unique words, its rich grammar makes the language more powerful. Most of the people still find comfortable reading and writing in Tamil which is also their mother tongue instead of studying in other languages [2]. Due to the globalization English has taken dominance over the other languages especially in the field of Education. But still, studying in mother tongue helps students in better understanding of the subject. Less research has gone into the field of comparison of word lengths and statistical analysis of English and Tamil. This paved interest in the researchers to do statistical analysis over the word length of English and Tamil language words.

2. DATASET

Here English words for Alphabet 'A' alone with its Tamil Equivalent is collected from the Pocket Oxford Dictionary considering the complexity of collecting for all the 26 alphabets from the Dictionary. There are about 524 words in A alphabet which has been chosen from the pocket dictionary to construct a dataset. It was found that the word length for both the English and Tamil words ranges from small length of 2 to around length of 15. For the 524 word dataset the statistical parameters such as mean, standard deviation, T-score and the P value to find the level of significance are calculated and the results were tabulated. T test is a type of inferential statistics. It is mainly used to find the level of significance, i.e., significant decrease between the mean of the two groups. There are three different types of T test available such as an independent sample t test, a paired sample T test and a one sample T test.

3. STATISTICAL ANALYSIS

The independent sample T test[3],[4] is also known as two sample T test or student's T-test. It is an inferential statistical test which determines whether there is a statistical significance between the two unrelated groups. Unrelated groups [7],[8] are also known as unpaired groups or independent groups i.e., both the groups are different and do not possess any dependency over the other.

The T score can be calculated by the formula:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}}$$

Where N1 is the total number of English words present in the Dataset
N2 is the total number of English words present in the Dataset

Sd_1 is the standard deviation of the samples in English words present in the Dataset

Sd_2 is the standard deviation of the samples in Tamil words present in the Dataset

X_1 is the mean average of the English words in the Dataset

X_2 is the mean average of the Tamil words in the Dataset

The null hypothesis[5],[6] for the independent T-test is that the population averages between the two unrelated groups are equal. Then: $H_0 = \mu_1 = \mu_2$

Where μ_1 is the Mean of Group 1 and μ_2 is the Mean of Group 2

H_0 is the hypothesis stating that mean word length of English and Tamil words are same

H_A is the alternative hypothesis stating that mean word length of English and Tamil words are not same i.e., $\mu_2 < \mu_1$.

In the above case the null hypothesis can be rejected considering the two mean populations of English and Tamil word lengths as equal.

Alternative hypothesis for accepting the statement states that: $H_A = \mu_1 \neq \mu_2$

Accept the hypothesis where the mean population between the English and Tamil word lengths is different with a level of significance. The mean average word length of English and Tamil were calculated from the datasets and the tabulation is listed below.

4. CALCULATION OF MEAN OF ENGLISH WORDS CONTAINING THREE ALPHABETS

From the constructed dataset two word lengths are chosen say 3 and 11 with its Tamil Equivalent. There are about fifteen words in the dataset measuring three characters as length and around fourteen words measuring eleven characters as length. The rule is to measure the length of both English and Tamil words and also calculate the mean for the same.

Consider the below example

All = அனைத்து, முழுமையான

Length of Tamil Word அனைத்து = 4

Length of Tamil Word முழுமையான = 5

All is a three letter English word that gives two equivalent Tamil meaning of two different length words say 4 and 5. Among those two words the maximum word length of 5 is chosen for calculating the mean value. The total number of samples for word length of three is fifteen. The English and Tamil equivalent words and their word length is listed below in the tabular column

Table 1. Sample data for English word of length three with its Equivalent Tamil word Mean

S.No	English Alphabets	Length of the English word	Equivalent Tamil Meaning	Length of the Tamil word
1	Arm	3	கை	1
2	Art	3	கலை	2
3	Aye	3	ஆம்	2
4	Act	3	நடிப்பு, செயல்	4, 3
5	Add	3	கூட்டு	3
6	Ago	3	முன்பு	3
7	Aid	3	உதவி	3
8	Axe	3	கோடாரி	3
9	Ail	3	வருந்து	4
10	Aim	3	நோக்கம்	4
11	Any	3	யாராவது	4
12	Awl	3	குத்தூசி	4

13	All	3	அனைத்து, முழுமையான	4,5
14	Ask	3	கேளுங்கள்	5
15	Apt	3	பொருத்தமான	6

The mathematical formula for calculating the average of English word length is given by:

$$\mu 1 = \frac{1}{n} * \sum_{i=1}^{N} x_i$$

μ_1 is the average word length of all the English words for the alphabet A.
N & n is the total number of English words for the alphabet A.
Xi is the mean length of each English Word.

The mathematical formula for calculating the average of Tamil word length is given by:

$$\mu 2 = \frac{1}{n} * \sum_{i=1}^{N} x_j$$

μ_1 is the average word length of all the English words for the alphabet A.
N & n is the no. of word samples.
Xj is the mean length of each equivalent word.
μ_2 is the mean word length of the Tamil words.

After calculations, it is observed that the average of English word for length μ1 is 3 and its Equivalent Tamil word μ2 is 3.533.

5. CALCULATION OF MEAN OF ENGLISH WORDS CONTAINING ELEVEN ALPHABETS

Similarly when the same mean calculation for a word length of eleven is repeated in an English word and its Equivalent Tamil word, an average word length for English is obtained as 11 and for Tamil it is obtained as 5.62. It is clearly observed from the above values that the mean word length of Tamil words is lesser than that of English words.

Table2. Sample data for English word of length eleven with its Equivalent Tamil word Mean

S.No	English Alphabets	Length of the English word	Equivalent Tamil Meaning	Length of the Tamil word
1	Achievement	11	சாதனை	3
2	Affiliation	11	தொடர்பு	4
3	Arrangement	11	ஏற்பாடு	4
4	Accommodate	11	இடங்கொடு	5
5	Affirmation	11	உறுதிமொழி	5
6	Agriculture	11	விவசாயம்	5
7	Anniversary	11	ஆண்டு விழா	5
8	Appointment	11	நியமனம்	5
9	Appropriate	11	அதற்கான	5
10	Archaeology	11	தொல்லியல்	5
11	Adventurous	11	சாகசங்கள்	6
12	Altercation	11	வாய்ச்சண்டை	6
13	Archipelago	11	தீவுக்கூட்டம்	7

14	Aristocracy	11	பிரபுத்துவம்	7
15	Anesthesia	11	மயக்க மருந்து	8
16	Ameliorate	11	சீர்படுத்து	10

6. RESULTS AND DISCUSSION

Table 2 shows the graph of the average values of English and Tamil word of various lengths ranging from two to thirteen for the alphabet 'A'. It is observed that the average length of English word is more than the Tamil word.

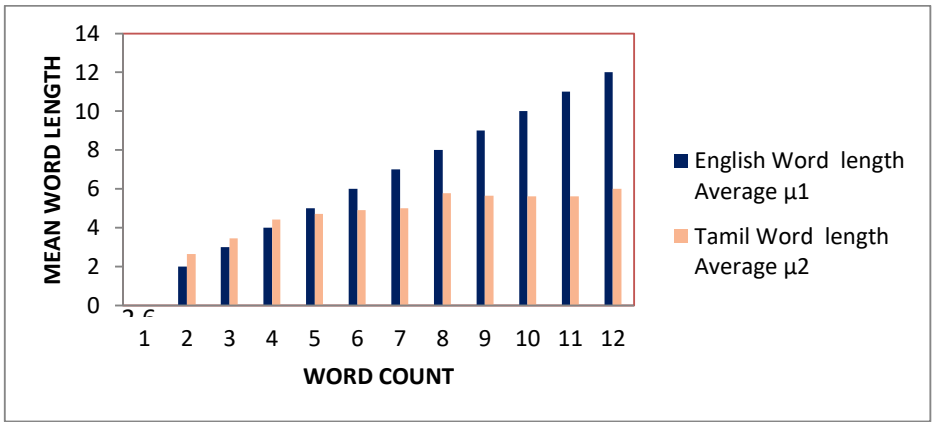


Fig1. Graph for English and Tamil word length Average values Calculated from the dataset

7. STATISTICAL ANALYSIS OF ENGLISH WORDS LENGTH OVER TAMIL

It is observed from the Table 3 that $\mu_2 < \mu_1$, the mean word length of English words is greater than equivalent Tamil words for a word length of 3 with all the alphabets starting with the letter 'A'.
$\mu_2 > \mu_1$, the mean word length of Tamil words is greater than English words for a word length of 3 and above with all the alphabets starting with the letter 'A'.

Table 3. Tabulation with calculated T and P values for various word lengths

S.NO	N	μ_1	μ_2	SD1	SD2	T VALUE	$H_0 = \mu_1 - \mu_2$	$H_A \neq \mu_1 - \mu_2$	P Value
1	3	2	2.66	0	1.157	-1	Accepted		0.3739
2	15	3	3.53	0	1.402	-1.3	Accepted		0.186
3	21	4	4.42	0	2.204	-1.9	Accepted		0.378
4	77	5	4.71	0	0	1.5		Accepted	0.1184
5	105	6	4.92	0	1.523	7.2		Accepted	Less than 0.0001
6	78	7	5	0	1.6375	5		Accepted	Less than 0.0001
7	76	8	5.78	0	2.276	8.5		Accepted	Less than 0.0001
8	84	9	5.66	0	2.3189	11.2		Accepted	Less than 0.0001
9	48	10	5.62	0	2.0795	14.6		Accepted	Less than 0.0001

10	14	11	5.62	0	2.683	12.6		Accepted	Less than 0.0001
11	6	12	6	0	0.5774	5.5		Accepted	Less than 0.0001
12	4	13	4.5	0	0.5774	29.4		Accepted	Less than 0.0001
13	1	15	15	0	4	-		Accepted	

μ_1 - Average Mean length of English words in the dataset
μ_2 - Average Mean length of Tamil words in the dataset

Table 4 has taken into account all the 524 words given in the dictionary for the letter 'A' and its equivalent Tamil words were analysed to test our hypothesis and the results of statistical calculations are given in Table 4.

Table 4. Tabulation with calculated T and P values for N = 524

Parameters	English words from Alphabet A	Tamil Equivalents words from English Alphabet "A"
Total No. of words, N	524	524
Mean	μ_1 = 7.185	μ_2 = 5.153
Standard deviation	2.1048	1.9603
T score	16.1721	
P Value	Less than 0.00001	

It is observed from Table 2 that the average mean value for length of English word is greater than the Tamil word length. From Table 3 it is statistically observed from the T and P values that the length of the English word length is significantly greater than the Tamil word length. For the total number of 524 words in the dataset the mean average value of English word length is 7.18 and Tamil word length is 5.15 with a T score of 16.1721.

8. SAMPLE MEMORY CALCULATION

Consider an example sentence 1

All the glitters are not gold

மின்னுவதெல்லாம் பொன்னல்ல.

For the above example sentence the total number of English words is 24 but the number of Tamil words is 13 which proves that the number of words in the English sentence is greater than the number of words in the Tamil sentence.

Consider an example sentence 2

A single tree does not make an orchard

ஒரேஒரு மரம் பழத்தோட்டம் ஆகாது

Similarly consider the second sentence consisting of 31 English words totally whereas its equivalent Tamil translation contains only 18 words proving the same concept. According to ASCII encoding scheme the memory space required to store a character is one byte. Each byte. is equal to 8 bits. For both English and Tamil 8bits of memory is required to store a character.

9. MEMORY CONSUMPTION OF ENGLISH AND TAMIL SENTENCES

It is also observed from Table 5 that Tamil sentences require less amount of memory when compared with the English sentences This proves that English word length is greater than the length of Equivalent Tamil word with a level of significance of 99.99%.

Table 5. Tabulation for Memory consumption for English and Tamil sentences

PARAMETERS	ENGLISH	TAMIL
Total Number of characters present in the sentence 1	24	13
Memory space required to store sentence 1	192 bits	104 bits
Total Number of characters present in the sentence 2	31	18

Memory space required to store sentence 2	248 bits	144 bits

10. CONCLUSION

From the above results, it is evident that predominantly every English word has an equivalent Tamil word with mean word length in the latter being less than the mean word length in the former. . Hence the misconception about Tamil language as being very complex, when compared to English proves wrong. Because of lesser word length in Tamil, the memory space taken in a computer system will be considerably less for Tamil computing when compared with English. This intends to extend to all alphabets of English to prove that though the number of alphabets in Tamil is far greater than the number of alphabets in English and the total number of dictionary words in Tamil is very voluminous when compared with the dictionary words in English, still Tamil word length proves to be lesser than English word length in terms of computing.

ACKNOWLEDGEMENT

The authors would like to thank Ms.Chamundeeswari.G, who was working as Asst.Professor in DCSE, SRM IST for compiling the dataset of English words and Tamil equivalent words.

REFERENCES

1. Shobhit Tiwari, Sourav Khandelwal, Sanjiban Sekhar Roy, "E-Learning Tool for Japanese language learning through English, Hindi and Tamil "A Computer Assisted Language Learning (CALL) based approach, 978-1-4673-0671-3/11/$26.00©2011, IEEE-ICoAC 2011.
2. Sunitha C, Dr.A.Jaya, " A phoneme based model for English to Malayalam Transliteration", 2015 International Conference on Innovation Information in Computing Technologies(ICIICT),Chennai,India
3. Ning Fang, Xiuli Zhao, " A Comparative study of learning style preferences between American and Chinese Undergraduate Students", 978-1-4673-5261-1/13/$31.00 ©2013 IEEE
4. Joshua R. Maximoff, Michael D. Trela, D. Richard Kuhn, Raghu Kacker, " A method for analyzing system state space coverage within a t wise testing framework" 2010 International Systems Conference doi:10.1109/systems.2010.5482481
5. Musabah Said Musabah Al Buraiki1, Dawood Al-Hamdani*2, " Examining the effectiveness of using online activities in Learning English Vocabulary", 2017 ICTA.
6. Chun-Hui Wu1, Ta-Cheng Chen1, , Yih-Her Yan2, Chun-Feng Lee1. " Developing an Adaptive e-learning System for learning excel" Proceedings of the 2017 IEEE International Conference on Applied System Innovation
7. Chin Jung Huang1, , Fa Ta Tsai2, , Hui Chiu Chen3, "A study on the app assisted English Vocabulary Learning", Proceedings of the 2017 IEEE International Conference on Information, Communication and Engineering IEEE-ICICE 2017 - Lam, Meen & Prior (Eds)
8. M. Kocaleva*, N. Stojkovikj*, A. Stojanova*, A. Krstev*, B. Zlatanovska, "Improving on teaching curriculum of calculus 2 at technical features", 978-1-5090-5467-1/17/$31.00 ©2017 IEEE 25-28 April 2017, Athens, Greece.
9. Su-Kyung, Young-Sun youn, Sang Jae Nam, Min-Ho son, Shin-Dug Kim, " Optimized Memory-Disk Integrated System with DRAM and Non Volatile Memeory",IEEE Transactions on Multi-Scale Computing Systems , Volume: 2 , Issue: 2 , April-June 1 2016.

கணினி வழி தமிழ் மொழி கற்றல் - கற்பித்தல்

(Computer-Aided Learning and Teaching of Tamil)

தமிழ் இணைய மாநாடு - உ0ககூ

சு.ஆனந்த பாரதி (இந்தியா)

முன்னுரை**:-**

தொன்மையும் தொடர்ச்சியும் வளர்ச்சியும் உடைய தமிழ்மொழியானது தமிழ்நாடு, இலங்கை, மலேசியா, சிங்கப்பூர் ஆகிய நாடுகளில் தாய்மொழியாக, முதல்மொழியாகக் கற்றுக்கொடுக்கப்பட்டுவருகிறது. தற்போது மொரீஷியஸ், பிஜி போன்ற நாடுகளிலும் அங்கு வாழும் தமிழர்கள் தமிழ்க் கல்விக்கு முக்கியத்துவம் அளித்துவருகின்றனர். அமெரிக்கா, ஐரோப்பிய நாடுகளிலும் புலம்பெயர்ந்த தமிழர்கள் தங்கள் குழந்தைகளுக்குத் தமிழைக் கற்பிக்கப் பல்வேறு முயற்சிகளை மேற்கொண்டுவருகின்றனர்.

தமிழ்மொழியானது இவ்வாறு முதல்மொழியாகவும்,இரண்டாம்மொழியாகவும், மூன்றாம் அல்லது அந்நியமொழியாகவும் உலகெங்கும் கற்பிக்கப்பட்டுவருகிறது. ஒரு மொழியை முதல்மொழியாகக் கற்பிப்பதற்கும் இரண்டாம்மொழியாகக் கற்பிப்பதற்கும் மூன்றாம்மொழியாகக் கற்பிப்பதற்கும் இடையில் வேறுபாடுகள் உண்டு. தமிழ்மொழிக் கல்விக்கான பாடத்திட்டம் தயாரிப்பதிலும், அவற்றைக் கற்பிக்கும் போதும் இந்த வேறுபாடானது மனதில் கொள்ளப்படவேண்டும்.

மொழித்தொழில்நுட்பம் **(Language Technology):-**

சமூக வாழ்க்கையில் நாம் நமது உணர்வுகளையும் கருத்துகளையும் தேவைகளையும் பிறர்க்குப் புலப்படுத்துவது மிகத் தேவையான ஒன்றாகும். இந்தப் புலப்படுத்தச் செயல்பாட்டின் (Communication)ஊடே நாம் வாழும் இயற்கையையும் சமூகத்தையும் புரிந்துகொண்டு, நமக்குத் தேவையான வகையில் மாற்றி,பயன்படுத்திக்கொள்கிறோம். செயல்முனைப்புள்ள இக்கருத்துப் புலப்படுத்தச் செயல்பாடே கருத்தாடல் (Discourse) என்றழைக்கப்படுகிறது. நாம் மேற்கொள்கிற கருத்தாடலில் மிகப் பெரிய பங்கு ஆற்றுவது நமது இயற்கைமொழியே ஆகும். இயற்கைமொழியோடு (Verbal) நமது உடல் அசைவுகள், முகத்தோற்றங்கள், சைகைகள், குறியீடுகள், படங்கள் போன்ற மொழிசாராக் கூறுகளும் (non-verbal) நமது கருத்தாடலில் பங்கேற்கின்றன.

பேச்சுமொழியாக இருந்த பல மொழிகள் தமது வரலாற்றில் எழுத்துமொழிகளாகவும் வளர்ந்தன. மனிதன் தனது கருத்துகளைக் காலம், இடம் கடந்து நீடிப்பதற்காக எழுத்துகள்மூலம் பதிவு செய்யத் தொடங்கியதையொட்டி எழுத்துவழக்குகள் தோன்றி வளர்ந்தன. கல்வெட்டுகள், ஓலைச்சுவடிகள், இன்றைய அச்சுப் படைப்புகள் அனைத்துமே நமது கருத்துகளைப் பதிவுசெய்து, காலம், இடம் கடந்து பிறருக்குத் தெரிவிப்பதற்காக நாம் உருவாக்கிய மொழித் தொழில்நுட்பங்களே (Passive Language Technology) ஆகும்.

தொழில்நுட்பமும் அறிவியலும் **(Technology and Science):-**

தொழில்நுட்பக் கருவிகள் என்பவை நமது இயற்கைத் திறன்களின் செயல்பாடுகளை மென்மேலும் விரிவாக்கும் கருவிகளே ஆகும் (enhancement of human abilities) . நமது கண் பார்வைத் திறனை விரிவாக்கும் கருவிகளே நுண்நோக்கி (Microscope) , தொலைநோக்கி(Telescope) ஆகியவை ஆகும். நமது வாய் மூலம் உருவாக்கும் ஒலிகளை வலுப்படுத்தவும் நெடுந்தூரம் எடுத்துச் செல்லவும் பயன்படும் கருவிகளே ஒலிபெருக்கி, தொலைபேசி, அலைபேசி ஆகியவை ஆகும். வேகமாகவும் விரைவாகவும் இடம்விட்டு இடம்பெயர நமக்குப் பயன்படும் பொறிகளே மிதிவண்டி, பேருந்து,தொடர்வண்டி ஆகியவை ஆகும். பறவைபோன்று வானத்தில் பறந்துசெல்ல நாம் உருவாக்கிய பொறியே விமானமாகும்.அறிவியல்

அறிந்துகூறுகிற உண்மைகளின் அடிப்படையில் தொழில்நுட்பம் நமக்கு பலவகைக் கருவிகளையும் பொறிகளையும் உருவாக்கித் தருகிறது.

இயற்கையின் பண்புகளை அறிந்து கூறும் இயற்பியல், வேதியியல்,உயிரியல் போன்ற அறிவியல்துறைகள் போன்றதே நமது இயற்கைமொழித் திறனைப் பற்றிக் கூறும் மொழியியல் துறையாகும் (Linguistics) . அதுபோன்று , நமது இயற்கைமொழித் திறனை மேலும் விரிவாக்க அல்லது வளர்த்தெடுக்க நாம் உருவாக்கிய மொழித்தொழில்நுட்பமே முன்னர் குறிப்பிட்ட எழுத்துமொழி, கல்வெட்டு, ஓலைச்சுவடி, அச்சு ஆகியவை ஆகும்.

செயல்முனைப்புள்ள பொறிகள் **(Active Technology oriented Machines):-**

இன்றைய அறிவியல் , தொழில்நுட்பம் ஒரு அடுத்த கட்ட, உயர் வளர்ச்சியை எட்டியுள்ளது. நாம் உருவாக்குகிற கருவிகளும் பொறிகளும் நமது கட்டளைகளை ஏதோ ஒரு வகையில் புரிந்துகொண்டு, நமது இடுகிற கட்டளைகளைச் செயல்படுத்துகின்றன. ஒரு குறிப்பிட்ட இயற்பியல் தூண்டுதலுக்கு ஒரு குறிப்பிட்ட செயலைப் பொறிகள் மேற்கொள்கின்றன அல்லது தருகின்றன (Stimulus - Response). நமது புவியில் இருந்துகொண்டே செவ்வாய்க் கிரகத்திற்கு ஆய்வுக் கருவிகளை அனுப்பி, நாம் செய்ய விரும்புகிற பணிகளை அவற்றிற்குப் புலப்படுத்தி, அவற்றைச் செயல்பட வைக்கிறோம். புரிந்துகொள்ளும் திறனற்ற பொறிகள் (Passive Technology Machines) என்ற நிலைமாறி,இன்று புரிந்துகொண்டு செயல்படும் பொறிகள் (Active Technology Machines) - ரோபோட்கள் - என்ற வளர்ச்சிநிலையை நமது தொழில்நுட்பம் பெற்றுள்ளது.

புரிந்துகொண்டு செயல்படும் பொறிகள் என்று கூறும்போது, எவ்வாறு புரிந்துகொள்கின்றன என்ற வினா நம் முன் நிற்கிறது. ஒலி, ஒளி , மின்னோட்டம் , மின்தடை போன்ற இயற்பியல் (Physical) கூறுகளைக் புலப்படுத்தக் கூறுகளாகக் கொண்டு, பல கருவிகள், பொறிகள் செயல்படுகின்றன. ஆனால் இன்றைய அறிவியல், தொழில்நுட்ப வளர்ச்சி அடுத்த கட்ட நிலையை எட்டியுள்ளது. இயற்பியல் கூறுகளோடு, நாம் பயன்படுத்தும் இயற்கைமொழிகளையும் (Natural languages) நம்மைப் போன்றே புரிந்துகொண்டு செயல்படும் பொறிகள் உருவாக்கப்பட்டுவருகின்றன.

கணினியில் மொழித்தொழில்நுட்பம் **(Language Technology in Computer):-**

இந்த உயர்ந்த கட்ட வளர்ச்சிக்கு அடிப்படையானதுதான் கணினியியல் துறையாகும். கணினி என்பது நமது கட்டளைகளை மொழிகள் வாயிலாகப் புரிந்துகொண்டு செயல்படும் ஒரு பொறியாகும். இன்று இந்த மொழிகள் கணினிக்கென்று உருவாக்கப்படுகிற செயற்கைமொழிகளாக (சி, சி+, ஜாவா, பைத்தான் போன்ற மொழிகளாக) அமைந்துள்ளன. இருப்பினும் இந்தச் செயற்கைமொழிகள் வாயிலாக நாம் இடுகிற கட்டளைகளைக் கணினிப் பொறி புரிந்துகொண்டு, நாம் இடுகிற பணிகளை நிறைவேற்றுகின்றன. எனவே இது கணினிக்கென்று உருவாக்கப்பட்டுள்ள செயற்கைமொழித் தொழில்நுட்பமாகும்.

அடுத்த கட்ட வளர்ச்சியாக , நமது இயற்கைமொழிகளின் வாயிலாகவே நமது கட்டளைகளைக் கணினி புரிந்துகொண்டு செயல்படுவதற்குத் தேவையான அறிவியலும், தொழில்நுட்பமும் இன்று வளர்ந்துவருகின்றன. கணினியியல் நோக்கில் நமது இயற்கைமொழிகளை ஆராயும் அறிவியலே கணினிமொழியியலாகும் (Computational Linguistics). அதன் அடிப்படையில் உருவாக்கப்படுகிற தொழில்நுட்பமே கணினிக்கேற்ற மொழித்தொழில்நுட்பமாகும் (Active Language Technology).

புதியவகை மொழித்தொழில்நுட்பம்**:-**

நாம் முன்னர் குறிப்பிட்ட கல்வெட்டு, ஓலைச்சுவடி, அச்சு போன்றவையும் மொழித்தொழில்நுட்பங்களே என்றாலும் அவற்றிற்கும் கணினிவழி மொழித்தொழில்நுட்பத்திற்கும் அடிப்படை வேறுபாடு உள்ளது. முந்தைய மொழித்தொழில்நுட்பங்கள் எல்லாம் நமது மொழிவழிக் கருத்துகளைப் பதிந்துவைக்கும் தொழில்நுட்பமாகவே (Passive Language Technology) அமைகின்றன. பதிந்துவைக்கிற கருத்துகளைப் புரிந்துகொண்டு செயல்படும் மொழித்தொழில்நுட்பம் அல்ல அவை. ஆனால் கணினிவழி மொழித்தொழில்நுட்பத்தின் அடிப்படையே நாம் மொழிவழி தெரிவிக்கிற கருத்துகளைப் புரிந்துகொண்டு, செயல்படும் செயல்முனைப்புள்ள மொழித்தொழில்நுட்பமாகும் (Active Language Technology) .

நாம் நமது இயற்கைமொழிவாயிலாக முன்வைக்கப்படுகிற கருத்துகளைப் புரிந்துகொண்டு செயல்படுவதுபோல, கணினியும் புரிந்துகொண்டு செயல்படவேண்டும். பேச்சு வழியாகவோ எழுத்துவழியாகவோ நாம் கருத்துகளை முன்வைக்கும்போது,கணினியானது மொழித்தொடர்களை ஆய்ந்து, கருத்துகளைப் பிரித்தெடுக்கும் திறனைப் (Natural Language Processing - NLP) பெறவேண்டும். நம்மைப் போன்றே மொழிவழிச் செயல்பாடுகளை மேற்கொள்ளவேண்டும்.

கணினித்தமிழ் வளர்ச்சியில் இரண்டு கட்டங்கள்**:-**

மேற்கூறிய கருத்துகளின் அடிப்படையில் நோக்கினால், கணினித்தமிழ் என்பது தமிழ்மொழியைக் கணினி போன்ற மின்னணு சாதனங்கள் கற்றுக்கொண்டு, அதனடிப்படையில் நாம் தமிழில் இடுகிற கட்டளைகளைப் புரிந்துகொண்டு செயல்படுவதே ஆகும்.

கணினித்தமிழின் முதல் கட்டம், தமிழ்மொழியைக் கல்வெட்டு,ஓலைச்சுவடி, அச்சு ஆகியவற்றில் பதிவதுபோல, கணினியில் பதிவு செய்வதாகும். தமிழ் எழுத்துருக்களை உருவாக்குவது.விசைப்பலகைகளை உருவாக்குவது, இணையதளம், மின்னஞ்சல், வலைப்பூக்கள், விக்கிபீடியா போன்றவற்றில் தமிழை இடம்பெறச்செய்வது போன்ற செயல்களை மேற்கொள்வதாகும்.இவற்றில் தற்போது நாம் குறிப்பிட்ட அளவு வெற்றியடைந்துள்ளோம். இந்த வெற்றியில் ஒருங்குறி (Unicode) வருகை, விசைப்பலகை தரப்படுத்தம் ஆகியவை நமக்கு பெரிதும் உதவியுள்ளன.

கணினித்தமிழின் அடுத்த உயர்ந்த கட்டம், கணினிக்குத் தமிழ்மொழியைக் கற்றுக்கொடுத்து, நமது பல்வேறு மொழித் தொழில்நுட்பத் தேவைகளை நிறைவேற்றிக்கொள்வதாகும். சொல்திருத்தி, இலக்கணத் திருத்தி , எழுத்து - பேச்சுமாற்றி (Text to Speech - TTS) , பேச்சு - எழுத்துமாற்றி (Automatic Speech Recognizer - ASR), ஒளிவழி எழுத்தறிவான் (Optical Character Recognizer - OCR),இயந்திரமொழிபெயர்ப்பு (Automatic Machine Translation - MT) போன்ற மொழித்தொழில்நுட்பங்கள் தமிழில் பெருகவேண்டும்.

கணினிக்குத் தமிழ் கற்பித்தல் **(Teaching Tamil to Computer):-**

இதற்கு முதல் அடிப்படை, தமிழ்மொழி அறிவைக் கணினிக்குக் கொடுப்பதேயாகும். மனிதமூளை இயற்கைமொழியைக் கற்றுக்கொண்டு, பயன்படுத்துவதற்கும், கணினி கற்றுக்கொண்டு,பயன்படுத்துவதற்கும் வேறுபாடுகள் உண்டு. மனிதமூளை ஒரு உயிரி-வேதியியல் உறுப்பாகும் (bio - chemical organ). கணினி அப்படிப்பட்டதல்ல. அது மின்னணுவியல் அடிப்படையில் அமைந்துள்ள ஒரு பொறியேயாகும் (Electronic Chip) . எனவே ஒரு இயற்கைமொழியை மனிதமூளைக்குக் கற்றுக்கொடுப்பதற்காக உருவாக்கப்படுகிற இலக்கணம், அகராதி போன்றவற்றை அப்படியே கணினிக்குக் கொடுத்து, அதைக் கற்றுக்கொள்ளச் செய்யமுடியாது.கணினியின் திறனுக்கேற்றவாறு மொழியைக் கற்றுக்கொடுப்பதில் இரண்டு கருத்துகளைக் கவனத்தில் கொள்ளவேண்டும்.

- ஒன்று, தமிழ்மொழி இலக்கணத்தைக் கணினிக்கேற்ற கணிதவாய்பாடுகளாக (mathematical formulae) மாற்றிக் கொடுப்பதாகும். சுழியம், ஒன்று என்ற இரண்டு எண்களின் அடிப்படையில்தான் கணினி எந்தவொரு அறிவையும் பெற்றுக்கொள்கிறது. இந்த இரண்டு எண்களைப் பலவகைகளில் கையாளுவதன்மூலமே கணினி தனக்கு இடப்படுகிற எந்தவொரு கட்டளையையும் செயல்படுத்துகிறது. மொழி அறிவும் எண்களின் அடிப்படையில் அமைகிற கணினி வாய்பாடுகளாக அமையும்போதுதான், கணினியால் அதைக் கையாளமுடியும்.எனவே மனித மூளைக்காக உருவாக்கப்பட்டுள்ள தொல்காப்பியம்,நன்னூல் போன்ற இலக்கண நூல்களைக் கணினிக்கேற்ற கணிதவழி இலக்கணமாக (Computational Tamil Grammar) மாற்றிக், கணினிக்கு அளிக்கவேண்டும்.
- மற்றொன்று, நாம் நமது மொழியைக் கையாளும்போது,உலகத்தைப்பற்றிய நமது பின்புற அறிவு (Pragmatic Knowledge)மிகவும் உதவுகிறது. அதன் பயனாக, சொல்லிலோ அல்லது தொடரிலோ பொருள் மயக்கம் (Word Sense Ambiguity) ஏற்பட்டால்,பேசப்படுகிற பொருள், பேசும் சூழல் போன்றவை நமக்கு உதவி செய்து, பொருள்

மயக்கத்தைத் தவிர்க்கின்றன (Disambiguation) .கணினிக்கு இதுபோன்ற பின்புல அறிவு இன்று அமையவில்லை.எனவே, அதனால் நம்மைப்போன்று பொருள் மயக்கத்தைத் தவிர்க்கமுடியாது. இச்சூழலில் பொருள் மயக்கத்தைத் தவிர்ப்பதற்குக் கணினிக்குப் பலவகைகளில் நாம் உதவவேண்டும்.

மேற்கூறிய இரண்டையும் கையாளுவதற்கான உத்திகளைத் தருவதே கணினிமொழியியல் என்ற ஒரு அறிவியல் துறையாகும்.இந்த அறிவியல்துறையின் அடிப்படையில் தமிழ்மொழி அறிவு (இலக்கணம், சொற்களஞ்சிய அறிவு போன்றவை) கணினிக்கேற்ற தமிழ்மொழி அறிவாக மாற்றியமைக்கப்படவேண்டும். இதுவே கணினித்தமிழ் இலக்கணமாகும்.

கணினிவழித் தமிழ் ஆய்வு **(Computational Tamil Research):-**

பேச்சொலி, எழுத்து, சொல், தொடர், பனுவல், பொருண்மை என்று பல நிலைகளில் தமிழ்மொழி அமைப்பு கணினிமொழியியல் நோக்கில் ஆராயப்படவேண்டும். அப்போதுதான் தமிழ்மொழியைக் கணினிக்குக் கற்றுக்கொடுக்க முடியும். இவ்வாறு கற்றுக்கொடுக்கும் போதுதான், அதனடிப்படையில் நமக்குத் தேவையான தமிழ்மொழித் தொழில்நுட்பங்களை (Tamil Language Technology) நம்மால் உருவாக்கமுடியும். தமிழ் மென்பொருள்கள் உருவாகும்.

மேற்கூறிய கணினித்தமிழ் இலக்கண உருவாக்கப்பணிகளில் முதலாவது, தரவுமொழியியல் (Corpus Linguistics) அடிப்படையிலான தமிழ் ஆய்வாகும். சங்கத் தமிழிலிருந்து இன்றைய தமிழ்வரை , தமிழின் இலக்கணத்தையும் சொற்களஞ்சியத்தையும் பெறுவதற்குத் தமிழ்த்தரவகம் (Tamil Corpus) உருவாக்கப்படவேண்டும். உருவாக்கப்பட்ட தமிழ்த் தரவகத்தின் அடிப்படையில் தமிழ்மொழியின் அமைப்பு (இலக்கணம்) எழுத்து, சொல், தொடர் போன்ற அனைத்து நிலைகளிலும் ஆராயப்படவேண்டும். உருபன் பகுப்பாய்வி (Morphological Parser), தொடரன் பகுப்பாய்வி (Syntactic Parser) போன்ற பல ஆய்வுக் கருவிகள் தமிழுக்கு உருவாக்கப்படவேண்டும்.

கணினிமொழியியல் அடிப்படையில் இன்றைய எழுத்துத் தமிழை எழுத்து, சொல் நிலைகளில் ஆராயும் உருபன் பகுப்பாய்வியை பேராசிரியர்.ந.தெய்வசுந்தரம் மற்றும் அவர்தம் குழுவினர்கள் இணைந்து உருவாக்கியுள்ளார். அதன் பயனாக இன்று சில மொழி பதிப்புக் கருவிகளை உருவாக்க முடிந்துள்ளது. குறிப்பாக, சொற்பிழைதிருத்தி, சந்திப்பிழைதிருத்தி , அயல்மொழிச்சொல் தவிர்ப்பு போன்ற சில தமிழ்மொழிக்கான மென்பொருள் கருவிகளை உருவாக்க முடிந்துள்ளது. தமிழ் உருபன் பகுப்பாய்வியைத் தொடர்ந்து, தமிழ்த்தொடரன் பகுப்பாய்வி உருவாக்கும் முயற்சி மேற்கொள்ளப்பட்டுவருகிறது.இந்த முயற்சி வெற்றியடையும்போது, தமிழ்த்தொடரியல் அடிப்படையில் தொடர் இலக்கணப் பிழைகளைத் திருத்தும் மொழிக்கருவியும் உருவாக்கமுடியும்.

தமிழ்ச்சொற்களஞ்சிய அறிவையும் கணினிமொழியியல் அடிப்படையில் கணினிக்கு அளிக்கப்படவேண்டும். அத்தோடு, தமிழ்மொழியின் ஒலியியல் அடிப்படைகளைப் பற்றிய ஆராய்ச்சி மிக முக்கியமானதாகும். தமிழ் எழுத்துகளைப் பேச்சொலிகளாக மாற்றி வாசித்துக்காட்டும் திறனையும், அதுபோன்று நமது பேச்சைக் கணினி கேட்டு , அதை எழுத்தில் எழுதிக்காட்டும் திறனையும் கணினிக்கு அளிப்பதற்கு இந்த ஆய்வு தேவையாகும்.

மேற்கூறிய அடிப்படை தமிழ்மொழி ஆராய்ச்சிப் பணிகள் வெற்றிகரமாக முடிவடைவதையொட்டியே, தமிழ்மொழிக்கான கணினிமொழித்தொழில்நுட்பம் வளர்ச்சியடையமுடியும்.

கணினிவழி தமிழ்க்கல்வி **(Computer Assisted Tamil Language Teaching):-**

கணினியில் மொழித்தொழில்நுட்பம் பெற்றுள்ள வளர்ச்சியின் பயனாக இன்று மாணவர்களுக்குத் தமிழ்மொழி கற்பித்தலில் கணினிக்குக் குறிப்பிடத்தக்க பங்கு உள்ளது. தமிழ் இலக்கணத்தையும் சொற்களஞ்சியத்தையும் மாணவர்களுக்குக் கற்றுக்கொடுப்பதில் ஆசிரியருக்குத் துணையாகக் கணினி

செயல்படமுடியும். மொழி கற்றலில் இடம்பெறும் பல்வேறு செயல்பாடுகளில் கணினியானது மாணவர்களோடு ஊடாட்டம் (Interaction) புரிந்து, அவர்களுக்கு உதவமுடியும். இரண்டுவகைகளில் கணினியைப் பயன்படுத்தலாம். ஒன்று, தமிழ் இலக்கணத்தை நேரடியாகக் கற்றுக்கொடுக்கவும் பயன்படுத்தலாம். மற்றொன்று, அவ்வாறு ஆசிரியர் மாணவருக்குக் கற்றுக்கொடுத்தபிறகு, மாணவரின் மொழிப் பயன்படுத்தத்தை வளர்ப்பதற்காகவும் பயன்படுத்தலாம்.

மேற்கூறிய இரண்டு பயன்பாடுகளுக்கும் பேராசிரியர்.ந.தெய்வசுந்தரம் அவர்கள் சில மென்பொருள்களை உருவாக்கியுள்ளார். மென்தமிழ் ஆய்வுத் துணைவன் (Mentamizh Research Companion) என்ற தமிழ் மென்பொருள் மாணவருக்குத் தமிழின் எழுத்து, அசை, சொல் , சந்தி ஆகியவைபற்றிய இலக்கணத்தையும் தமிழ்ப் பனுவல்களில் சொற்கள் பயன்படும் முறை பற்றியும் கற்றுக்கொடுக்கும் ஒரு மென்பொருள் ஆகும். தமிழ்மொழி ஆய்வாளர்களுக்கும் பல வகைகளில் பயன்படும் மென்பொருளாகும் இது. அதன் அடிப்படையில் மாணவர்கள் தங்களது எழுதும் திறனை வளர்த்துக்கொள்ளக் கணினிவாயிலாக உரைகளைத் தயாரிக்கும்போது, அவர்களுக்கு உதவும் சொற்பிழை திருத்தி, சந்திப்பிழைதிருத்தி, அயற்மொழிச்சொல்சுட்டி, அகராதிகள் போன்ற பல கருவிகளை உள்ளடக்கிய மென்தமிழ் என்ற ஒரு தமிழ்ச்சொல்லாளரையும் (Tamil Wordprocessor) உருவாக்கியுள்ளார்.

இக்கட்டுரையில் மென்தமிழ்ச் சொல்லாளரைப் பயன்படுத்தி,எவ்வாறெல்லாம் மாணவர்கள் தங்கள் எழுதும்திறனை வளர்த்துக்கொள்ளலாம் என்பதுபற்றி விளக்கம் அளிக்கப்படுகிறது.

1. தமிழில் மாணவர்களுக்கு ஒலி ஒற்றுமையுடைய ல,ள,ழ எழுத்துகள், ந,ன,ண எழுத்துகள், ர,ற எழுத்துகள் ஆகியவற்றைப் பயன்படுத்துவதில் குழப்பங்கள் ஏற்படலாம். அப்போது அவர்களுக்கு உதவிசெய்ய மயங்கொலிச்சொல் அகராதி ஒன்று கொடுக்கப்பட்டுள்ளது. மணம், மணம் என்ற இரண்டு சொற்களுமே தமிழ்ச்சொற்கள்தான். ஆனால் பொருள்வேறுபாடு உடைய இருவேறு சொற்கள் அவை. அதை எடுத்துக்காட்டும்வகையில் அகராதி அமைந்துள்ளது.
2. அடுத்து, சில சொற்களில் சரியான எழுத்துகளுக்குப் பதிலாக தவறான எழுத்துகளை மாணவர்கள் பயன்படுத்தலாம். அப்போது அது தவறு என்பதை எடுத்துக்கூறி, அச்சொல்லைத் திருத்த மென்தமிழ் உதவும். எடுத்துக்காட்டாக, தமிள், ஆங்கிளம், மளாயாஎன்று தவறாக எழுதப்பட்டிருந்தால், மென்தமிழ் அவற்றைத் தமிழ், ஆங்கிலம், மலாயா என்று திருத்தி அளிக்கும்.
3. மேலே குறிப்பிட்டுள்ள சொற்கள் வேர்ச்சொற்கள். அவற்றில் விகுதிகள் இல்லை. ஆனால் விகுதி ஏற்ற சொற்களில் தவறுகள் இருந்தாலும் அவற்றையும் மென்தமிழ் திருத்தி உதவும். தமிளுக்கு ,ஆங்கிளத்தில்தாண் , மலாயாவிளிருந்து என்று விகுதி ஏற்ற சொற்களில் எழுத்துப் பிழைகள் காணப்பட்டாலும், அவற்றையும் மென்தமிழ் திருத்தித் தரும்.
4. தமிழில் சொற்களுக்குள் உயிர் எழுத்தை அடுத்து உயிர் எழுத்து வரக்கூடாது. அவ்வாறு வரும்போது, அவற்றிற்கிடையில் உடம்படுமெய் தோன்றும். இதற்கு அடிப்படை தமிழ் அசையின் அமைப்புவிதிகளேயாகும். இதில் மாணவர்கள் தவறு செய்திருந்தால்,மென்தமிழ் தவறுகளைத் திருத்தித் தரும். தெருஆ, இலைஆ என்று சொற்கள் தவறாகக் காணப்பட்டால், மென்தமிழ் அவற்றைத்தெருவா, இலையா என்று திருத்தித் தரும்.
5. தமிழில் மகர ஒற்று இறுதி பெயர்ச்சொற்கள் வேற்றுமை விகுதிகளை ஏற்கும்போது, இடையில் அத்துச் சாரியை தோன்றும். அதுபோன்று நெடில்தொடர் குற்றியலுகரச் சொற்கள் வேற்றுமை விகுதிகள் ஏற்கும்போது இறுதி குற்றியலுகரம் பயின்றுவருகிற வல்லின எழுத்து இரட்டிக்கும். இந்த விதிகளில் மாணவர்கள் தவறு செய்திருந்தால், மென்தமிழ் அவற்றைத் திருத்தித் தரும். மரமை ,தரமோடு , வளமில் என்பதை மரத்தை , தரத்தோடு, வளத்தில் என்றும் நாடை , காடில், ஆடோடு என்பதை நாட்டை, காட்டில், ஆட்டோடு என்று மென்தமிழ் திருத்தித் தரும்.
6. தமிழ்ச்சொற்களில் அடிச்சொல்லோடு விகுதிகள் இணையும்போது, அகச்சந்தி என்றழைக்கப்படும் புணர்ச்சி விதிகள் சரியாகக் கையாளப்படவேண்டும். இல்லையென்றால் பொருளே மாறிவிடும். எடுத்துக்காட்டாக, கடை+கள் என்பதை கடைகள் (இங்கு கள் என்பது பன்மைவிகுதி) என்று எழுதினால் ஒரு பொருள். கடைக்கள்(இங்கு கள் என்பது ஒரு வேர்ச்சொல்) என்று எழுதினால் மற்றொரு பொருள். எனவே அகச்சந்திகளில் மாணவர்கள் தவறு செய்யக்கூடாது. அவ்வாறு தவறுகள் செய்யப்பட்டிருந்தால், மென்தமிழ் அவற்றைத் திருத்தித் தரும். பசுகள், பலாகள், தெருகள் என்று தவறாக எழுதப்பட்டிருந்தால் அவற்றைப் பசுக்கள், பலாக்கள், தெருக்கள்என்று திருத்தித் தரும். இரண்டு சொற்கள் இணைந்து வரும் தொகைகளில் நிகழும் சந்திப் பிழைகளைத்

தற்போது மென்தமிழ் மென்பொருளால் திருத்த இயலாது. அதற்கான முயற்சிகள் மேற்கொள்ளப்பட்டுவருகின்றன.

7. தமிழில் ஒரு சொல்லுக்கும் (நிலைமொழி) அதற்கு அடுத்த சொல்லுக்கும் (வருமொழி) இடையில் எழுத்துகள் தோன்றுவதும், திரிதலும்,மறைதலும் உண்டு. இதுவே புறச்சந்திகள் என்று அழைக்கப்படுகின்றன. புறச்சந்தியில் மாணவர்கள் தவறு செய்தால், மென்தமிழ் பிழைகளைத் திருத்துவதில் மாணவர்களுக்கு உதவும். படித்து பார்த்தான், படிக்க பார்த்தான், அவனை பார், அவனுக்கு கொடு, மிக பெரிய போன்ற தவறான தொடர்களைப் படித்துப் பார்த்தான், படிக்கப் பார்த்தான், அவனைப் பார், அவனுக்குக் கொடு, மிகப் பெரிய என்று மென்தமிழ் திருத்திக் கொடுக்கும். தவறாக ஒற்றுகளை மாணவர்கள் இட்டிருந்தாலும் மென்தமிழ் அவற்றைத் திருத்தித் தரும். படித்தப் பையன், வந்துப் பார்த்தான் என்று தவறாக மாணவர்கள் எழுதியிருந்தால் மென்தமிழ் அவற்றைப் படித்த பையன், வந்து பார்த்தான் என்று திருத்தித் தரும்.
8. மாணவர்கள் தமிழில் எழுதும்போது, பிறமொழிச்சொற்களைக் கலந்து எழுதியிருந்தால் அவற்றையும் திருத்தித் தரும் வசதி மென்தமிழில் இடம் பெற்றுள்ளது. வேர்ச்சொற்கள் மட்டுமல்லாமல் விகுதிகள் ஏற்ற சொற்களாக இருந்தாலும் மென்தமிழ் அவற்றைத் திருத்தித் தரும்.

   அயல்மொழிச் சொல் கலந்த உரை :

   நான் பஸ்ஸில் ஸ்கூலுக்குச் சென்றேன். அங்கு டீச்சரைப் பார்த்தேன். பின்பு கேண்டீனில் டிபன் சாப்பிட்டேன். அங்கிருந்துட்ரெயினில் யுனிவர்சிட்டிக்குப் போனேன். அங்கு புரபெசரைப்பார்த்தேன்.

   மென்தமிழ் திருத்திய உரை:

   நான் பேருந்தில் பள்ளிக்குச் சென்றேன். அங்கு ஆசிரியரைப் பார்த்தேன். பின்பு உண்பகத்தில் சிற்றுண்டி உண்டேன். அங்கிருந்து தொடர்வண்டியில் பல்கலைக்கழகத்துக்குப் போனேன். அங்கு பேராசிரியரைப் பார்த்தேன்.

9. மேலும் எண்களைத் தமிழ் எழுத்தில் எழுதிக் காட்டும் கருவியும் மென்தமிழில் இடம் பெற்றுள்ளது.

   98789 456346

   தொண்ணூற்றெட்டாயிரத்து எழுநூற்று எண்பத்தொன்பது, நான்குஇலட்சத்து ஐம்பத்தாறாயிரத்து முந்நூற்று நாற்பத்தாறு.

10. உரையில் உள்ள சொற்களை அகரவரிசைப்படுத்தவும், னகரவரிசைப்படுத்தவும் மென்தமிழில் வசதிகள் உண்டு. மேலும் சொல்லடைவும் தயாரிக்கலாம்.
11. தமிழ் - ஆங்கில அகராதி, ஆங்கிலம் - தமிழ் அகராதி , அயல்மொழிச்சொல் அகராதி என்று பல வகை அகராதிகளும் மாணவர்களுக்குப் பயன்படும் வகையில் தயாரிக்கப்பட்டுள்ளன.

முடிவுரை:-

இன்றைய கணினி உலகில் தமிழ்மொழியை மாணவர்கள் கற்கும்போது, அவர்களுக்கு உதவியாகப் பயன்படும் வகையில் மென்தமிழ் மென்பொருள் உருவாக்கப்பட வேண்டும். இன்றைய எழுத்துத் தமிழின் இலக்கணத்தைக் கணினிக்கு மென்பொருள் வாயிலாக மென்தமிழ் அளித்துள்ளது. அதன் அடிப்படையில் மாணவர்கள் பிழைகள் ஏதும் செய்திருந்தால், கணினியானது செயல்முனைப்புடன் அவற்றைத் திருத்தித் தரும். அவர்கள் நல்ல தமிழில் எழுதப் பயிற்சி அளிக்க மென்தமிழ் போன்ற தமிழ் மென்பொருள்கள் பயன்படும். மென்தமிழ் போன்ற பல பயன்தரு மென்பொருள்கள் வெளிவர இளைய தலைமுறையினரை ஊக்குவிக்க வேண்டும். தமிழ் இலக்கண அறிவை முழுமையாகக் கணினிக்குக் கொடுப்பதில் மேலும் பல ஆராய்ச்சிகள் மேற்கொள்ளவேண்டியுள்ளது. அதனடிப்படையில்தான் மென்தமிழ் மென்பொருளின் செயல்பாடும் வளர்ச்சியடையும்.

கு ம ர கு ரு

பொறியியற் கல்லூரி । வணிக மேலாண்மைப் பள்ளி । வேளாண்மைக் கல்லூரி
பன்முகக் கலை அறிவியல் கல்லூரி । சக்தி மேம்பாட்டுக் கல்வி மையம்

ஒழுக்கமே வாழ்வு

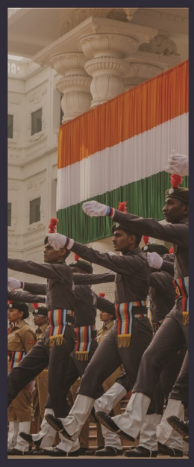
"கல்வியைப் போலவே தொழில்களும் கடவுளுக்குச் சமமானவை. படிக்கும் காலத்திலேயே தொழிலுக்கான பயிற்சியை மாணவர்கள் பெறவேண்டும்."

அருட்செல்வர் டாக்டர்.நா.மகாலிங்கம்









குமரகுரு பன்முக கலை அறிவியல் கல்லூரி

அடிப்படை அறிவியலிலிருந்து சமூக அறிவியல் வரையும், இலக்கியத்திலிருந்து நுண்கலைவரையும் அனைத்தையும் வழங்கி, சமூக முன்னேற்றத்திற்கான ஆய்வுகளை முன்னெடுத்து, வாழ்நாள் முழுதும் கற்றலில் ஆர்வம் கொண்டு, உலகியலின் போக்கில் தாக்கத்தை ஏற்படுத்தும் தனியாத ஆர்வம் கொண்ட இளைஞர்களை உருவாக்கும் கல்வி நிறுவனம்.

குமரகுரு தொழில்நுட்பக் கல்லூரி

150 ஏக்கர் வளாகத்தில் 6000 மாணவர்களுடனும் 400 பேராசிரியர்களுடனும் 34 ஆண்டுகளாக 'ஒழுக்கமே வாழ்வு' எனும் உயர் குறிக்கோளுடன், ஆராய்ச்சியையும் தொழில் இணைவையும் அடிப்படையாகக் கொண்டு தேசிய தர வரிசையில் முதல் நூறு நிறுவனங்களில் ஒன்றாகத் திகழ்கிறது.

கே.சி.டி. வணிக மேலாண்மைப் பள்ளி

தொழிலகங்கள் மற்றும் சமூகத்தின் முன்னேற்றத்தில் முனைப்புடன் செயல்படும் இளைய தலைமுறைக்கான தலைவர்களை உருவாக்கும் இந்தியாவின் முதல் 100 நிறுவனங்களில் ஒன்றாகும் கல்வி, மனித வளம், மாணவர்கள் சேவை, ஐ.டி. தொழில்நுட்பம், தொழிலினைவு, சமூக பங்களிப்பு போன்றவைகளில் மிகுந்த ஈடுபாட்டுடன் செயல்படும் இணையற்ற நிறுவனம்.

குமரகுரு வேளாண்மைக் கல்லூரி

தொழில்நுட்பம் மூலமாக வேளாண்மை உற்பத்தியைப் பெருக்கி, வேளாண்மையைத் தற்சார்புத் தொழிலாக மாற்றுவதற்குரிய வேளாண்மைக் கல்வியை, கிராமப்புற மாணவர்களுக்கு வழங்குவதை நோக்கமாகக் கொண்ட நிறுவனம்.

ஆராய்ச்சி அமைப்புக்கள்

மாணவர் ஆராய்ச்சிகள்

iQube
Innovate • Incubate • Incorporate

GARAGE

re KCT'S PLATFORM FOR RESEARCH & EXPLORATION

தொழிலக ஆராய்ச்சிகள்

KUMARAGURU KC.IRI CENTRE FOR INDUSTRIAL RESEARCH & INNOVATION

TIFAC

FORGE
போர்ஜ்

கே.சி.டி.யின் ஒர் அங்கமான 'போர்ஜ்' வளர்ந்து வரும் நவீனத் தொழில்நுட்பத்தையும், கணினி அறிவியலையும் இணைத்து ரோபோட் தானியங்கிஇயந்திரங்களையும் செயற்கை அறிவூட்டல் கருவிகளையும், டிஜிட்டல் வணிகத்தையும் வளர்த்தெடுத்துப் பல்வேறு தொழில் துறைகளிலும் இந்த நுண்ணறிவுப் பயன்பாட்டைக் கொண்டுசெல்லும் நோக்கில் செயல்படுகிறது.

குமரகுரு கல்வி வளாகம்
சரவணம்பட்டி
கோயம்புத்தூர்
0422 2661100
kct.ac.in
kctbs.ac.in
kclas.ac.in
kia.ac.in