Named Entity Transliteration for Cross-Language Information Retrieval using Compressed Word Format Mapping algorithm

Srinivasan C Janarthanam
School of Informatics
University of Edinburgh
Edinburgh, UK
s.janarthanam@ed.ac.uk

Sethuramalingam S
Search and Information Extraction Lab
International Institute of Information Technology Hyderabad, AP, India
sethu@research.iiit.ac.in

Udhyakumar Nallasamy
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA
udhay@cmu.edu

ABSTRACT

Transliteration of named entities in user queries is a vital step in any Cross-Language Information Retrieval (CLIR) system. Several methods for transliteration have been proposed till date based on the nature of the languages considered. In this paper, we discuss about our transliteration algorithm for mapping English named entities to their proper Tamil equivalents. Our algorithm employs a grapheme-based model, in which transliteration equivalents are identified by mapping the source language names to their equivalents in a target language database, instead of generating them. The basic principle is to compress the source word into its minimal form and align it across an indexed list of target language words to arrive at the top n-equivalents based on the edit distance. We compare the performance of our approach with a statistical generation approach using Microsoft Research India (MSRI) transliteration corpus. Our approach has proved very effective in terms of accuracy and time.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.7 [**Natural Language Processing**]: Text analysis

General Terms

Algorithms, Performance, Experimentation

Keywords

Cross-Language Information Retrieval, Transliteration, Compressed Word Format, Modified Levenshtein distance

1. INTRODUCTION

Transliteration refers to expressing a word in one language using the orthography of another language. Its usage is particularly common in scientific and news articles when they refer to named entities or events of another language, is different from the one used to write these articles. In this paper we present a novel approach to transliterate named entities in English to Tamil.

The amount of Indian language content on the web has seen tremendous growth in the recent past owing to the origin of several regional news websites and information repositories like Wikipedia pages in major Indian languages like Hindi, Telugu, Tamil, Bengali etc. This growth can be attributed to the increased Internet access in suburban areas and subsequent development of locally relevant content serving these communities who are more comfortable in using their native language to access such information. These multi-lingual websites can gain greater visibility in the web across the world, if they can be indexed and retrieved using normal English queries. Hence, in addition to the relevant English documents, relevant documents in languages like Tamil, Hindi, etc can also be presented to the user. In some cases, there may be more relevant documents in regional languages than in English. In order to do this, the system should translate the query in English into other languages and search the appropriate language archives as well.

Translating queries from English to other languages presents a lot of interesting problems. Bilingual lexicons can be used to translate the queries from one language to another. However, one of the main problems with this approach is the coverage of such lexicons. As pointed by [11], out-of-lexicon words seem to constitute 83% of the highly frequent query terms. Analysis conducted by [12] revealed that around 50% of the out-of-vocabulary words were named entities. Results from [5] show that the average precision scores of a CLIR system get reduced by 50% when the named entities were not properly transliterated. Therefore transliteration of named entities to the target language presents an interesting challenge.

In this paper, we present an approach to transliterate named entities from English to Tamil. Apart from difference in orthography, there are several other issues with this language pair. For instance, (i) Characters in both languages align not only in one-to-one configurations but also one-to-many and many-to-one etc. configurations and (ii) Vowels and vowel digraphs in English must be aligned to appropriate short and long vowels in Tamil. In this paper we present a novel approach to transliteration in the cross language information retrieval framework. We suggest mapping instead of generating a name in the target language. We propose an algorithm to compress the given named entity in to its minimal form called Compressed Word Format (CWF) and then map it to the entries in a list of named entities in the target language. We propose two hypotheses - (i) Compressed Word Format mapping model is more accurate than the statistical generation models in English-to-Tamil transliteration, and (ii) In case of mapping based approaches, Compressed Word Format is more accurate and precise than the actual word forms. In this work, we have verified these hypotheses. From the evaluation results, we found that by compressing the words to their minimal forms, we were able to make the mapping process faster and more accurate. We also found that the mapping method was more accurate than the statistical generation approach.

In section 2, we briefly discuss the related work in named entity transliteration. In section 3, we present the algorithms related to our approach. We discuss our development process in section 4 and our evaluation and results in section 5.

2. PREVIOUS WORK

Transliteration of words from a source language (S_L) to a target language (T_L) is mostly considered to be a problem of generation of target language equivalents using statistical approaches. The major techniques for transliteration can be broadly classified into two categories namely Grapheme-based and Phoneme-based. An n-gram based statistical model for transliteration of English to Arabic names was implemented in [5]. Chinese orthography for transliterating English words into Chinese was used in [3]. In Indian language context, a word origin based approach for splitting Indian and foreign origin words and transliterating them based on their phoneme equivalents was shown by [9]. A discriminative, CRF-HMM model for transliterating words from Hindi to English was used in [10]. Based on their transliteration system efficiency, they have shown improvements in the performance of their CLIR system for CLEF Hindi-English queries.

The concept of splitting words into their composite consonants and vowels for transliteration was found to be used by [7] earlier. They used a grapheme-based model for English to Persian Transliteration. Character alignments were based on probabilities using GIZA++. Their methodology involves splitting of words into their constituent consonants & vowels from the source and target languages and focus on combining various combinations of vowels and consonants and coming up with the most probable ones.

Our approach of transliteration based on mapping can be compared to the approach followed by [6] for Arabic to English transliteration of proper Nouns. Their main focus is on building a transliteration model for Arabic-English machine translation system. They have used a Noisy-channel model for deriving English equivalents based on the alignment probabilities of English and Arabic characters. A consonant-based mapping scheme is used to compare the generated English words with their large collection of English names. The final score for the best English transliteration is generated based on the sum of scores at the three different phases of their system. Our scenario and approach differs from theirs in two ways, (1) their transliteration system is basically for machine translation where one possible transliteration is valid. Our system is built for the CLIR scenario where one fixed transliteration may not be sufficient, (2) we have used a simple, heuristic based mapping of English and Tamil characters when compared to their HMM based Arabic-English characters alignment model.

3. OUR APPROACH

Named entities in English are mapped to their proper Tamil equivalents by comparing their minimal consonant skeletal forms or Compressed Word Format (CWF), produced based on a set of linguistic rules. Prior to it, pre-processing on the list of Tamil names has to be performed. Once we derive the compressed word formats for the English query word and for the list of compressed Tamil words equivalents, we search and match the right equivalent in the index based on our modified Levenshtein algorithm.

3.1 Pre-processing

Before the actual transliteration process, our approach involves certain pre-processing operations. Firstly, the named entities in the target language, Tamil, are collected and listed. This can ideally be done by running named entity recognizers on the archives. These names are then romanized so that they can be easily compared to the English queries. There are several romanization schemes. In most of these schemes, Tamil characters (vowels and consonants) are mapped to one or more Roman characters, in order to increase readability. But, we used a custom romanization scheme (given in the Appendix) that maps every Tamil character to Roman characters in a one-to-one mapping fashion. One-to-one mapping strategy ensures that the Romanized forms are compact and avoids any state-based processing. Then using the CWF algorithm (given below), we derive the compressed word format of every word in the list. Finally, an index of tuples is created, with each tuple containing the compressed string and the actual string.

3.2 CWF algorithm

In the CWF algorithm, we compress both English and Tamil named entities from their actual forms (AF). Compressed Word Format of words is created using an ordered set of rewrite and remove rules. Rewrite rules replace characters and clusters of characters with other characters or clusters. Remove rules simply remove the characters or clusters. This algorithm is used for both English and Tamil names, but their rule sets are different. A set of English rules are given with examples.

Step 1: Rewrite consonant digraphs (group of consonants with no intervening vowels) representing one phoneme as single character consonants. This replacement is done to reduce edit distance between CWF forms.

e.g. chidhambaram → cidambaram

Step 2: Rewrite double consonants like 'kk', 'rr', etc as respective single consonant.

e.g. musharraf → musaraf

Step 3: Rewrite clusters based on target language heuristics. In our current work, we considered specific heuristics for Tamil.

Step 4: Remove the vowels. Our final CWF forms will only have the minimal consonant skeleton.

e.g. cidambram → cdmbrm

Similar rules are developed for reducing English names to CWF format.

3.3 Modified Levenshtein algorithm

Levenshtein's Edit Distance algorithm is popularly used to calculate the edit distance between any two strings in the same language. In our current work, we use it to calculate the distance between the source language string and the target language string in CWF format. Since these strings use different character sets, we modify the algorithm to handle acoustically equivalent characters in both languages. For instance, 'd' in English and 'T' in Tamil are considered equivalent. These pairs of characters were identified during the development cycle. We refer to this algorithm as Modified Levenshtein (MLev) algorithm in this paper.

3.4 CWF mapping algorithm

CWF Mapping algorithm is our transliteration algorithm that takes as input a source language named entity string (e.g. Chidambaram) and produces a ranked list of transliterated names in the target language (Tamil). The steps involved are given below.

Step 1: Derive the compressed word form (CWF_E) of the input string. This is obtained using the CWF algorithm (for English).

e.g. Chidambaram → cdmbrm

Here, we see how the "Ch" is replaced with "c" and that all the vowels have been removed.

Step 2: Compute the edit distance of the CWF_E with all the CWF_T Tamil entries in the index. Collect all the matches and their edit distances.

e.g. (cTmprm, ciTamparam):0, etc..

Edit distance is computed using a modified Levenshtein algorithm (MLev) We call the edit distance calculated using CWF forms and MLev algorithm as CWF+MLev.

Step 3: Rank the candidate matches based on the CWF+MLev edit distance. When there are more than one candidate at the same edit distance, finer ranking can be made using edit distance between the actual forms of source and target strings using our Modified Levenshtein algorithm (AF+MLev).

4. DEVELOPMENT PHASE

Development phase refers to the two one-time processes of designing the algorithms. One is to collect the heuristic remove and rewrite rules (RR rules) for the CWF algorithm and another is to identify the acoustically equivalent (AE) pairs for the modified Levenshtein (MLev) algorithm.

An initial set of RR rules and AE pairs were intuitively hand-coded. We then added to this initial set based on our development data during the development phase. For this purpose we created the development datasets. One of the authors manually transliterated around 7200 Indian names from Tamil to English. These names were a part of state-wise, electoral data containing the list of voters' names, provided online by the Election Commission of India[1]. We also collected around 700 foreign names with their English equivalents from Tamil Wikipedia[2] pages. At the end, we had around 8000 matching pairs in our development datasets. Each matching pair had the English named entity string and its matching Tamil string.

We ran our CWF and MLev algorithm on the two development datasets to calculate the average edit distance between matching pairs. Ideally, our edit distance algorithm must calculate the distance between matching pairs to be zero (because they perfectly match according to human annotation standards). However with the initial set of heuristics, the results were not even close, because the strings were using different character sets. We manually inspected those pairs with high edit distances between them and identified RR rules and AE pairs that need to be added to avoid the extra distance between them. For instance, replacing 'ch' with 'c' (in English strings) reduces the distance as it is transliterated to 'c' in Tamil strings. Similarly, adding 'd' and 'T' as equivalent pairs helps in reducing the distance, as they are acoustically similar although orthographically different.

We have compared our CWF+MLev scores with the standard Levenshtein edit distance scores between the actual forms of the strings (AF+Lev). The results of the experiment are shown in Table 1.

Table 1. Average Edit distance CWF+MLev vs. AF+LEV

Dataset	CWF+MLev distance	AF+LEV distance
Indian names	0.1845	4.6186
Foreign names	0.7922	6.9532

At the end of the development phase, we had an algorithm that gives the lowest edit distance for the matching pairs. The above table shows the average CWF+MLev edit distance between matching pairs, which is closer to zero than the standard Levenshtein edit distance (AF+Lev) between actual forms. The implication of this result is that, at distance 1 or less, our algorithm is more likely to search the index and produce a matching pair by using CWF+MLev distance than AF+Lev distance. For AF+Lev distance to produce a matching pair, the distance threshold has to be set to 7 (as its average distance is 6.9 for foreign names). However, at the distance threshold 7, the algorithm will pick up more spurious false matches along with the real matching string. Hence the precision will degrade although we achieve a high recall. However, with our approach of combining MLev algorithm with CWF word forms, we can produce high recall at shorter distance thresholds and therefore retain high precision.

5. EVALUATION

In order to prove our hypotheses, we ran two experiments. In the first experiment, we compared the performance of CWF Mapping algorithm against the IIIT's CRF+HMM Generation Model [10]. In the second experiment, we compared the performance of our algorithm against the standard Levenshtein algorithm. For both these experiments, we used a parallel dataset containing names in five Indian languages provided by Microsoft Research India. It consists of 3300 names (3000 in the training set and 300 in the testing set) parallel in English, Tamil and other Indian languages. We paired the English names with their corresponding Tamil names in both training and testing lists. We created an index of Tamil names (as described in Section 3.1) using both the training and testing lists.

[1]ECI list - http://www.eci.gov.in/DevForum/fullname.asp

[2]Tamil Wikipedia - http://ta.wikipedia.org

5.1 First experiment

We ran our CWF Mapping algorithm on the 300 names from the test set. The algorithm found potential matches for all the 300 names from the index of 3300 names and output a ranked list for each input English name. The ranking is done based on the edit distance between the query and the candidate, in such a way that the candidate with the lowest edit distance is ranked high, and so on. However in the ranked list, sometimes, there is more than one candidate per rank because they all have the same edit distance. This makes it difficult for us to compare our results to other algorithms that produce a list with one candidate per rank and usually report their accuracy at top 1, 2, 5 and 10 candidates.

Table 2 shows the potential candidates for the English input query 'Bakul' at top 1, 2, 5 and 10 ranks. At top 1 rank (which is just rank 1), there are 2 potential candidates. At top 2 ranks (which consists of candidates from Rank 1 and 2, we have 4 candidates) and so on. This illustrates our case of not having one candidate per rank.

Table 2. Ranked list of candidates for the English input query name ' bakul', which has 'pakUl' as the correct romanized Tamil equivalent

Top N Rank	Number of candidates	Candidates
Top 1 rank	2	pAkul; pakUl;
Top 2 ranks	4	pAkul; pakUl; pOkil; pOklE;
Top 5 rank	11	pAkul; pakUl; pOkil; pOklE; kul; kulE; pAkE; pAlE; pAkki; patkal; uppal;
Top 10 ranks	24	pAkul; pakUl; pOkil; pOklE; kul; kulE; pAkE; pAlE; pAkki; patkal; uppal; kAl; AkalE; akOlA; kOlE; pIlA; kellA; pOkkE; pellO; pikki; pinkal; qpilA; piGkAlA; pinkalE;

Hence, in order to provide a fair comparison, we tried to figure out the average number of candidates that are identified for top n ranks over the test set. This is shown in Table 3.

Table 3. Average number of candidates

Top n ranks	1	2	5	10
Average number of candidates	1.10	2.38	7.87	20.13

From the above table, we see that there are on average 1, 3, 8 and 20 (rounded) candidates at top 1, 2, 5, and 10 ranks. Therefore, we report our accuracy results at top 1, 2, 5 and 10 ranks and compare them with top 1, 3, 8 and 20 candidates from the generation algorithm. Table 4, reports the accuracy of our algorithm at top n ranks.

Table 4. Ranked list of candidates for the entire 300 English query names in the MSRI test data

Top n rank	Number of queries	Accuracy
1	271	0.9064
2	280	0.9365
5	287	0.9599
10	295	0.9866

Out of 300 query names, the exact match was found to be in Top 1 rank (the first candidate) for 271 query names. Similarly, the exact match was found for 280 query names at top 2 ranks (in the first 3 candidates), for 287 query names at top 5 ranks (in the first 8 candidates), and for 295 query names at top 10 ranks (in the first 20 candidates).

In the following table (Table 5) we compare our results with the CRF-HMM generation model accuracy results at top 1, 3, 8 and 20 candidates.

Table 5. Transliteration accuracy of our CWF mapping algorithm compared with IIIT's CRF-HMM model

System	Top 1	Top 3	Top 8	Top 20
CRF-HMM	0.4181	0.6154	0.7659	0.8261
CWF Mapping	0.9064	0.9365	0.9599	0.9866

The accuracy scores in Table 5 clearly show that our mapping based technique clearly outperforms the IIIT's statistical generative model for English-Tamil transliteration by a large margin. Figure 1 shows the improvement in transliteration accuracy between the two systems at different ranks. This proves our first hypothesis that mapping based approach is more efficient than the statistical transliteration approach for English-Tamil transliteration.

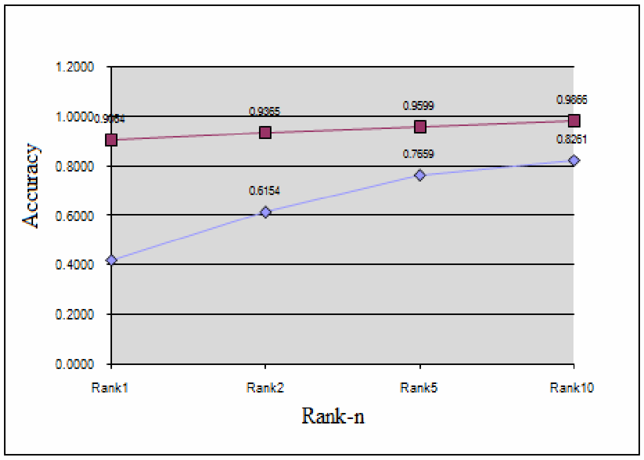

Figure 1. Transliteration performance comparison

On analysis, we found that most of the candidates generated by the IIIT's generation algorithm were not even found in our index. This has significantly reduced the accuracy of the algorithm. In order to improve the accuracy, we mapped each of the generated candidates to its closest (in terms of standard Levenshtein distance) match in the index. This step ensures that each candidate in the list is now in the index as well. This additional step increased the accuracy of the IIIT algorithm, proving again that mapping approach is more beneficial. We then compared the results of CRF-HMM model (now enhanced by mapping) to our results. This is given in Table 6.

Table 6. Accuracy of CWF mapping algorithm compared with modified IIIT's CRF-HMM model

Top n candidates / Model	**1 (Top 1 rank)**	**3 (Top 2 ranks)**	**8 (Top 5 ranks)**	**20 (Top 10 ranks)**
Modified CRF-HMM	0.7860	0.9064	0.9398	0.9398
CWF Mapping	*0.9064*	*0.9365*	*0.9599*	*0.9866*

The above results show that our mapping algorithm still performs significantly better than IIIT's generate and map model. In addition to accuracy, the mapping model provides an advantage of lesser execution time than the IIIT model. In order to map 20 generated candidates to names in the index, we had to run Levenshtein's algorithm 20 * 3300 times (named entity index size is 3300 names). And this was run for every query in the 300 name test dataset. In contrast, in our mapping model, we only have to run our modified Levenshtein (MLev) algorithm 3300 times to identify the top 20 candidates. In addition, for each of these 20 candidates, we had to run Levenshtein algorithm between the actual forms of the pairs for the purpose of finer ranking, so 3300 + 20 iterations in total. Hence for m entries in the index and n candidates, the IIIT's generate and map model executes Levenshtein algorithm (n * m) times, while our mapping model only executes it n + m times. Also, since after compression the strings are only half as long, the modified Levenshtein algorithm takes only half the time as the standard Levenshtein algorithm. Reduction in execution time becomes a significant advantage as index gets larger.

5.2 Second experiment

In our second experiment, we compared two versions of our mapping algorithm. One used the CWF+MLev distance and another used AF+Lev distance for ranking the candidates. This experiment was done to demonstrate the effect of CWF and MLev algorithms in our mapping model. We evaluated them on the MSRI test dataset (300 names) and the index containing 3300 names. The following table (Table 6) shows how accurate the algorithm is by using the two edit distances. In the parentheses, we mention the average number of candidates.

Table 7. Transliteration accuracy of our compressed word format, modified Levenshtein algorithm (CWF+MLev) compared with normal Levenshtein algorithm

Top n candidates	**CWF+MLev**	**Levenshtein**
Top 1 rank	0.9064 (1)	0.5953 (12)
Top 2 ranks	0.9365 (3)	0.8796 (141)
Top 5 ranks	0.9599 (8)	1 (3002)
Top 10 ranks	0.9866 (20)	1 (3004)

The comparative results in the Table 6 shows that Compressed Word Format distance based mapping helps in achieving far better results when compared to the normal Levenshtein edit distance. One should note that there are 12 competing candidates at rank 1, while using AF+Lev distance instead of CWF+MLev distance. This means that, although 59% of queries found their exact match in rank 1, there are 11 other competing candidate and we have no way to tell which of the 12 candidates is the exact match. Precision is hugely sacrificed to improve recall. Similarly, we have 141, 3002 and 3004 candidates at top 2, 5 and 10 ranks respectively. Although there is 100% recall at top 5 ranks, the number of candidates mapped is 3002. We cannot possibly use 3002 candidates to search the Tamil archives. Considering the loss of precision and high accuracy at top 1 and 2 ranks, we could conclude that the mapping algorithm using CWF+MLev distance is more effective than the one using AF+Lev distance. This validates our second hypothesis that compression improves accuracy and precision in the mapping models.

6. CONCLUSION

In this paper we have presented an efficient algorithm for transliteration of English named entities to Tamil. We presented the problems in transliteration in the language pair. We then presented the CWF algorithm where strings were compressed without losing essential information that is needed for transliteration. We have shown how to use compressed word forms and modified Levenshtein algorithms to produce accurate transliterations of named entities in the target language. Our results have proved that mapping is a better option than generating in the context of transliteration. Our results show that by combining compressed word format (CWF) with modified Levenshtein algorithm, we can bring the best of both worlds together, i.e. increase accuracy without sacrificing precision.

7. FUTURE WORK

Although CWF forms reduce the execution time of modified Levenshtein algorithms, with increasing size of the index, the execution time could get significantly longer. We will work on ways to make the search faster. We also want to work on integrating our mapping model to a CLIR system and evaluate extrinsically the effect of compression and mapping on document retrieval.

8. ACKNOWLEDGMENTS

Our sincere thanks to Mr. Prasad Pingali and Mr. Surya Ganesh of IIIT-Hyderabad for sharing their results with us. Our heart-felt thanks to Dr.A.Kumaran, Microsoft Research India for providing the Indian language transliteration parallel corpus for our evaluation and testing.

9. REFERENCES

[1] Prochasson, E., Viard-Gaudin, C., and Morin, E. 2007. Language Models for Handwritten Short Message Services. In Proceedings of the Ninth international Conference on Document Analysis and Recognition (ICDAR 2007) Vol 1 - Volume 01 (September 23 - 26, 2007). ICDAR. IEEE Computer Society, Washington, DC, 83-87.

[2] Knight, Kevin and Graehl, Jonathan. 1997. Machine transliteration. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, pp. 128-135. Morgan Kaufmann.

[3] Virga, P. and Khudanpur, S. 2003. Transliteration of proper names in cross-lingual information retrieval. In Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-Language Named Entity Recognition - Volume 15 Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 57-64. DOI= http://dx.doi.org/10.3115/1119384.1119392

[4] Al-Onaizan, Y. and Knight, K. 2002. Machine transliteration of names in Arabic text. In Proceedings of the Acl-02 Workshop on Computational Approaches To Semitic Languages (Philadelphia, Pennsylvania). Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 1-13.

[5] Nasreen Abdul Jaleel , Leah S. Larkey, Statistical transliteration for english-arabic cross language information retrieval, Proceedings of the twelfth international conference on Information and knowledge management, November 03-08, 2003, New Orleans, LA, USA

[6] Mehdi M. Kashani, Fred Popowich, & Fatiha Sadat. Automatic transliteration of proper nouns from Arabic to English. The Challenge of Arabic for NLP/MT. International conference at the British Computer Society, London, 23 October 2006; pp.76-83.

[7] Sarvnaz Karimi, Andrew Turpin, Falk Scholer. 2006. English to Persian Transliteration. SPIRE 2006: 255-266.

[8] Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender, Ed. Acm Press Frontier Series. ACM Press, New York, NY, 19-33.

[9] Harshit Surana and Anil Kumar Singh, 2008. A More Discerning and Adaptable Multilingual Transliteration Mechanism for Indian Languages. Proceedings of International Joint Conference on Natural Language Processing(IJCNLP)-2008, Hyderabad, India

[10] Surya ganesh, Sree Harsha, Prasad Pingali, Vasudeva Varma. 2008. Statistical Transliteration for Cross Language Information Retrieval using HMM aligment and CRF, Proceedings of International Joint Conference on Natural Language Processing(IJCNLP)-2008, NERSSEAL Workshop, Hyderabad, India

[11] Pu-Jen Cheng , Jei-Wen Teng , Ruei-Cheng Chen , Jenq-Haur Wang , Wen-Hsiang Lu , Lee-Feng Chien, Translating unknown queries with web corpora for cross-language information retrieval, Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, July 25-29, 2004, Sheffield, United Kingdom

[12] Davis, M. W., and Ogden, W. C. (1997). Free resources and advanced alignment for cross-language text retrieval. In: The Sixth Text Retrieval Conference (TREC-6). NIST, Gaithersbury, MD

[13] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10 (1966):707–710.

10. APPENDIX

Our Romanization scheme showing one-to-one mapping between Tamil characters and the Roman characters is given below.

அ	a	க	k	வ	v
ஆ	A	ங	G	ழ	z
இ	i	ச	c	ள	L
ஈ	I	ஞ	J	ற	R
உ	u	ட	t	ன	n
ஊ	U	ண	N	ஜ	j
எ	e	த	T	ஷ	Z
ஏ	E	ந	D	ஸ	S
ஐ	Y	ப	p	ஹ	h
ஒ	o	ம	m		
ஓ	O	ய	y		
ஔ	W	ர	r		
ஃ	q	ல	l		