Development of Tamil Spelling and Sandhi Checkers in a Crowd-Sourcing Environment

Vasu Renganathan
University of Pennsylvania vasur@sas.upenn.edu

1. Introduction

This paper discusses in detail the development of a Tamil Spelling and Sandhi checker application online in a Crowd-Sourcing environment. This resource as made available at: http://www.thetamillanguage.com/spellcheck is furnished with a comprehensive set of Tamil words in their lexical form along with a morphological tagger that is developed following the concepts and principles of the theory of Lexical Phonology. Any bilingual text with Tamil when provided in Unicode characters, this application is capable of checking both sandhi and spelling of the words. All of the inflected words are segmented using a morphological tagger to get their root form and correspondingly mapped against the lexical words that are stored in the electronic dictionary to make sure that they conform to the rules of Tamil morphology. The rules in the morphological tagger are derived in such a way that they follow the rules of the three level morphology as defined in the theory of Lexical phonology. When any inflected word is not recognized by the tagger, but are morphologically acceptable, they are stored in a custom database of inflected words for future use. This custom database that is updatable by the process of crowd-sourcing methods is constantly being monitored to insure that the words stored therein are valid Tamil words in all respects. Thus, the power of this Spell checker lies in the number of valid entries as stored in this custom database as well as the rules as included in the tagger. Thus far, this database contains about twenty five thousand words and this number is growing constantly based on the number of users who contribute to it. This paper while attempting to illustrate in detail the linguistic nuances and other practical issues that one would face while building spelling checkers for Tamil in general, it also tries to layout a method of crowd-sourcing as to how some of the obstacles while making spelling checkers can be resolved. The Sandhi checker application, on the other hand, takes into consideration only the external Sandhi rules as defined in the Tamil traditional grammars and attempts to make sure that the words belonging to particular categories inflect for external sandhi appropriately.

2. Building Morphological Taggers for Tamil

Although it is theoretically possible to parse all of the inflected words of Tamil programmatically, one would encounter into instances where the words are too difficult to parse due to their novel and non-standard form. When rules are written to parse words from right left, two types of such rules need to be kept in mind. One to identify suffixes in a number of subsequent steps without having to apply any morphological rules and the other to perform all the morphological sandhi rules in every step of the parsing process to arrive at the root form of the word. Thus, with words like *aṭippaṭaiyāka, aṭippaṭaiyākum,aṭippaṭaiyāṉa* etc., one can attempt to recognize suffixes such as *āka, ākum, āṉa* followed by the glide to get to the dictionary word *aṭippaṭai*. This can be illustrated based on the level ordered morphology as below:

{{W-L1}L2-0}(y)L3-*ākum/āṉa/āka}*

L1 indicates the word creation component, L2 is meant for all the case suffixes and subsequently L3, on the other hand, is meant for the adverbial and adjectival markers. In order to parse words in this fashion, one would need to know ahead of time all of the morphologically acceptable combinations of the word *aṭippaṭai* in all of these three levels. Inflections of words, in this sense, is infinite in nature and it would be unmanageable for one to come up with a list of all possible inflectional possibilities in any finite manner, nor can one categorically state that all of the L2 and L3 suffixes can be filled-in with all of the words. This means that the inflectional possibilities of words in Tamil is almost random in nature and theorizing the possibilities would be a futile task. In order to resolve this issue of infinite possibilities of inflected words, the present method is attempting to make use of the crowd-sourcing option, which allows us to store words in their inflected forms in the database. Consider that we include the above rules to parse words with finite set of suffixes conforming to the level ordered morphology as described in the theory of lexical phonology, accounting for the finite set of combinations is quite possible with any efficient morphological tagger. (Cf. Refer to Renganathan 2001 and Renganathan 2016 for a detailed description of how one can build a Tamil tagger following the principles of the theory of Lexical Phonology). However, the other type of words that can not be accounted for by such rules can be stored in a custom database and verified manually for their correctness. Following words that are not accountable by the above rules are stored in the custom database as below.

aṭippaṭaittuṟaikaḷai
aṭippaṭaiyā
aṭippaṭaiyākak
aṭippaṭaiyilāṉa
aṭippaṭaiyilāṉatu
aṭippaṭaiyilēyē
aṭippaṭaiyiliruntē
aṭippaṭaikkaaṉa

As one can note these words indicate the occurrences of case suffixes at L2 level along with the ones at L3 level:

{{W-L1}(y)L2-il/ai}(y)L3-*ākum/āṉa/āka/*āṉatu/ēyē*}*

In the tagger part of this tool, the words like *aṭippaṭaiyāka, aṭippaṭaiyākum,aṭippaṭaiyāṉa*can be accounted for using the already available rules without the L2 suffixes. But the rest of the words as presented here with L2 suffixes can be accounted for by adding them to the rule-base. What is important to note in this context is that these words need to be verified manually by a human to make sure that they are valid occurrences before adding them to the rule base, either by exploiting any machine learning algorithm or simply by adding the rest of the forms as part of the rule-base. If the tagger needs to be made robust to reduce the burden of the custom database, one needs to train the tagger in such a way that additional rules for the inflections of the words like *aṭippaṭai* can be derivable from the above set of words. What is significant to note in this context is that number of possible combinations of every word in Tamil is thus infinite and can not be accounted for by any plausible morphological tagger unless one can produce possible combinations through a custom database as defined here.

The other complex context to parse words in Tamil is to apply morphological rules in every step of parsing until one gets to the root word. For example, consider the following set of words:

mutalīṭā	mutalīṭṭāḷarkaḷukku
mutalīṭṭāḷarkaḷ	mutalīṭṭiṉāl
mutalīṭṭaic	mutalīṭṭaik
…	

In order for one to get to the root word *mutalīṭu* from the above set of words, parsing all of the suffixes in a particular order should follow the insertion of the enunciative vowel *u*, which is removed when adding suffixes to –*u* ending words. Similarly, words like *marattai, marattāl, marattukkāka* etc., with –am ending words, the sandhi rule to remove the increment *–att* to *–am* needs to be implemented. What one might envision under this circumstance is that it might be possible to make rules to parse words that are finite in nature, but in the real world there exists infinite number of combinations out of a finite set of words and suffixes. To complicate the process further, there are words with prefixes such as –*ak, -at, -iv, -av* etc., as in *akkāraṇattukkāka, attoḻilnuṭpattukkā, ivvēḷaiyil, avvēḷaikkā, av*vaṭippaṭaitturaikaḷai etc., which make the number of possibilities of inflected words even higher in number due to their two way agglutinative characteristics. If our aim is to identify the root word out of any inflected word, what we see in this context is that there is a two-way parsing method that one needs to implement and each of these paths leads to unimaginable and unaccountable number of possibilities of inflected words.

3. Infinite number of inflections with finite number of words

What poses as a problem to correctly identify Tamil words as whether they are correct forms or not is not the total number of morphological rules in the language, but their infinite number of possibilities of word formation techniques. Possible occurrences of words in their lexical forms in the language may be finite in the sense that building a relatively big dictionary can account for the already available words as well as the newly created words, but accounting for all of the possible combinations of suffixes with each of these words would be impossible to account for. Accounting for words in English, Hindi and other word level languages can be simple provided a robust dictionary as well as a set of comprehensive rule base is built. But, what happens in the agglutinative languages, like Tamil, is that the agglutinative and iterative property makes the combinations of word forms infinite in nature and they are almost unaccountable by any definite set of computing rules.

4. Crowd-sourcing method

The application that is under discussion is accessible at

http://www.thetamillanguage.com/spellcheck

as well as with an ability to read any website using the following URL in a GET method:

http://www.thetamillanguage.com/spellcheck/url_spellcheck.php?url=http://

In the former method one would need to copy/paste Tamil text or any bilingual text with Tamil and perform the spellcheck. In the latter method, on the other hand, one can specify any website using its URL and spellcheck the content in the GET string. In both methods, this tool identifies only Tamil words and attempt to recognize them by making use of the morphological parser, electronic dictionary and the custom database. When attempting to parse words, this application tries to get to the root word and match it with that of the lexical words as stored in the electronic dictionary. If this step fails, it tries to see if the inflected word is already available in the custom database. These two steps are, thus, used to identify all of the morphologically acceptable words in Tamil. At the end of the process, all the misspelled words along with the unrecognized words are presented to the user to let them store the acceptable words but unrecognized by this application into the custom database. At this step, the users are required to go through the words manually and identify only the correct words to store them in the database. This step is unavoidable for the reason that only this sort of methods can be used to deal with the infinite nature of possible combinations of Tamil words. All of the words as stored in this custom database are counter checked periodically for their validity in terms of conforming to the morphological as well as stylistic rules. Although this step is more tedious and time consuming compared to the other steps of spell checking and storing, it becomes the most important step for the reason that the custom database is crucial to account for all of the infinite number of word formation techniques. Storing any incorrect words in this custom database would lead to defective nature of this application. As of now, this custom database is thoroughly monitored to make sure that it does not contain any of the following type of words: a) Spoken Tamil forms, b) English words written in Tamil script and c) misspelled Tamil words. What it entails is that this tool can be used to identify not only all of the misspelled words but also the spoken Tamil forms along with the English words written in Tamil script.

4.1. Three types of custom database tables

Although this tool allows users at large to store words in the database, three types of restrictions are implemented to avoid any unwanted circumstances. When the admins and other known responsible users are using this tool, the words they store are kept in a database separate from the others, which are constantly updated by users at large. The difference between these two types of databases is that the latter requires additional scrutinization by the admins where as the former not. In a way, the former type of database may be called as the monitored and secure database while the later an unmonitored one but scrutinized with additional efforts. Besides, a third type of database is also made use of to store words from automatic harvesting by what is called the process of web crawler. In this method all of the unrecognized and unaccountable words are allowed to be updated in this database constantly without any human involvement as it grows. The advantage of this method is to build the database with innumerable number of words automatically without any human intervention, but the disadvantage of this method, however, is that words in this custom database requires excessive amount of time to scrutinize them manually. Thus, words from these three database tables namely inflected_words_closed, inflected_words_open and harvested_words collectively contribute to the strength of this spelling checker in one way or the other. This application, as of now, can check ten words per second. However, when more rules are incorporated into the rule base and the tagger is capable of recognizing more number of words, the efficiency of this application can be improved further. A cumulative list of the words from all of these three database tables can be viewed at:

http://www.thetamillanguage.com/spellcheck/viewcustomdatabase.php.

5. Sandhi checker

This application is also enabled with checking the Tamil sandhi forms as conforming to traditional Tamil grammars. As of now, the requirements such as dative form with the suffix *–kku*, accusative form with the suffix –ai, infinitive form with the suffixes –kka, -ka, -a, adverbial marker –āka, adverbial participle markers –ttu, ṭṭu, ṟṟu, i, u etc., are adequately taken care of to make sure the stop sound in the following word is duplicated. Unlike the spelling checker, this part of the routine is made to check words in a sequence. It also checks to make sure that the adjectival participles and other case forms do not duplicate the following stop sounds. In order to identify these grammatical categories within the text, this application utilizes the morphological tagger to parse suffixes and finally matching the words in the dictionary for conformity. To cite one example, in order for this tool to identify the words like *marattai* as an accusative form, it uses the tagger to separate the suffix –ai, perform the *–am* ending sandhi rule and finally checks to make sure that the word *maram* is in the dictionary. When the word is not in the dictionary or if the sandhi rule fails, this tool fails to identify the accusative form and subsequently fails to make the sandhi check. Thus, the words belonging to one of the above categories need to conform to the spelling checker in order for the Sandhi checker to work properly.

6. Technical details of this application

This application is written in the programming language PHP with JQuery and Javascript front-end interfaces. An instance of a MySQL database is used to store and fetch records from tables. A number of open-source PHP libraries are used to recognize Tamil unicode characters as well as parse them based on their binary properties. These library modules are used extensively to parse Tamil words in a number of different ways of string manipulation techniques.

7. Machine learning algorithms

A comprehensive machine learning algorithm is applied on the unrecognized words as listed in the custom database, so the rule base within the tagger can be strengthened further. A data driven spellchecker is one of the options that can be explored to accomplish the tasks involving all of the novel constructions in Tamil words. "In the solthiruthi project we made advances to have a framework for a data-driven spellchecker, with algorithms to recognize conjoined letters and few stemmer functions (of many possibilities allowed in Tamil language)." (Syed et al., this volume). Also see Annamalai (2017) for a number of other approaches such as neural net configuration, data driven approach etc., to recognize a massive set of unrecognizable Tamil words by computer

8. Conclusion

Any successful Spelling Checker application would heavily rely on how one can account for all of the infinite number of inflectional forms of words in Tamil. Unlike the word level languages, like English, Hindi etc., Tamil is heavily agglutinative in nature and it makes possibilities of the combination of suffixes and prefixes in words exponential in nature. In order to manage this exponential nature of Tamil word forms, a crowd-sourcing method is implemented and demonstrated in this paper. The more the use of this crowd-sourcing application by users and subsequently building the custom database with more number of plausible words, less cumbersome it would be to recognize the errors in Tamil texts of any volume. It is important to note in this context that a similar effort of crowd-sourcing was successfully implemented to deal with the inflected words from Sangam Tamil text for the purposes of glossing application that attempts to gloss words from the poems using the information as provided in the Lexicon. (Cf. http://www.thetamillanguage.com/sangam/glossing.php).

9. References

Annamalai, M. (2017) "Classifying Tamil words – part-2," blog post at https://ezhillang.blog/2017/12/20/classifying-tamil-words-part-2/

Renganathan, Vasu (2016). Computational Approaches to Tamil Linguistics. Cre-A. Chennai, India.

Renganathan, Vasu (2001). Development of Morphological Tagger for Tamil, In the Proceedings of the International Conference on Tamil Internet 2001, Kuala Lumpur, Malaysia.

Syed Abuthahir, T. Arulalan, Sathia Narayanan, Surendhar Ravichandran, A. Arunram, T. Shrinivasan and Muthiah Annamalai, "Growth and Evolution of Open-Tamil." This volume.