A Report on Tamil Paraphrase Identification System

Name: Praveena. R

6-10-2017

Introduction

Paraphrase identification system has to identify if any two sentences convey same meaning or not. Paraphrase identification finds many important applications in the field of Natural Language Processing (NLP) like MT evaluation, test summarization, information retrieval, question answering etc. Deep semantic understanding is required for getting better performances of complex NLP problems like paraphrase identification. The formal definition for paraphrases and non paraphrases are given as follows:

An alternate representation of existing sentence produces paraphrases. Thus any two sentences having same meaning although they are in different form are known as paraphrases. Consider the example given below. S1 & S2 are paraphrases, justifies the semantic overlapping of S1 and S2 is high. Pair of sentences which do not convey same meaning is termed to be non paraphrases. S3 & S4 are non paraphrases because although the two sentences contain same words, it do not imply same context.

S1: கேரள மாநிலம் திருச்சூரில் கூடல்மாணிக்கம் கோயில் திருவிழா துவங்கியது.
[ ]
S2: கூடல்மாணிக்கம் கோயில் திருவிழா கோலாகலமாக துவங்கியது.
[ ]
S3: தேர்தலுக்கு பிறகு அ.தி.மு.க. பணம் கொடுத்தது என கூறிய தயாநிதிக்கு டுவிட்டரில் எதிர்ப்பு.
[ ]
S4: அ.தி.மு.க.வினர் பணம் கொடுத்து, வெற்றிபெற்றுவிட்டதாகக் கூறிய தயாநிதி மாறனை, டுவிட்டர் பதிவர்கள் கலாய்த்து வருகின்றனர்.
[ ]

Dataset Description

Data	**2-class**
Training	5000
Testing	7000
Average number of words	
Sentence 1	11.092
Sentence 2	12.044
Pair	11.568

Table 1: Dataset Description

Initially, there were no freely available dataset for this task. Therefore, an unlabeled corpus consisting of 1, 20,000 sentences are collected from various web sources. These sentences are then used for word embedding. Another corpus consisting of 8000 sentences that are paraphrases and non paraphrases are collected for this paraphrase identification task explained in this paper. The corpus mentioned later is so called supervised one and is split up into training and testing sets, and are tagged as they fall into respective categories. These were manually collected and then preprocessed to remove redundancies. Such collected sentences are then annotated using labels 0 and 1 for non paraphrases. Table 1 describes about the dataset used for paraphrase identification discussed in this paper. The average number of words in source sentences and target sentences in both problems individually as well as pair wise is also depicted in the above table. Source sentences and target sentences of 2-class problem had an average number of words as 11.092 and 12.044 respectively. The average number of words in a pair for 2 class problem is 11.568.

Methodology

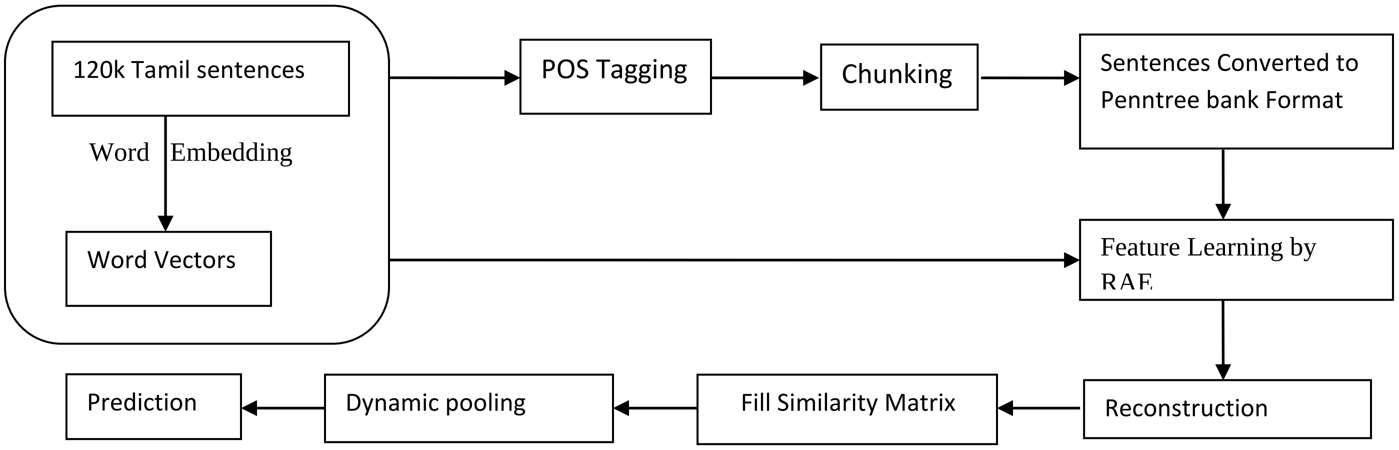

Fig 1: Overall Working of Tamil Paraphrase Identification system

The primary experiments carried out for producing a base line system is depicted in the fig 1. The word vectors obtained from a large collection of Tamil sentences along with supervised sentences falling in paraphrase, and non paraphrase categories in Penn treebank format underwent a RNN training to generate phrase/sentence embedding. The RNN architecture used is unfolding recursive autoencoders. These sentences in varying length are altered to be in fixed size using dynamic pooling. The output of dynamic pooling phase is passed to a classifier for predicting the category into which test sentences belong to. Various procedures by which the system is made are explained in the following subsections.

Parsing Using Chunker

Parsing phase produces a tree representation, called parse trees for all labelled sentences. The output of parser is the sentences divided into chunks consisting of noun phrases and verb phrases. Due to the unavailability of openly available accurate Malayalam parser, all sentences are initially POS tagged. The POS tagged sentences got chunked using an in-house chunker. The chunked sentences are then converted into Penn treebank format. Syntactic information for sentences is captured by this process of converting sentences to Pen treebank format.

Recursive Autoencoders

Recursive autoencoders (RAE) are used for learning features from the nodes of parse tress. The aim of RAE is also extended to find the corresponding vector representations for phrases of different or varying sizes that lie in each node of parse trees. RAE uses word vectors obtained from the word embedding phase for generating vector representations of phrases, having the binary parsed tree as input in the form $P \rightarrow (\text{word}1, \text{word}2)$. The trees are obtained from parsing the sentences which is explained in subsection 4.2. The Fig 2 shows how an RAE works for a sentence with four words. In the figure PV and SV stands for phrase vector and sentence vector respectively.

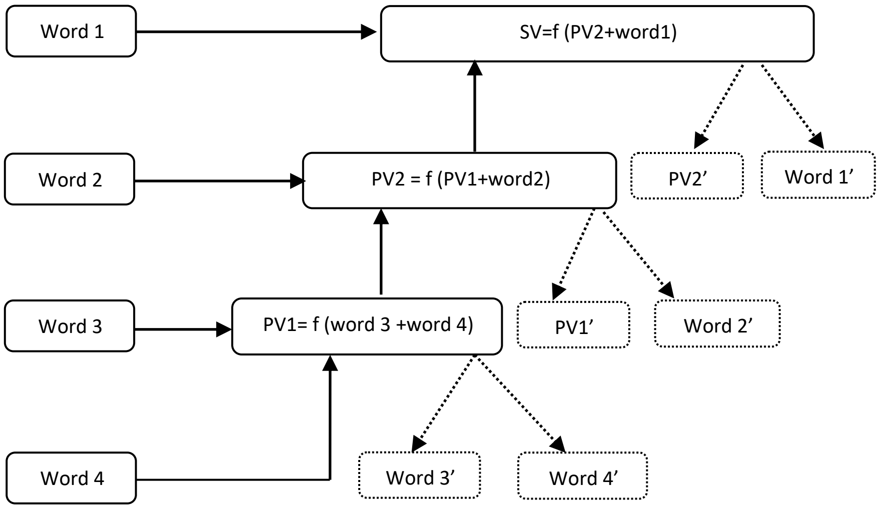

Fig.2 Working of Recursive Autoencoders

RAE produces phrase vectors in an unsupervised way. It is possible to reconstruct parent node from child representation. For example, parent P is computed from its children $word\,1\,and\,word\,2$ by a standard neural network layer using $\mathrm{p} = \mathrm{f}(\mathrm{We}[\mathrm{c}_1; \mathrm{c}_2] + \mathrm{b})$ where $[c_1; c_2]$ is the concatenation of $\mathrm{c}_1\,\mathrm{and}\,\mathrm{c}_2$. After multiplying the parameter *We* with the concatenation of $\mathrm{c}_1\,\mathrm{and}\,\mathrm{c}_2$, a bias term is added. The resulting vector is applied to some element wise activation function f such as *tanh*. To know how effectively the vectors are created, the word vectors and phrase vectors are reconstructed as in Eq: 3.

$$[\mathrm{c}_1^{'}; \mathrm{c}_2^{'}] = \mathrm{We}^{'}\; p + b^{'} \qquad (3)$$

The Euclidean distance between the original input and reconstructed defines reconstruction error. The objective of reconstruction phase is to have minimum reconstruction error which is obtained by Eq: 4.

$$E_{\mathrm{rec}}(\mathrm{p}) = \left\| [c_1; \mathrm{c}_2] \right\| - \left\| [c_1^{'}; \mathrm{c}_2^{'}] \right\|^2 \qquad (4)$$

In our experiment, we used a variation of standard recursive autoencoders which is called unfolding recursive autoencoders. They differ from standard RAE only in the reconstruction step. The unfolding RAE will reconstruct the entire children beneath a node when a standard RAE reconstructs only the direct children. Mathematically, the reconstruction happening in unfolding RAE is defined as in Eq: 5.

$$E_{\mathrm{rec}}(\mathrm{p}) = \left\| [x_i; ....; x_j] \right\| - \left\| [x_i^{'}; ...; x_j^{'}] \right\|^2 \qquad (5)$$

We used a subset of parsed trees from the whole labeled dataset and the sum of reconstruction error of every tree is minimized.

Dynamic Pooling

Sentences considered for training and testing will be of different size. In order to fix the sentence length, we use the concept of dynamic pooling. A similarity matrix *S* is produced from the Euclidean distance calculated for checking the reconstructed error. We know that sentences are of unequal length and hence the similarity matrix S also will be of unequal dimension. For bringing sentences to be in a fixed size, the similarity matrix S is mapped to get a matrix S_p and it is divided into roughly equal number of rows and columns. The S_p is set to have a pooling region and minimum value among them is chosen.

Conclusion

This paper presents paraphrase identification for Tamil. The sentences are mapped to their corresponding vectors using RAE embedding. Word2vec embeddings are used for obtaining word vectors. On the way it captures the phrase vectors also which serves as the features to be learnt by the RAE. Since the sentences vary in their length, we apply dynamic pooling on the sentences to make it into a fixed size representation. This new set of vectors is then passed to logistic regression. Paraphrase identification still stays as a complex yet challenging problem in the field of NLP. The approach used in the proposed method is capable of disambiguating word level ambiguities in the identification of contexts for detecting paraphrases. We could obtain the state of the art performance for Tamil paraphrase identification system, but still there exist a room for improvement. It is possible to extend the current system by adding new feature sets along with different embedding techniques and different dimensional vectors. Implementation of same approach for other morphologically rich languages is possible in the future. It helps to understand how the morphological aspects of a language effects in identifying paraphrases whose underlying idea is to find the semantic similarity between sentences. We presented chunking based paraphrase identification in the paper. Instead POS tag based paraphrase identification can also be implemented in future.