BITS_PILANI@DPIL-FIRE2016:Paraphrase Detection in Hindi Language using Syntactic Features of Phrase

Rupal Bhargava[1] Anushka Baoni[2] Harshit Jain[3]

Yashvardhan Sharma[4]

WiSoc Lab, Department of Computer Science
Birla Institute of Technology and Science, Pilani Campus
Pilani-333031

{rupal.bhargava[1], f2013683[2], f2013289[3],yash[4]} @pilani.bits-pilani.ac.in

ABSTRACT

Paraphrasing means expressing or conveying the same meaning or essence of a sentence or text using different words or rearrangement of words. Paraphrase detection is a challenge, especially in Indian languages like Hindi, because it is very essential to understand the semantics of the language. Detecting paraphrases is very relevant in real life because it has a lot of importance in applications like Information Retrieval, Extraction and Text Summarization. This paper focuses on using Machine Learning classification techniques for detecting paraphrases in Hindi language for the DPIL Task in Fire 2016. A feature vector based approach has been used for detecting paraphrases. The task involves checking whether a given pair of sentences conveys the same information and meaning even if they are written in different forms. Given a pair of sentences in Hindi, the proposed technique labels whether the pair of sentences are Paraphrases (P), Semi-Paraphrases (SP) or Not Paraphrases (NP).

CCS Concepts

•**Information systems** → **Summarization;** *Information integration; Data analytics; Data mining;*

Keywords

Paraphrase Detection, Text Summarization, Classification, Machine Learning

1. INTRODUCTION

The word 'paraphrase' means rephrasing or restating the meaning of a paragraph or text using some other words or vocabulary. Paraphrase detection is an important task for many natural language processing applications. Some of the applications involve question-answering systems, machine translation systems, systems used for plagiarism checks, finding similarities between sentences, text summarizers, etc. Plagiarized texts usually copy phrases as it is or replace some words with similar words. Paraphrase detection will help in detecting plagiarized work and ensure that the documents written are unique and not copied. Question Answering system makes use of paraphrases to find the correct answers to asked questions. A lot of work has been done in paraphrase detection for English language. However for Hindi and other Indian languages, not much work has been done and there is a lot of scope for research. The most common way of detecting paraphrases is modeling the problem as a classification problem. This paper implements a supervised classification model for detecting Paraphrases. POS Tags, Stems of Words and Sound-ex codes corresponding to the words in sentences are used as features.

The rest of the paper is organized as follows: Section 2 discusses related work in the area of Paraphrase Detection. Section 3 presents the analysis of the Data set provided by DPIL task organizers. Section 4 discusses the methodology used and Section 5 explains the proposed algorithm. Section 6 gives a detailed analysis of the results obtained and error analysis. Section 7 presents the conclusion and possible future work.

2. RELATED WORK

Paraphrase detection has been a major area of research in the recent times because of its significance in many areas of Natural Language Processing. Few of the approaches adopted for English language are mentioned in this section. Huang et al. [4] has proposed an unsupervised recursive auto-encoder architecture for paraphrase detection. The recursive auto-encoder uses *tanh* as the sigmoid-like activation function and gives the representation of sentences along with their sub-phrases. These representations are then used for paraphrase detection. To extract the same number of features for different sentence pairs, two approaches are used, aggregating representations to form a single feature and using a similarity matrix approach. With first approach they achieved 66.49% accuracy while with the second method accuracy of 68.06% was achieved . Kotti et al.[10] also proposed an unsupervised feature learning technique with Recursive Auto-encoders (RAE) for detecting paraphrases on twitter. In their proposed technique they first converted data to parse trees using phrase-structure parser and then passed it to the RAE for training. The vector generated from the RAE is converted to form a similarity matrix and thus paraphrase detection is done using this matrix. Fernando et al.[3] presented an algorithm using word similarities whereas Ngoc et al. [11] proposed simple features like n-grams, edit distance scores, METEOR word alignment, BLEU for detecting paraphrases and semantic similarity tasks on twitter data. Similarly, analysis of various similarity measures like sentence-level edit distance measure, simple n-gram overlap measure, exclusive longest common prefix (LCP) n-gram measure, BLEU measure and sumo measure along with a paraphrase detection based on abductive machine learning has been proposed in [2]. Sethi et al. [9] proposed a technique for paraphrasing or re-framing Hindi sentences using NLP. The main steps involved dividing the paragraph into sentences, tokenizing the sentences into words, applying reframing rules and then combining the results to form new paragraphs. Malakasiotis et al. [5] proposed three methods for paraphrase detection using string similarity measures.

3. DATA ANALYSIS

The data-set provided by the task organizers [1] is from newspaper domain and contains pairs of sentences. There are two Subtasks and each Subtask has its own training and testing data.

3.1 SubTask 1

The pairs of sentences in the Training Data set contains 1000 'Paraphrases' (P) and 1500 'Not Paraphrases' (NP). Test Data set for SubTask 1 consisted of 900 pairs for Hindi Language.The number of paraphrases with common words versus the number of common words is shown in Figure 1. For e.g A point (5,72) represents that there are 72 such paraphrases which have five common words.

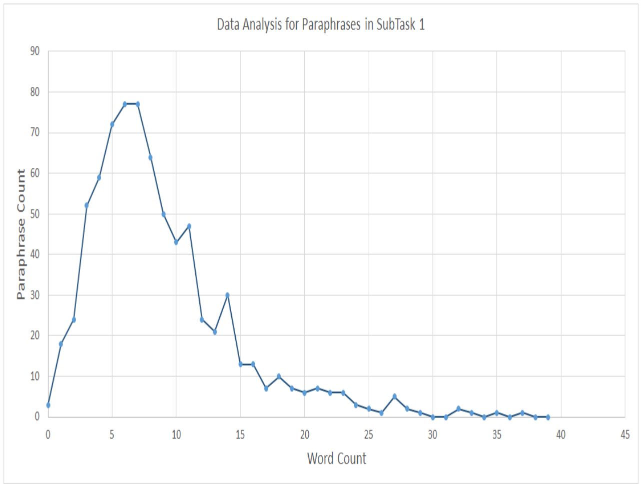

Figure 1: Data Analysis of Paraphrase for SubTask 1

3.2 SubTask 2

For Subtask 2,Training Data set consisted of 1000 pairs of sentences that are Paraphrases (P), 1000 pairs that are Semi-Paraphrases (SP) and 1500 that are Not Paraphrases (NP). For Test Data set, 1400 pairs of Hindi sentences were provided.The number of Paraphrases and Semi-Paraphrases with common words versus the number of common words is shown in Figure 2.

4. PROPOSED TECHNIQUE

The proposed work has been divided in multiple phases as shown in Figure 3. Initially pre-processing of the data

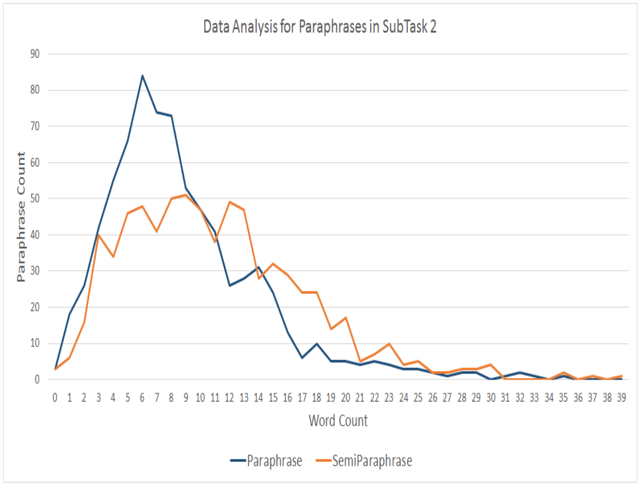

Figure 2: Data Analysis of Paraphrase and Semi Paraphrase for SubTask 2

is done which involves converting the xml format Data Set into csv format so that the data can be read from the csv file and processed for extracting features.

Second phase processes the training data to extract important features from the data so that the proposed classification model could be trained. The following three features were extracted for the proposed training model:

1. **POS Tags:** POS (Part-Of-Speech) Tags are labels that are given to words to identify the part of speech or lexical categories of words. The eight parts of speech are: the verb, the noun, the pronoun, the adjective, the adverb, the preposition, the conjunction, and the interjection. Words that have the same POS Tags play similar roles in the grammatical structure of sentences. For obtaining the respective POS tags for the Hindi words, RDRPOSTagger[1] [6] was used. The input passed to the RDRPOSTagger contains the pairs of sentences and the output generated by RDRPOSTagger had the respective POS Tags next to each word. Only the POS Tags corresponding to each word in the sentence are extracted from the output and appended to form a string thus obtaining POS Tags for each sentence in the data set.

2. **Stem of the words:** Stemming is a process of extracting the 'word stem' or 'root' of the word. For extracting the stem of the Hindi words, a Hindi stemmer[2] was used which implements the suffix-stripping algorithm described in [8]. A string for each sentence in the data set with the corresponding stems of the Hindi words is then obtained.

3. **Soundex codes:** Soundex is a phonetic algorithm for indexing names by sound as pronounced in English. Soundex[3] provides an implementation of the modified version of soundex algorithm for Indian languages including Hindi. This package is used for the

[1]https://rdrpostagger.sourceforge.net
[2]http://research.variancia.com/hindi_stemmer/
[3]https://pypi.python.org/pypi/soundex/

corresponding soundex codes for the words in the sentences. Using soundex codes for words in the sentence, a string comprising of soundex codes corresponding to each sentence is generated.

After extracting these three features, the similarity scores corresponding to each feature has been calculated. The python package, fuzzywuzzy [4] is used to calculate the similarity scores. Each similarity score lies in the range [0,1] and uses Levenshtein Distance to calculate the differences between string sequences. The Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. The similarity score is calculated for each pair of POS Tags sentences (feature 1), sentences with stem of the words (feature 2) and sentences with soundex codes corresponding to the Hindi words (feature 3) hence creating a feature vector with the similarity scores corresponding to the sentence pair.

After feature vector generation, different machine learning techniques are used for training so that the best model for predicting the labels could be chosen after analysis. For SubTask 1 and SubTask 2, Logistic Regression, Naive Bayes, Random Forest Classifier and Support Vector Machine were used for classification. These models were implemented using the python library sklearn [7].

5. ALGORITHM

Algorithm 1 takes Paraphrases as input where each Paraphrase(P[i]) contains two Hindi sentences (P[i].Sentence)and outputs a Label for its corresponding Paraphrases. The functions PosTags, WordStem and Soundex, each take Sentences of Paraphrase as its parameter and return the array of corresponding POS Tagged Sentences, WordStem Sentences and Sentences with Soundex Codes respectively. SimilarityScore generates the similarity score for each of its input array. SimScore1, SimScore2 and SimScore3 are the individual vectors for the three features, which are then passed to the CreateVector function to form the final FeatureVector. Classifier function takes the FeatureVector as input, assigns labels to the Paraphrases and then returns a LabelVector. Classifier function implements different models (Logistic Regression, Naive Bayes, SVM and Random Forest) for predicting labels.

6. EXPERIMENTS AND RESULTS

6.1 Evaluation and Discussion

To test the accuracy and F-measure, data set provided by the task organizer was divided into a ratio of 75% and 25% for training and testing respectively. The results (Accuracy and F-Measure) were evaluated using sklearn [7] for the different models (Logistic Regression, Naive Bayes, SVM and Random Forest). Results obtained for SubTask 1 is shown in Figure4. Proposed system gave an accuracy of 90.4% and F-measure 87.6% for Logistic Regression followed by Naive Bayes and Random Forest, both with 89.5% accuracy. For binary classification problems, logistic regression gives the best results in most cases because it assigns labels by calculating odds ratio and then applies a non-linear log transformation. Moreover, the performance can be fine-tuned by changing and adjusting parameters in the functions provided by sklearn [7] for Logistic Regression. As SubTask 1 was a binary classification problem hence results obtained via Logistic Regression were better than the others. On the other hand, SubTask 2 was a multi-class classification problem (Labels-P, NP or SP). Hence in this case, Random Forest gave the best results with 69.2% accuracy and 68.8% F-measure followed by Naive Bayes (64.6% accuracy and 62.4% F-measure) as shown in Figure 5. Random Forest calculates labels by using sub samples of the data set and uses averaging to improve the accuracy whereas Naive Bayes uses a conditional probability approach for assigning labels.

[4]https://pypi.python.org/pypi/fuzzywuzzy

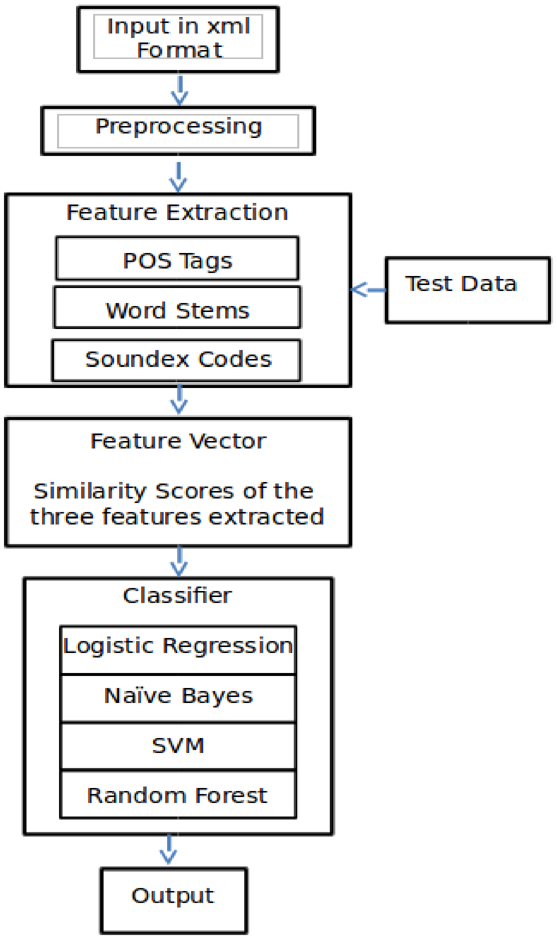

Figure 3: Block diagram for Paraphrase Detection

Hence runs submitted for SubTask 1 used Logistic Regression classifier and SubTask 2 used Random Forest. As per the final results declared by the Task organizers, the proposed technique was ranked third when compared with other teams with Accuracy of 0.897 and F-measure of 0.89 as shown in Figure 6 and 7 respectively for SubTask 1. In SubTask 2, the proposed technique is ranked fifth with Accuracy and F-measure of 0.717 and 0.712 as shown in Figure 8 and Figure 9 respectively.

6.2 Error Analysis

Few errors that could have attributed to the decrease in evaluation measures can be-

Algorithm 1 Algorithm for Detecting paraphrases

```
 1: Input: Paraphrase P, where all paraphrases have a
    unique id and contains two sentences (Hindi)
 2: Output: LabelVector gives the corresponding labels for
    the paraphrases. Depending upon the task it can have
    value of P, NP and SP
 3: Initialization: SimScore1[]=0,SimScore2[]=0,SimScore3[]=0
 4: for i=0 to P.Count do
 5:     Pos[]=PosTags (P[i].Sentence)
 6:     Stem[]=WordStem (P[i].Sentence)
 7:     Sound[]=Soundex (P[i].Sentence)
 8:     SimScore1.append (SimilarityScore(Pos[]))
 9:     SimScore2.append (SimilarityScore(Stem[]))
10:     SimScore3.append (SimilarityScore(Sound[]))
11: end for
12: FeatureVector=CreateVector(SimScore1, SimScore2,
    SimScore3)
13: LabelVector=Classifier(FeatureVector)
```

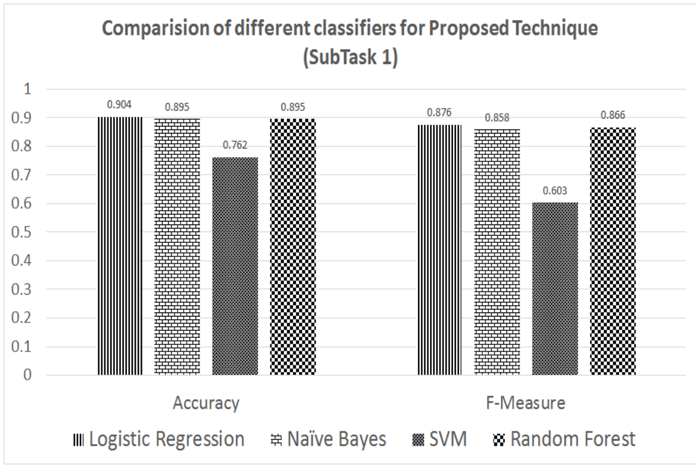

Figure 4: Results for Subtask 1 using different classifier for proposed system

1. RDRPOS Tagger- Nguyen et al.[6] states that the RDRPOSTagger achieves a very competitive accuracy in comparison to the state-of-the-art results. But a different Hindi POS Tagger can also be used to improve this phase. Also RDRPOSTagger can be combined with an external initial tagger to increase its accuracy.

2. Similarly, the Hindi Stemmer used might have incorrectly returned the stem words, which can be a reason for wrongly classified Paraphrases. The algorithm for extracting the root words can be improved further to better the results.

3. Other factors that could have led to errors are accuracy of soundex library and similarity measure used.

7. CONCLUSIONS AND FUTURE WORK

In this paper, a feature vector based approach with three features (POS Tags, Word Stems and Soundex codes) is discussed for paraphrase detection of Hindi Language. Levenshtein Distance was used to calculate the similarity measure. Proposed system achieved accuracy of 89.7% and F-measure of 89% for SubTask 1 using Logistic Regression. For SubTask 2, proposed system gave an accuracy of 71.7% and F-measure of 71.2% using Random Forest Classifier as evaluated by task organizers. The model accuracy can be further improved by incorporating more features like calculating similarity between two strings having only nouns of the original sentences as identified by the POS Tagger, replacing the nouns by their soundex codes or their stems. Only verbs of the original sentences can also be used to obtain features where the verbs are replaced by their soundex codes or stems. The current model has been trained on the data set provided by task organizers. We can incorporate more data to extend the model. Using an ensemble classifier and combining different models like Decision Trees, Naive Bayes, SVM, etc. can be used for predicting labels that may further improve results. Moreover the proposed technique only uses syntactic features, semantic features can be incorporated for improvising the algorithm.

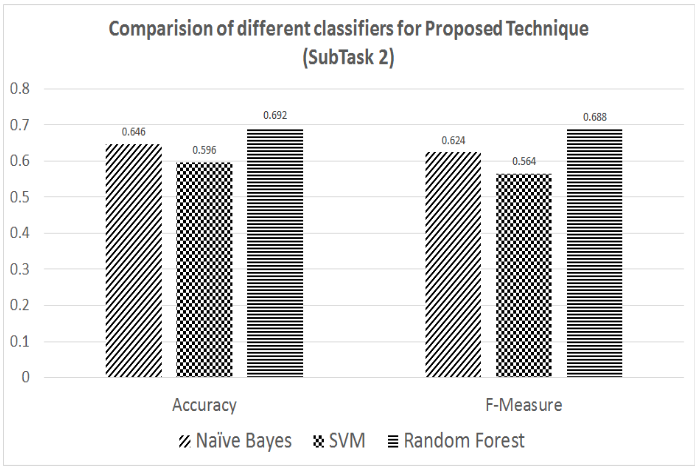

Figure 5: Results for Subtask 2 using different classifier for proposed system

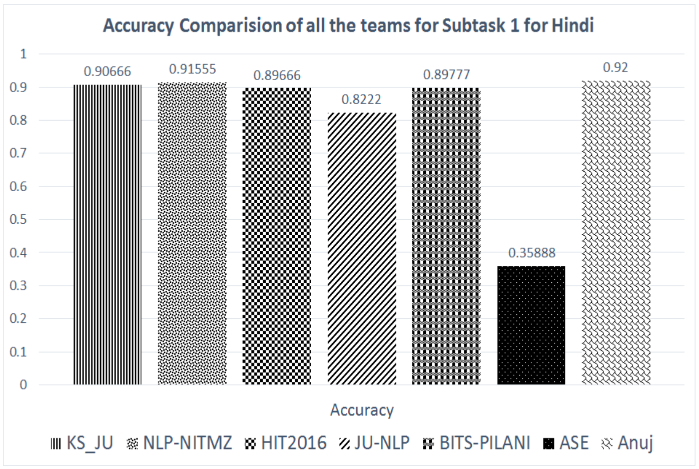

Figure 6: Accuracy comparison for all teams in SubTask 1

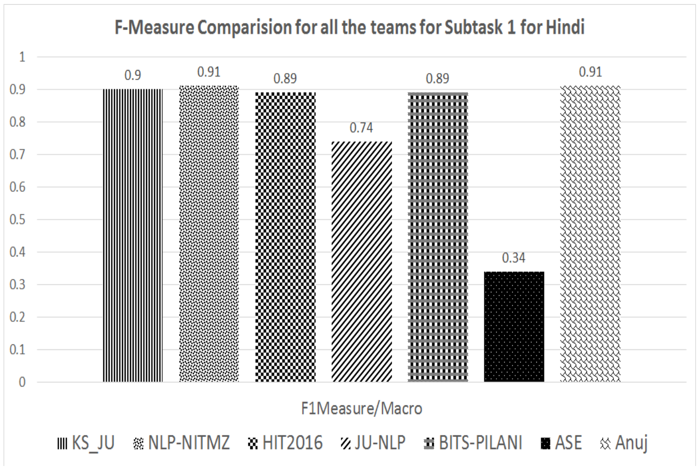

Figure 7: F-Measure comparison for all teams in SubTask 1

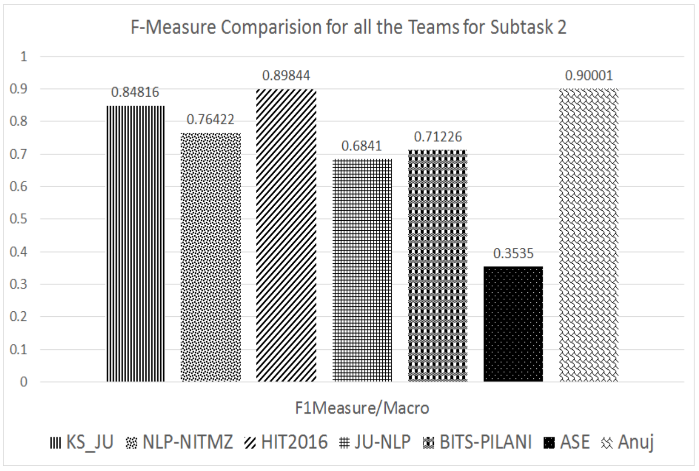

Figure 9: F-Measure comparison for all teams in SubTask 2

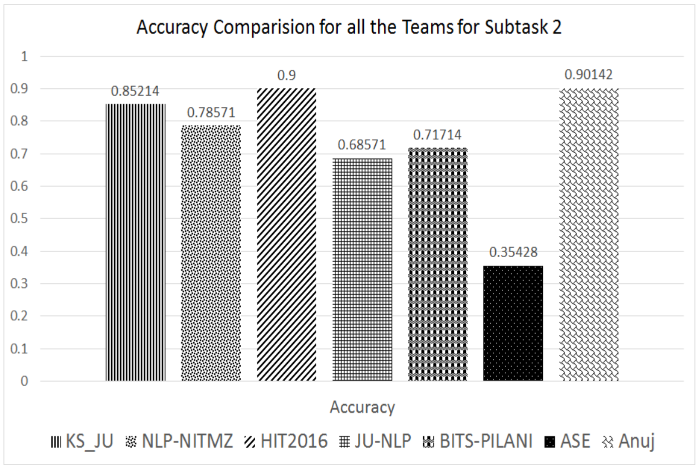

Figure 8: Accuracy comparison for all teams in SubTask 2

References

[1] M. Anand Kumar, S. Shivkaran, B. Kavirajan, and K. P. Soman. DPIL@FIRE2016: Overview of shared task on detecting paraphrases in indian languages. In *Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation, Kolkata, India, December 7-10, 2016*, CEUR Workshop Proceedings. CEUR-WS.org, 2016.

[2] E.-S. M. El-Alfy, R. E. Abdel-Aal, W. G. Al-Khatib, and F. Alvi. Boosting paraphrase detection through textual similarity metrics with abductive networks. *Applied Soft Computing*, 26:444–453, 2015.

[3] S. Fernando and M. Stevenson. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*, pages 45–52. Citeseer, 2008.

[4] E. Huang. Paraphrase detection using recursive autoencoder. *Source:[http://nlp. stanford. edu/courses/cs224n/2011/reports/ehhuang. pdf]*, 2011.

[5] P. Malakasiotis. Paraphrase recognition using machine learning to combine similarity measures. In *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*, pages 27–35. Association for Computational Linguistics, 2009.

[6] D. Q. Nguyen, D. D. P. Dai Quoc Nguyen, and S. B. Pham. Rdrpostagger: A ripple down rules-based part-of-speech tagger. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014*, pages 17–20. Citeseer, 2014.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[8] A. Ramanathan and D. D. Rao. A lightweight stemmer for hindi. In *the Proceedings of EACL*, 2003.

[9] N. Sethi, P. Agrawal, V. Madaan, and S. K. Singh. A novel approach to paraphrase hindi sentences using natural language processing. *Indian Journal of Science and Technology*, 9(28), 2016.

[10] M. S. Sundaram, K. Madasamy, and S. K. Padannayil. : Paraphrase detection for twitter using unsupervised feature learning with recursive autoencoders. In *Workshop Proceedings of the International Workshop on Semantic Evaluation 2015 (Sem Eval-2015), Denver, Colorado, US*, pages 45–50. Citeseer, 2009.

[11] N. P. A. Vo, S. Magnolini, and O. Popescu. Paraphrase identification and semantic similarity in twitter with simple features. In *The 3rd International Workshop on Natural Language Processing for Social Media*, page 10, 2015.