Testing and Performance Evaluation of Machine Transliteration System for Tamil Language

Kommaluri Vijayanand[1, 2]*, Inampudi Ramesh Babu[1, 3], Poonguzhali Sandiran[1, 2]
(1) Department of Computer Science and Engineering,
(2) Pondicherry University, Puducherry - 605 014, India.
(3) Acharya Nagarjuna University, Nagarjuna Nagar - 522 510, India.
kvixs@yahoo.co.in, rinampudi@yahoo.com, poon_8724@yahoo.com

Abstract

Machine Translation (MT) is a science fiction that was converted into reality with the enormous contributions from the MT research community. We cannot expect any text without Named Entities (NE). Such NEs are crucial in deciding the quality of MT. NEs are to be recognized from the text and transliterated accordingly into the target language in order to ensure the quality of MT. In the present paper we present various technical issues encountered during handling the shared task of NE transliteration for Tamil.

1 Introduction

Out of several underlying issues relating to Machine Translation (MT) against the dependence on the human editors, Named Entity Recognition (NER) play a pivotal role. When a MT system is developed and executed, majority of the initial test cases are bound to fail, when the system attempt to translate the names, acronyms etc. Special attention is required to handle such cases where in NER and transliteration task play a pivot role (Vijayanand and Subramanian, 2006).

We had participated in the shared task towards the languages English to Tamil, English to Hindi and English to Kannada after receiving the reference corpora which consists of 1000 names for each language pair (Li et al., 2009b). Though we committed responsibility for the three language pairs viz., English to Tamil, English to Kannada and English to Hindi, we mainly concentrated on the English to Tamil language pair. The Tamil Machine Transliteration System shall be available for demonstration during the workshop.

The present transliteration system is implemented using JDK 1.6.0 for transliterating the Named Entities in Tamil language from the source names in English. The character combination in English such as *A, Aa, I, ee, u, oo, ai, o, ou,* forms vowels in Tamil. Similarly the characters *k, ng, ch, t,* etc., form consonants and the characters *ka, nga, cha* etc., form compound characters. One single character in English produce different pronunciations and for each pronunciation, there exists a separate character. For example in the words *Madura* and *Ramya* the sound of *a* is different when *a* is suffixed with *r* and *y*. Similarly, the character *n* has different pronunciations depending upon the suffix. For example in the words *Sanchit*, *Pannu*, *Nandini*, *Jahangir* the character *n* has different pronunciations depending upon the suffix. We need to identify and consider these criteria when we transliterate the words from the languages English to Tamil. Thus the present system takes into account all such cases and generate the possible transliterations for the data given by the shared task (Li et al., 2009a).

The paper is organized in such a way that, we enumerate various rules that are formulated and deployed in favor of segmentation are explained with suitable examples in section 2. The technical details regarding the system design is presented in the section 3. The results generated by the system and the evaluation of the transliterations that are carried out using different kinds of data are explained in the section 4, followed by the section 5 that concludes the papers with the overall remarks and future work.

2 Segmentation Rules

Every word is a combination of characters and transforms its sound based on the characters that surrounds it as described in the previous section. During transliteration it is quite important to identify the break points with in the word to pronounce

*Research Scholar at Acharya Nagarjuna University, India and Visiting Scholar of Université Joseph Fourier (Grenoble 1), Grenoble, France.

Proceedings of the 2009 Named Entities Workshop, ACL-IJCNLP 2009, pages 48–51,
Suntec, Singapore, 7 August 2009. ©2009 ACL and AFNLP

the given word correctly. Towards enforcing such constraint we had devised and employed various rules towards segmentation based on the phonetic conversions. They are enumerated as follows :

1. If the second index to the current index of the word is *a*, *e*, *I* or *u*, then it is considered to be one individual segment.
2. If the second index to the current index of the word is *h* and the third index to the current index of the word is *a*, *e*, *I*, *o* or *u*, then it is considered as one segment.
3. If the second and third index to the current index of the word is *a*, *e*, *I*, *o* or *u* and if it is same character i.e., *aa*,*ee*, *oo*, then it is considered as one segment.
4. If the second index to the current index of the word is *a*, *o* and the third index to the current index of the word is *e* or *u*, then it is considered as one segment.
5. If the second and third index to the current index of the word does not satisfy any of the above four conditions then the current index of the word is considered to be as one segment.

Based on these rules the partition algorithm was sketched and implemented in favor of partitioning the word. The partitioning algorithm is applied only for the named entities and explained with the following example.

Let us consider a word *Chandrachur*, the present system navigate through five steps for segmenting this word as listed below :

1. The word is fragmented as Cha | ndrachur

   Initially the system parse from the initial character *c* and checks the second index. It recognizes that the second index is *h*. Then it reads the third index according to the rule number 2. It then recognizes that the third index is *a*. So the system partitions up to that third index and consider it as one segment.

2. Further segmentation : Cha | n | drachur

   Now the system starts from the fourth index and consider that index as the current index. It continues checking the fifth index. As it does not satisfy any of the rules, it partitions the fourth index from the source word and consider it as one segment.

3. Cha | n | d | rachur

   In the third step checking starts from the fifth index and now it is considered to be the current index. Then it checks the sixth index. Since, it does not satisfy any of the rules, the system partitions the fifth index from the source name and it is considered as one segment.

4. Cha | n | d | ra | chur

   Now the system starts from the sixth index and consider this to be the current index. It checks the seventh index, after recognizing the presence of *a*, then it checks whether the eighth index is *a*, *e* or *u*, as per the rule 3 and rule 4. As it does not satisfy with those rules, the system partitions from sixth index to seventh index as one segment.

5. Cha | n | d | ra | chu | r

   Finally checking starts from the eighth index which is treated as the current index and the system checks the ninth index. The ninth index consists of *h*. Thus, checks the tenth index for the presence of *a*, *e*, *I*, *o* or *u* according to the rule 2 and satisfies with that rule. Thus, the system partitions from eighth index to tenth index as one segment and the eleventh index become one segment.

Similarly, for the word *Manikkam* the system applies the partitioning algorithm and segment the word as shown below :

1. Ma | nikkam
2. Ma | ni | kkam
3. Ma | ni | k | kam
4. Ma | ni | k | ka | m

3 The System Design

The system was designed in such a way that it produces four to six transliterations for a given word in English. We stored all the possible combinations of characters in English and its corresponding Tamil characters in a database and created an interface to read the test file. The system is facilitated to browse the test file using the file handling technique which was designed applying the logical concepts. Consonants in English when combined with vowels in English to form compound words in Tamil. Compound words have many forms for a single combination.

The present system extract the source names and store them in an array list. These source names are retrieved from an array list sequentially and stored in a string variable for further processing. The value of the string is parsed character wise and check for the existence of a vowel or *h*, in the next two positions to its index i.e., for each character the next two characters are checked, if there exists vowels or *h*, then these characters are extracted up to that index and stored in another string variable. Other wise only that variable is stored and compared with the database that contain Tamil characters, for each combination of characters that are present in English. Thereafter each index in an array list of each transliteration will be combined with each index in another array list of transliterated letter combination, stored in another variable. This process will continue until the system encounter the end of each array list. After getting all the combinations, these combinations are stored in an array list and it is written to the file.

It is to be noted that only one source name is assigned to the string variable at a time. After getting the target name of that source name, the next source name is retrieved from an array list. After retrieving the source name it is passed to the next module for segmentation. The segments formed are stored in an array list. Then these target characters for each segment is retrieved from the database and stored in a separate list. There after the values in an array list are merged appropriately and stored in an array list.

4 Results and Evaluation

This section describes briefly about the results and evaluation conducted and present the results. We had employed various techniques and algorithms as explained in previous sections, to select the appropriate transliterated word that matches the source name from the n-best candidate list using six metrics.

4.1 Results

The result file consists of source name with its ID and the ranked list of target names. The target names are generated along with the source names, after being processed by the system. The source names are the names given in the test file. The target names are the names that are generated by the system. The target names are ranked according to their ID's. The target names are Unicode characters in Tamil. After applying various techniques we produce the result file. It is worth stating that the result file is generated in the XML format using UTF-8 encoding schema.

The present system transliterates for 1000 source names and generates up to six best candidate lists (Target names). We conducted testing for the given data towards transliterations. The first transliteration present in the four best candidate lists are considered to be the correct hit. The evaluation is carried out using Python. These six metrics are implemented in python. The metrics are as follows :

1. Word Accuracy in Top-1 (ACC)
2. Mean F-Score
3. Mean Reciprocal Rank (MRR)
4. MAP ref
5. MAP_10
6. MAP sys

MAP refers to the Mean Average Precision. Python is preferred because it is an excellent programming language, easy to understand, dynamic and truly object oriented.

4.2 Evaluation

The Result file and the Test file in XML format and the python script developed with six metrics reads the above mentioned files. Execution of the script requires Python interpreter. The Result file is the one generated by the Transliteration System and the test file is created manually with a single transliteration for each source name and testing is conducted. As part of the shared task (Kumaran and Kellner, 2007) evaluation was done by running the system and thus 6 metrics are displayed as output, with each metric given the value 0 or 1. These metrics declare the performance of the system. The max-candidates argument in the script is assigned 10 (max-candidates=10). It is also changed according to target names provided. The output of the evaluation of our Transliteration System are as follows :

1. Word Accuracy in Top-1 (ACC) : The ACC of our system is 0.403974,
2. Mean F-Score : The Mean F-Score of our system is 0.865840.
3. MRR = The Mean Reciprocal rank of our system is 0.449227.
4. MAP ref : The MAP ref of the system is 0.390545.

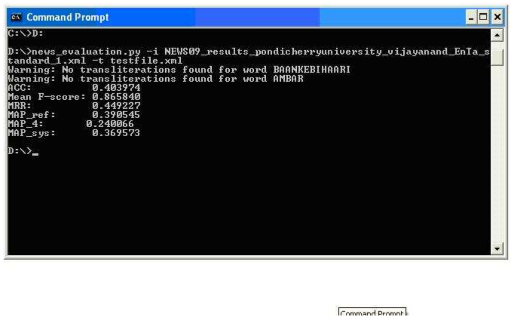
5. MAP_10 : The MAP_10 value of the system is 0.240066.
6. MAP sys : The MAP sys of the system is 0.369840.

The output that was generated by our system is presented in appendix.

5 Conclusions

After working with the experiment that was carried out for evaluating the metrics, we conclude that the accuracy in top-1 score of our system is 0.061. The reason could be that the accurate transliteration is not generated in the top scored transliteration. We could improve the performance of the present system by involving all the possible transliterations. With the initial test results are very low when compared to Urdu to Hindi transliteration system (M. G. et al., 2008), yields 97.12% and Hindi to Urdu delivers 97.88% of accuracy and NER system favor of the Bengali language which had demonstrated the evaluation results with a precision of 80.12% (Ekbal and Bandyopadhyay, 2008).

After participating in the shared task we had tested the transliteration system thoroughly by applying various techniques as explained in the present paper. So far we had carried out the transliteration for six named entity candidates. In future we would like to extend the task for transliteration candidates unto twenty. Thus the named entity transliteration task that is being carried out would be a solution for the long standing research problem in handling the named entities that is quite common in speech and text machine translation.



```
C:\>D:

D:\>news_evaluation.py -i NEWS09_results_pondicherryuniversity_vijayanand_EnTa_s
tandard_1.xml -t testfile.xml
Warning: No transliterations found for word BAANKEBIHAARI
Warning: No transliterations found for word AMBAR
ACC:          0.403974
Mean F-score: 0.865840
MRR:          0.449227
MAP_ref:      0.390545
MAP_4:       0.240066
MAP_sys:      0.369573

D:\>_
```

FIG. 1 – Screenshot of Evaluation Result.

Acknowledgments

I am thankful to the anonymous referees for their valuable advices towards improving this paper. I am thankful to my students Kanickairaj Caroline, Dhivya Moorthy and Kothandapani Selvi for rendering their service and cooperation in fulfilling this task. I extend my gratitude to all the elders for their support and encouragement.

References

Asif Ekbal and Sivaji Bandyopadhyay. 2008. Named entity recognition using support vector machine : A language independent approach. *International Journal of Computer Systems Science and Engineering*, 4(2) :155–170.

A. Kumaran and Tobias Kellner. 2007. A generic framework for machine transliteration. In *30th Annual ACM SIGIR Conference*, Amsterdam.

Hauzhou Li, A Kumaran, Vladimir Pervouchine, and Min Zhang. 2009a. Report on news 2009 machine transliteration shared task. In *Proceedings of the ACL-IJCNLP 2009 Named Entities Workshop (NEWS 2009)*, Singapore.

Hauzhou Li, A Kumaran, Min Zhang, and Vladimir Pervouchine. 2009b. White paper of news 2009 machine transliteration shared task. In *Proceedings of the ACL-IJCNLP 2009 Named Entities Workshop (NEWS 2009)*, Singapore.

Abbas Malik M. G., Christian Boitet, and Pushpak Bhattacharyya. 2008. Hindi urdu machine transliteration using finite state transducers. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, Manchester.

Kommaluri Vijayanand and Ramalingam Subramanian. 2006. Anuvadini : An automatic example-based machine translation system for bengali into assamese and oriya. In *Proceedings of the First National Symposium on Modeling and Shallow Parsing of Indian Languages (MSPIL-06)*, IIT Bombay, India.