

Hidden Markov Model Based Identification of Transliterated Regional Language Words in Text Documents

Achuth Sankar S. Nair¹, Vrinda V. Nair², Vinod Chandra S. S.³

¹Centre for Bioinformatics, University of Kerala, Thiruvananthapuram, Kerala, India

²Department of Electronics & Communications, College of Engineering Thrissur, Kerala, India

³Department of Computer Science, College of Engineering Thiruvananthapuram, Kerala, India
{ sankar.achuth, vinodchandras }@gmail.com, nairvrinda@rediffmail.com

Abstract

Text in roman script is today popular in the internet among non-English speaking countries, as the script is almost universally enabled in text processors. In countries like India which is a linguistic cauldron, it is very common to see English text in email messages and chat transcripts, with generous sprinkling of local language words in roman script. Dubbed as Hinglish (Hindi and English), Manglish (Malayalam and English) etc., these roman transliteration of non-English languages contribute a major noise in analyzing English text in these countries / groups. The present work reports the initial result of developing an HMM-based classifier for such linguistic noise in English text. The system reported is trained to classify English and non-English words, however it can be expanded to a distinct collection of models for various languages and provide a 'linguistic coloring' feature in text editors.

1. Introduction

Noisy text (we confine to English), is traditionally due to spelling errors, abbreviations, non-standard words, false starts, pause filling words etc. However in the context of non-English speaking countries like India, major noise originates from the use of transliterated regional language words. This phenomenon is seen widely in the internet chat rooms and email messages. In the phonetic Indian languages, typing overhead is more compared to English and hence there is a natural inclination to use roman script to produce so called Hinglish (Hindi and English), Manglish (Malayalam and English), Benglish (Bengali and English) etc. This is very true of young generation non-resident Indians who may be familiar with spoken form of their mother language, but not its script, which leads to text such as follows.

“Hi Dadiji,

How are you? How is chottu? I am having my vacation here and I will soon be sending some special Kapada and Makan for you...”

When this kind of text is to be machine processed, word models of various languages are required to sieve out the underlined words. We consider them along with non-English (mistakes, fillers etc.) noise together, but there is a script to clarify the non-English using distinct models for each regional language concerned. We report in this paper a preliminary investigation using stochastic models. The Hidden Markov Model (HMM) is the stochastic tool used. In our approach, we use a first-order Hidden Markov Model [Jelinek, 1976; Rabiner, 1989]. HMMs have a set of states $S = \{S_1, S_2, \dots, S_n\}$ which emit symbols with different probabilities. There is a transition probability between each succeeding states. In our case each state is supposed to emit a set of letters and the whole model generates a word that is constituted in a document. Figure 1 shows the overview of an HMM used. The assumption of first order HMM is not complete or final. This paper is organized as follows. Section 2 discusses some related works. Section 3 describes the architecture of HMM for identification of lingual noise arising due to local language words in roman scripts from a document. Section 4 presents results and conclusions.

2. Related Works

There are some models, which can be used to capture word dependencies and to perform inference on sequences, be they whole documents or short passages [Denoyer *et al.*, 2001]. They allow extending the classical paradigms of information retrieval by considering sequences of text elements instead of the classical bag-of-words representation. Another classification method is seen for biological data [Chen, *et al.*, 2006]. This uses a support vector machine-trained classifier, followed by a novel phrase - based clustering algorithm. This clustering step

autonomously creates cluster labels that are descriptive and understandable by humans. There is a new statistical model for the classification of structured documents and uses multimedia document classification [Denoyer *et al.*, 2003]. Its main originality is its ability to simultaneously take into account the structural and the content information present in a structured document, and also to cope with different types of content (text, image, etc). The Arabic word classification [Pechwitz and Maergner, 2003] is based on a semi-continuous 1-dimensional HMM. In this work the hand written words are considered. Another work [Steyvers *et al.*, 2004] deals with author-topic model classification using probabilistic model. Each author is represented by a probability distribution over topics, and each topic is represented as a probability distribution over words for that topic. The words in a multi-author paper are assumed to be the result of a mixture of each authors' topic mixture. The topic-word and author-topic distributions are learned from data in an unsupervised manner using a Markov chain Monte-Carlo algorithm. Our work gives another perspective, which depicts an HMM, with output probabilities of each word in a text. Based on this value one can classify the noisy word such as transliterated regional language text from a document. We describe the details of the model here which was tested on several bilingual corpora.

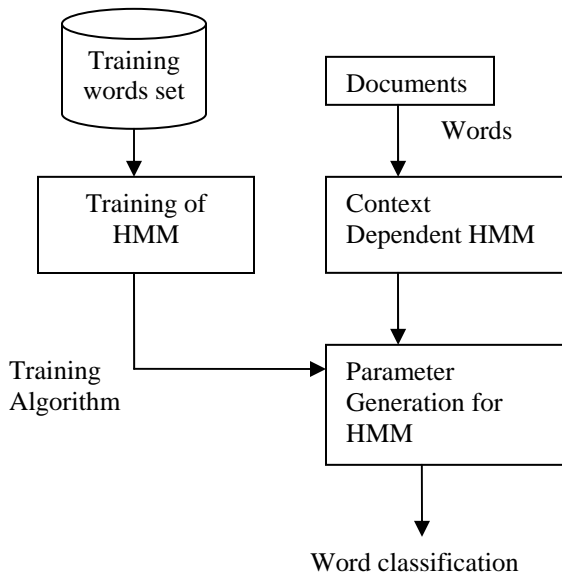


Figure 1. HMM Based Word Identification

3. HMM Architecture

Here a word with length 'n' is an input and probability of existence is the output. The training set is made by choosing words from dictionary. We used a training set with 4006 words. For 'start' state there is transition probability vector of size 1x26. T0 is a vector with number of words starting

with 'a', 'b' etc. P0 is the initial transition probability vector from the 'start' state. Figure 2 shows the architecture of the HMM for word identification. The states at each level are marked from 1 through 26. There is a start and stop state. The diamond shaped state, named as 'Q', is called *Flanking* state [Rivas and Eddy, 2001]. The states S1 and S2 are called switching states. In this model we take a maximum length of 15 letter words. The control moves to switch states, both S1 and S2, when the word length is greater than 15. Otherwise the control enters to S2 and stops the move. The transition probability is 1 for this switching. There are 300 words, in the training set, starting with 'a', so that $300/4006=0.0749$ is the first element in P0. Similar is the calculation for all other letters.

Transition probability from other states can be calculated by same method. There are state transitions and 'end' state transitions. There are two switching states S1 and S2. So transition probability is a 1x27 vector for each state. Similarly all states produce a vector of same size. The transition values becomes a matrix with size of 26 x 27, say P1. Each state except 'start' and 'end' states emits a letter. State 1 gives 'a' state 2 gives 'b' and so on. If the first letter is 'a', there may be different transition values from state 1. Consider the word 'abacus', then there is a transition from 'start' to a1, followed by b2, then b2 to c1 again, then c1 to d3, then d3 to e21, then e21 to f19 and finally f19 to 'end'. The word count is taken from the training set by first letter 'a' followed by 'b'. For the word 'abacus' it is 1. Similarly for all words, each pair of letter is considered. T1 (1:26, 1:26) is matrix with the word count of all letters. T1 (1, 1:26) is the vector with word count start letter 'a' followed by all letters ('a' to 'z'). For example, there is word count of 81 'ab' combinations in the training set. The last column shows the transition from a letter to 'end' state. The transition from a state is divided by total transition from each letter to give the probability values in P1 matrix. T2 (1:26) is matrix with word count of ending letters in the training set. For example, there are 19 words ends with 'a' and 877 words ends with 'e' in the training set. So the transition probability from state 'a' to 'end' state is $19/4006 = 0.0047$ and from 'e' to end is $877/4006 = 0.2189$. These values are stored in the last column of P1 (27th column).

The emission probability is calculated as follows. There are 26 hidden outcomes in each state. There are 15 levels in our model, so the emission probability is a 15x26 matrix. The first value is, the number of 'a' emitted at level 1 divided by total number of letters emitted at the same state. There are 300 numbers of 'a' emitted at state a1 because our training set contains 300 words starting with 'a'. Hence the emission probability is $300/4006= 0.0749$. The emission probability of 'z' at level one state is $2/4006=0.0005$. At level 2 the emission probability of 'a' is $190/4006= 0.0474$. Similar calculation for all other emissions. At level 15 the number of emission is less. The emission probability of a letter is taken as 0.0001, if there is no emitted letter in that state. Output probability is the multiplication of transition

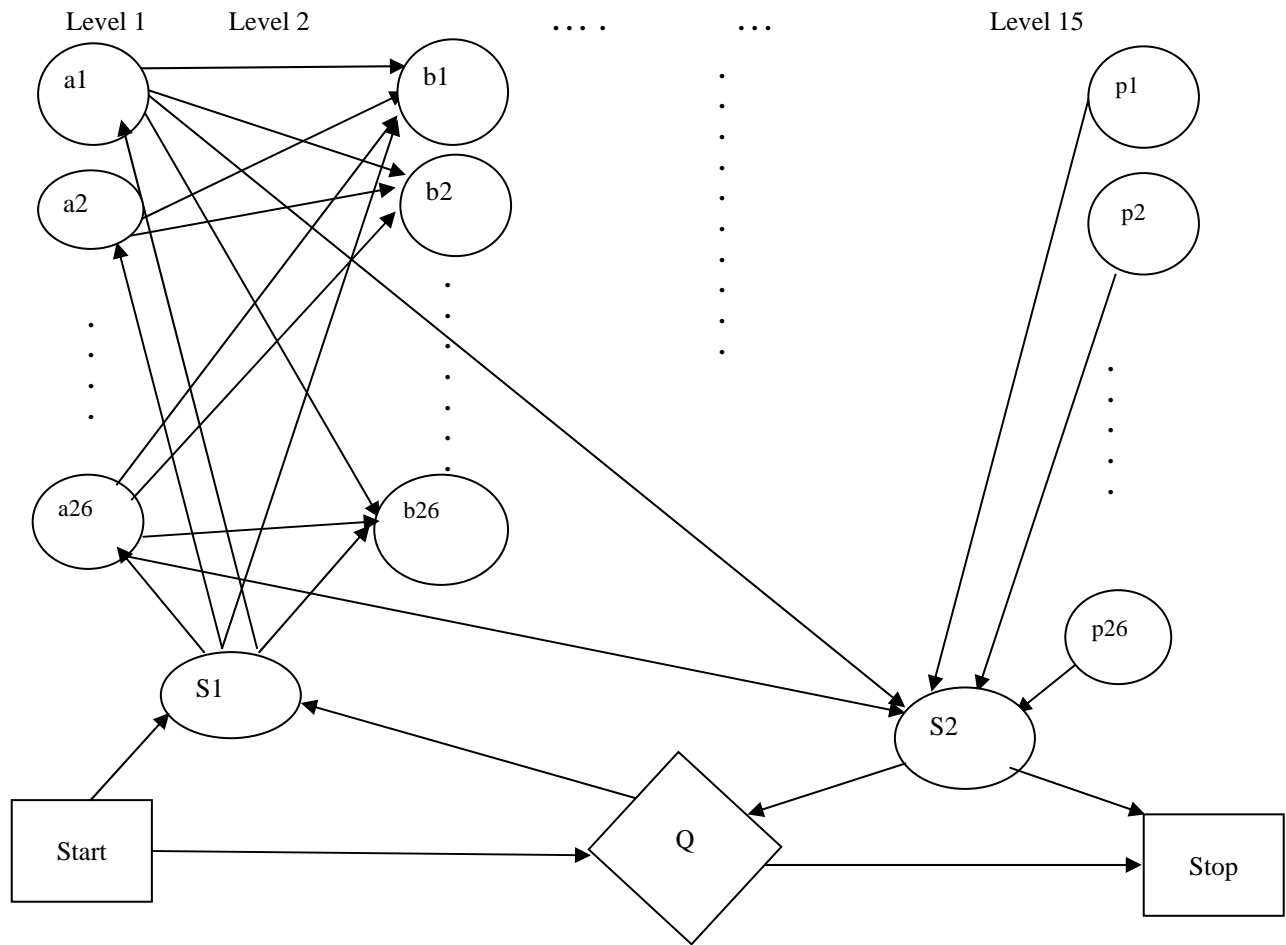


Figure 2. HMM for word identification

and emission probabilities. This value is too small, so logarithmic value is taken. The higher value means a lower chance to exist. For example, the output probabilities of the words ‘maski’ and ‘about’ are given below.

Output probability (‘maski’) = $\text{Log} (0.0434 \times 0.0434 \times 0.0690 \times 0.0175 \times 0.1231 \times 0.0937 \times 0.0651 \times 0.0544 \times 0.0125 \times 0.0134 \times 0.0002) = \text{Log} (4.4990e-019) = 42.2453$ (Low probability)

Output probability (‘about’) = $\text{Log} (0.0749 \times 0.0749 \times 0.3537 \times 0.1161 \times 0.0266 \times 0.0163 \times 0.1830 \times 0.0482 \times 0.0388 \times 0.0381 \times 0.1258) = \text{Log} (9.6061e-015) = 32.2764$ (High probability).

The *Baum-Welch* algorithm [Lucke, 1996; Shu *et al.*, 2003] is a good method for training a Hidden Markov Model. In *Baum-Welch* algorithm, the transition probability is calculated as the expected number of times each transition is used, given the training data. It begins with reasonable guesses for an initial model then calculates a score for each sequences in the training set over all possible paths though

this model. During the next iteration, a new set of expected emission and transition probabilities are calculated.

The updated parameters replace those in the initial model and the training sequences are scored against the new model. This process is repeated until there is a very little change in parameters between iterations. We used a modified form of *Baum-Welch* algorithm. Initial probability matrix is calculated using our training set. This training set ranges a word length of 2 to 15. If a ‘highly probable’ word is outputted by our model, then this word is added into the training dictionary if this word does not exist in the current training set. Using the new training set, the transmission and emission probabilities are recalculated. The *Viterbi* Algorithm [Kavcic and Moura, 2000; Shung *et al.*, 1993] is a recursive algorithm which reduces the complexity of computations drastically. This algorithm is used to find the probability of occurrence of words in a given text by traversing through multiple paths. The most promising path is only taken for probability calculation. In this work we used *Viterbi* algorithm. There are multiple paths generated for each word in a document. The *Viterbi* algorithm finds the best path and calculates the probability. There are

multiple output probabilities for the words ‘maski’ and ‘about’, by calculation. The *Viterbi* algorithm chooses the best values from this set.

4. Results and Conclusion

In this paper, we presented an HMM based approach for identifying noise arising due to local language words in roman script in a document. We tested the model successfully on real data. Table 1 shows some words and its existence probability in logarithmic value and comments. The comments are made by the comparison with other probabilities. We took a right value from a calculated probability set by *Viterbi* algorithm. This table contains some words in Malayalam language. These words can be identified, using the HMM described above. The proper nouns like ‘Sankar’ and abbreviations like ‘ISO’ are identified by this Model. For comparing the words, the output probability is multiplied by $10^{(\text{length of the word})}$ so that it gives a normalized value irrespective of the length of the word. The words like ‘kwsp’ or ‘qt’ are error words in a text. The words in a document can be marked based on the output probability. A learning curve is plotted for number of training words ranging from 500 to 5000. Figure 3 shows this learning curve. The positive logarithmic probability value of the words and number of words in the training set are y and x axes respectively. The positive log (probability) values obtained are 76.8 to 3.1 for a set of words. It is possible to fine tune the model reported here and test it out on a more broad based corpus and develop into a ‘linguistic coloring’ tool that can be added to text editors. Further the model can be trained for proper nouns, acronyms and abbreviations so that these would not be treated as noise. Compared to conventional dictionary based methods, this method is far superior in terms of time and space complexity. Noisy English words and regional language words can also be differentiated in future by redesigning the model and suitably training the model with regional language words too.

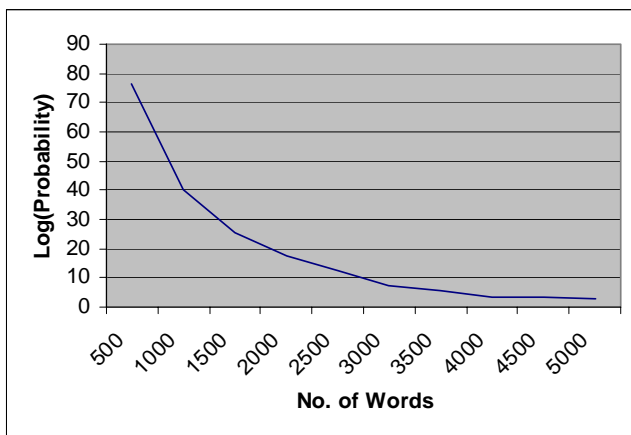


Figure 3. Learning Curve

Word	Probability (log value)	Comments
is	3.1582	Very High
was	3.5058	Very High
message	3.2685	Very High
chayakkada	20.5833	Very low
land	2.5204	Very High
manushayan	7.7762	Low
luttappi	14.2514	Very low
bride	2.0880	Very High
kwsp	10.4705	Very low
qt	11.4726	Very low
‘ISO’	5.6630	Low
‘Manmohan’	9.6825	Low
abacus	4.5184	Very High
probability	4.6105	Very High
energy	4.8171	Very High
pashu	8.2335	Low

Table 1. Sample Words and Probability

5. References

- [Jelinek, 1976] Frederik Jelinek. Speech Recognition by Statistical Methods. Pages 532-556. *Proceedings of the IEEE, Vol 64*, April 1976.
- [Rabiner, 1989] Rabiner L. R. A Tutorial on Hidden Markov Model and Selected Applications in Speech Recognition, Pages 257-285. *Proceedings of IEEE*, 77(2), 1989.
- [Denoyer *et al.*, 2001] Ludovic Denoyer, Hugo Zaragoza and Patrick Gallinari. HMM-based Passage Models for Document Classification and Ranking, Pages 1-12, *23rd BCS European Annual Colloquium on Information Retrieval*, 2001.
- [Chen, *et al.*, 2006] David Chen, Hans-Michael Müller and Paul W Sternberg. Automatic document classification of biological literature, Pages 1-11, *BMC Bioinformatics*, August 2006.
- [Denoyer *et al.*, 2003] Ludovic Denoyer, Jean-Noel Vittaut, Patrick Gallinari, Sylvie Brunessaux and Stephan Brunessaux, *Structured Multimedia Document Classification*, Pages 1-8, *DocEng’03*, November 20–22, ACM, 2003.
- [Pechwitz and Maergner, 2003] Mario Pechwitz and Volker Maergner. HMM Based Approach for Handwritten Arabic Word Recognition Using the IFN/ENIT – Database, *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR 2003)*
- [Steyvers *et al.*, 2004] Mark Steyvers, Padhraic Smyth, Michal Rosen-Zvi and Thomas Griffiths. Probabilistic Author-Topic Models for Information Discovery, *10th*

ACM SigKDD conference knowledge discovery and data mining, August 22-25, 2004. Washington, USA.

- [Rivas and Eddy, 2001] Elena Rivas and Sean R Eddy, Noncoding RNA gene detection using comparative sequence analysis, Pages 1-19, *BMC Bioinformatics*, October 2001.
- [Lucke,1996] Lucke, H, Which stochastic models allow Baum-Welch training?, Pages 2746 – 2756, *IEEE Transactions on Speech and Signal Processing*, Vol 44, Nov 1996.
- [Shu *et al.*,2003] Han Shu, I. Lee Hetherington, and James Glass, Baum-Welch Training For Segment-Based Speech Recognition, Pages 43-48, ASRU, IEEE Transaction 2003.
- [Kavcic and Moura, 2000] Aleksandar Kavcic, and Jos M. F. Moura. The Viterbi Algorithm and Markov Noise Memory, Pages 291-301, *IEEE Transactions On Information Theory*, Vol. 46, Jan 2000.
- [Shung *et al.*,1993] C. Bernard Shung, Horng-Dar Lin, Robert Cypher, Paul H. Siegel, and Hemant K. Thapar, Area-Efficient Architectures for the Viterbi Algorithm-Part I: Theory, Pages 636-644, *IEEE Transactions On Communications*, Vol. 41, April 1993.