National College of Ireland

Detecting Paraphrases in Hindi Language Using Machine Learning

MSc Research Project
Data Analytics

Jyotsna Patel
x17108322

School of Computing
National College of Ireland

Supervisor: Soumyabrata Dev

National College of Ireland
Project Submission Sheet – 2017/2018
School of Computing

Student Name:	Jyotsna Patel
Student ID:	x17108322
Programme:	Data Analytics
Year:	2016
Module:	MSc Research Project
Lecturer:	Soumyabrata Dev
Submission Due Date:	11/12/2017
Project Title:	Detecting Paraphrases in Hindi Language Using Machine Learning
Word Count:	7053

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	
Date:	20th December 2018

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Detecting Paraphrases in Hindi Language Using Machine Learning

Jyotsna Patel
x17108322
MSc Research Project in Data Analytics

20th December 2018

Abstract

Paraphrase detection is one of the popular research domains in natural language processing these days. It has various applications like information retrieval, question answering, text summarization etc. This research focuses on implementing optimal abductive network model by using a learning paradigm for blending K-Nearest Neighbor and Decision Tree. This research is developed mainly on four stages, preprocessing, feature extraction, classifier building, and performance evaluation. For feature extraction, the semantic and syntactic analysis will be used. Preprocessing step is basically data preparation where tokens are created, and data is cleaned by removing any non-alphanumerical data and in feature extraction, features are extracted from dataset such as cosine similarity, IDF Score, POS Tagging and common N-Gram Score. Further, the chosen classifier will be blended in adaptive learning and trained and tested on the features extracted. The final step is to evaluate the result on the following performance measures: accuracy, recall, precision, and F-Score. This is the first research to deploy K-Nearest Neighbor and Decision Tree with Adaptive Boosting on Hindi Dataset. After evaluation, we observed that the KNN is performing slightly better than the Decision Tree on the given Dataset but when we voted for the best classifier, it showed that the Decision Tree is better than KNN for text classification.

Keywords Paraphrasing, Machine Learning, Adaptive learning, Hindi Languages, Bayesian Network, SVM

1 Introduction

Paraphrasing is not a new concept, but it is one of the old concepts which has been used since ages. Paraphrasing simply means that by observing the meaning of one sentence, forming another one that holds the same meaning as previously Brun and Hagège (2003). For being eligible to classify in paraphrase section, it is not necessary for sentences to be of same length, it can be of a different length as the first sentence can be ten words long whereas another sentence can be of six words only. What is necessary that both sentences should convey same meaning. There are many applications of paraphrase detection because of which it became a popular topic for research recently in most of the language available on this earth. Those applications are text summarization, question answering, semantic parsing, plagiarism detection, information retrieval etc Wu et al. (2016). Lots

of paraphrasing detection research available for the English language but there are still a few types of research available in other languages Bhargava et al. (2017). Due to globalization, now every nation′ss language has become important and that generated the demand for paraphrasing detection application in other languages also which led towards the sudden increase of paraphrasing detection research. **In this project, the research goal is to propose a novel approach of machine learning technique for tackling the problem of paraphrasing detection in the Hindi language by merging syntactic, lexical and semantic similarity measure and to boost the performance of a set of weak classifiers and combining them together for better accuracy**.

1.1 Motivation

There are several techniques available for paraphrasing detection. In this research, the techniques will be used to tackle this problem is the first similarity between strings will be measured further it will be applied to other abstractions of a pair of input phrases. Further, the sentences will use vectors bag of words which will extract the syntactic information from sentences Madnani and Dorr (2013). Further, sentences will be modeled in a way to compute similarity measures between two words and then evaluate the similarity of degree. In more simpler words, paraphrasing detection works by identifying the similarity and feeding that similarity to machine learning algorithms as an input and further evaluating the performance measure of that algorithm Blacoe and Lapata (2012). For achieving the goal of paraphrasing detection of Hindi Language, Adaptive boosting has been proposed in this research. This research will benefit the Indian States which uses Hindi as an official language and will also benefit Hindi language Scholars. As mentioned earlier that paraphrasing detection have many applications such as text summarization, question answering, plagiarism detection, information retrieval etc. For this research, the similar data will be used as it has been used in DPIL (Detecting Paraphrases in Indian Language) which was organized by FIRE (Forum for Information Retrieval Evaluation) in 2016 Kulkarni et al. (2012). This project is further divided into various section, first section is state of art. In this section, the previous work done in this domain has been critically evaluated. Next section is Scientific Methodology and Architectural Design which contains the discussion about the methodology and architecture used for the purpose of this research. Next comes, the section about data preparation which includes detail discussion about data preprocessing and feature extraction. After that, the next section is for implementation, which is about the software, tools, and techniques used in this research. And then comes Result section, this section includes evaluation of deployed models and detail discussion about results and in last the conclusion section.

1.2 Domain Overview

Paraphrasing detection is one of the most challenging Natural Language Processing task in machine learning techniques Socher et al. (2012). Recently there are lots of focus on this research domain. The nature of the challenge with Hindi Language is Preprocessing of Hindi text like tokenization, removing stop words, synonyms, calculating similarity etc. This research focus on the successful identification of the paraphrased sentences in Hindi Language and data has been requested from DPIL@FIRE2016.

1.3 Research Requirement/Specification

1.3.1 Research Question

This research addresses the problem of lack of applications available for Paraphrasing Detection in Hindi Language which is affecting the uses of Hindi globally. Research in this domain is also affected due to lack of proper packages and resources for Hindi.
RQ: The research goal of this project is to propose a novel approach of machine learning technique for tackling the problem of paraphrasing detection in the Hindi language by merging syntactic, lexical and semantic similarity measure and to boost the performance of a set of weak classifiers (K-Nearest Neighbor, and Decision Tree) and combining them together for better accuracy.
In order to achieve success in this research, the following research objectives are specified and tackled. [Figure 1]

Objectives	Description
Objective 1	Critical evaluation of past work on Paraphrasing Detection In different languages.
Objective 2	Data Preprocessing, feature generation, Implementation and Evaluation of Hindi Dataset Adaptive learning Modelling using K-Nearest Neighbor and Decision Tree algorithms.
Sub_Objective2 (i)	Hindi Dataset extracted from DPIL@FIRE2016 and Preprocessed.
Sub_Objective2 (ii)	Feature Generation from dataset.
Sub_Objective2 (iii)	Implementing and evaluation of K-Nearest Neighbor Theorem
Sub_Objective2 (iv)	Implementing and evaluation of Decision Tree Theorem.
Objective 3	Comparison of deployed model in Sub_Objective (iii) and Sub_Objective (iv)
Objective 4	Evaluate the model in terms of accuracy, precision, recall and F Measure for Paraphrasing detection.
Objective 5	Comparison of computed results with past research work results to know if this research is making any impact and suggestion on how we can make the paraphrasing detection in Hindi on the basis of this research result in future.

Figure 1: Research Objective

1.3.2 Research Purpose

Paraphrase detection is one of the popular research domain in natural language processing these days. It has various applications like information retrieval, question answering, text summarization etc. In this research, the techniques that will be used to tackle this problem is, similarity between strings will be measured, further, it will be applied to other abstractions of a pair of input phrases. There are several techniques available for paraphrasing detection. But Hindi language still facing scarcity of research due to lack of proper dataset, efficient feature extraction tools etc. This research will use the tools WordNet Lee and Cheah (2016), N -Gram etc. for feature generation which is easily available and tested for other languages apart from Indian language which can be inferred from the literature review. This research can highly benefit to the Hindi scholars and people who are studying or working in Hindi as their main language. North part of the Indian government$'s$ official language is Hindi, so they can also use this research for their benefit. Colleges which have courses in Hindi can use this research for plagiarism detection, question answering etc.

2 State Of Art

Due to globalization, Hindi Language has gained a considerable amount of status in the world which generated the demand of paraphrasing applications in Hindi Language and which further led towards the sudden increase of paraphrasing detection research in the Hindi Language. According to a survey, Hindi Language is the the third most spoken language in the world but still it has significant amount of research available. There is a need for research on this topic. This research implies the past research available in Hindi, English and other Languages, from which Hindi researches were most helpful for this research. Many Natural Processing tasks and machine learning algorithm have been performed on paraphrase detection in the Hindi Language. In the section STATE OF ART, past research work is critically evaluated and concise comparison among the research has been provided along with the providing required review on the past research.

One research was done in several Indian languages such as Hindi, Punjabi, Tamil, and Malayalam. In this research, convolution neural network and recursive neural network technique were employed on the DPIL@FIRE2016 dataset. The performance of these four Indian languages was compared with the performance of the English language using the same techniques. After comparison, it was observed that RNN performed better for the English language than it performed for the Hindi language because the Dataset of English was twice bigger in size than Hindi. The comparison of Indian Languages also differed greatly. RNN performed better for Hindi and Punjabi than Tamil and Malayalam. The major reason for this difference is that semantically Tamil and Malayalam are more complex than Hindi and Punjabi. The reason behind CNN performing better for English as compared to all four Indian languages is lack of presence of word2vec tool for Indian Languages Bhargava et al. (2017).

Another research was done on Indian languages in 2018 by Sarkar (2016) as similar to Bhargava et al. (2017) on Hindi, Punjabi, Tamil, and Malayalam, but in this research, the algorithms used were multinomial logistic regression and support vector machine. The dataset used in this research was same as previous DPIL@FIRE2016. The difference here from previous research is that these four Indian languages were not compared to the English language. The algorithms were trained with the semantic and lexical level similarities

of two sentences. Researcher participated in the shared task of detecting paraphrases in Indian languages organized by the forum for information retrieval and evaluation. The subtask was again divided into Task1 and Task2. Sarkar participated in both and used a system with multinomial logistic regression for this research. Out of this conference another algorithm support vector machine was tested by Sarkar for the same research and the result was compared. It was concluded that both algorithms performed better for Task1 as compared to Task2.

Another participant of DPIL@FIRE2016 researched four Indian languages Hindi, Punjabi, Tamil, and Malayalam. In this research Tian et al. (2018) compared four machine learning algorithms such as decision tree, K nearest neighbor, gradient boosting and support vector machine performance for paraphrasing detection and also used the lexical feature with n-gram as a classification feature. Comparison of classification method taken out and features were analyzed for determining which feature is most important in paraphrasing detection. For evaluating the Task corpus of 2016 DPIL@FIRE2016 were used. This research showed that gradient boosting performed better than the other three algorithms. Malayalam and Tamil obtained Higher F1 Score in Task1 and Task2 whereas Punjabi obtained a higher F1 score in Task2 only.

In 2014 El-Alfy et al. (2015) proposed the implementation of weak textual metrics based on adaptive learning technique for paraphrasing in the English language as their novel approach. In this research, corpora were used for the evaluation and extraction of similarity metrics. The completion of research has been done at three levels. At the first level performance of week textual metrics was boosted by modeling adaptive learning. At the second level performance was improved by combining network method and fusing multiple decisions of the simple network into it. The inheriting capability of adaptive learning was used as a benefit in paraphrase detection because it chooses smaller and more relevant subsets automatically. According to this research adaptive learning is capable of providing a greater or comparable outcome for paraphrasing detection as compared to other machine learning algorithms.

Comparison of three specific algorithms such as support vector machine, random forest and Gaussian Nave Bayes for the Hindi language training dataset which was done by Saini and Verma (2018) and further divided into two subtasks for achieving paraphrase detection. In cross-validation, the random forest outperformed the other two algorithms with an F1 score of 94%. Furthermore, features were used, and best classifiers were introduced in this research to extend the work and increase the performance rate which improved the performance rate by 1%. In this research, random forest performed best by getting highest accuracy and F1 measure as compared to others two and further research recommend the use of this research in various paraphrase detection application.

Burrows et al. (2013) in 2013 introduced a new concept of paraphrasing detection which focuses on two aspects: acquisition via crowdsourcing and acquisition of passage level sample. Mainly two challenges were faced with acquisition via crowdsourcing, one is that the effectiveness of paradigm depends on quality assurance and other is that without crowdsourcing testing corpora is insanely expensive. The latter challenge suggests towards the lack of researches performed and evaluated on passage-level paraphrases or shorter. The information present in these short sentences sample analysis is limited for detecting paraphrase application. This experiment infers that K nearest neighbor classifies paraphrase and non-paraphrased sentences 95% accurately at 52%recall.

The samasa which also means a compound of Hindi was discussed by Redkar et al. (2016) which can be helpful for paraphrasing detection in Hindi. Samasa uses few words to de-

scribe meaning which is one of the most important of any language. Samasa-Karta is a tool which uses the semi-automated technique to improve the richness and coverage of the language. Samasa-Karta uses IndoWordnet to extract words and create samasa or compound of words. Samasa-Karta formed compounds by passing the rule-based system through various grammar rules.

Paraphrasing detection in the Russian language was researched by Loukachevitch et al. (2017) in 2017 which tested a few methods and features. This research experimented on the four features such as part-of-speech feature, string-based feature, thesaurus based feature, and information retrieval features. For experimenting on these four features three machine learning algorithms were proposed such as Support vector machine, gradient boosting and random forest. Out of all four algorithms, random forest performed best in the tuning of all four features and parameters whereas gradient boosting performance was worst with parameter tuning. There is another research on paraphrasing detection for the Russian language by Loukachevitch et al. (2018) in which contribution of semantic features were discussed for the purpose of Russian language paraphrasing detection. In this research, multiple semantic similarities which were successful for English were applied in this.

Based on the fact that the two sentences comparison required at multiple level granularity for paraphrasing detection Yin and Schtze (2015) proposed a deep learning technique BI-Convolution-Network-MI. The level of computing interaction with feature matrices and multi-granular sentences were represented by a convolution neural network. For modeling the paraphrase detection model, all parameters like embedding and classification were directly optimized in a convolution neural network. The network was pertained due to insufficient training data for purpose of modeling

Neural LSTM network approach was proposed for paraphrasing detection by Prakash et al. (2016). LSTM is an extended version of sequence learning which is successfully deployed for multiple NLP tasks. Traditional paraphrase generator uses either handwritten rules or thesaurus-based alignment or statistical machine learning techniques. In this research, it was discussed that stacking a residual of LSTM network layers is also a useful approach for paraphrasing generation, but this technique also has a drawback, it does not require the substitution of every single word in a source sequence and thats why for machine translation it may not be sufficient enough. This experiment includes three different huge datasets and performance was shown by automated evaluation metrics.

RAE (Recursive Autoencoder) were introduced by Socher et al. (2011) in 2011for paraphrasing detection because it was known for producing high accuracy and RAE required syntactic and semantic deep analysis of two sentences. RAE is one of the unsupervised methods that is based on unfolding learning and feature vector in the syntactic tree. These features measure the similarity between words and phrases. The size of the matrix obtained from this method changes with the variation of the length of sentences. A layer of dynamic pooling is used for computing the variable size of matrices from fixed sized matrices. This method performed best as compared to other MSRP paraphrases corpus method.

Shi and Dong (2018) presented a study comparing 17 Chinese speaking graduate students as their first language. The participants in this research were from both Chinese and an American university. Out of the 17, 12 were from Chinese University and 5 were from American University. When they were asked to check for paraphrasing in their research, they found 117 paraphrases, out of which, 66 were in Chinese and 51 in English. Chinese paraphrases contained more direct sentences which were mostly translated, and English

were paraphrased. English paraphrases had 24.96% feature selection information while Chinese paraphrases had 22.49%.
Predict argument tuple were introduced by Nguyen-Son et al. (2015) in 2015 for paraphrasing detection for similar word and identical phrases. This approach generally has a high recall, but the accuracy of this approach is generally low. A similarity matching metric was developed in this research that identifies similar words, identical phrases and minor changes in the phrases for paraphrasing detection. After combining this metric with 8 standard machine translation metrics, it achieved the highest accuracy of 77.6% in paraphrasing detection.
Another language is researched for paraphrasing detection rather than English is the Turkish language by Eyecioglu and Keller (2018). In this paper, the paraphrase corpus is constructed for the Turkish language. This corpus is constructed on the basis of human judgment and includes semantic similarity scores of a pair of two sentences and it permitted to experiment with both semantic similarity and paraphrase detection.
Prayogo et al. (2018) proposed a Bayesian network for paraphrase identification in Indonesian language focusing on classifying whether two sentences are paraphrased or not. This research methodology also includes preprocessing, semantic and syntactic feature extraction, classifier building and performance evaluation. The accuracy achieved in this experiment is 75.6% with recall at 76.5%, F1 measure at 75.8% and precision is 75.2% which conclude that Bayesian network is overall a suitable technique for paraphrasing detection.

2.1 Outline of Our Contribution

After critically evaluating the past researches, the primary aim was to request the dataset to work on, due to unavailability and lack of Hindi text dataset for research. Inspired from Bhargava et al. (2017), Saini and Verma (2018), Sarkar (2016), Kulkarni et al. (2012), Bhargava et al. (2016) we have requested the same dataset from DPIL@FIRE2016 (detecting paraphrases in Indian languages organized by FIRE Forum for Information Retrieval Evaluation in 2016) for this research purpose. In addition, the past researches reviewed in the previous section helped in choosing a better model and feature extraction for this research. Research objective defined in Figure 1. is achieved with the help of an investigation of past researches. According to the research of Prayogo et al. (2018) Bayesian network performed better than other textual algorithms for paraphrasing detection in Hindi language and provided the knowledge and results of Bayesian Network for textual dataset is better. Whereas Feature extraction explained in Saini and Verma (2018) achieved a better result while training the model for Hindi dataset from which few are used in this research. Hence, based on these methods KNN And Decision Tree Algorithm are used in deploying this project for achieving better results. The author El-Alfy et al. (2015) gave a firm overview of use of adaptive learning which was considered in this research. The performance was evaluated in this research by using performance measures such as Accuracy, Precision, Recall and F-Score on the basis of researches done by Saini and Verma (2018) , Bhargava et al. (2017) , Sarkar (2016) and Bhargava et al. (2016) . After careful consideration of all valuable points from literature review positive and valuable concepts were implemented. The next section of this research report consists of architectural design and scientific methodology approach.

3 Scientific Methodology and Architectural Design

3.1 Introduction

This Section consists process flow of this research and explanation of traditional methodology and modified methodology and also the architectural design of the research.

3.2 Scientific Methodology Approach

The methodology approach used for this research is Cross Industry Process For Data Mining also referred as CRISP-DM. CRISP-DM is one of the best approaches suggested by professionals. This research is structured on CRISP-DM Methodology. CRISP-DM has six stages breakdown in which procedure is structured hierarchically Nadali et al. (2011). The best feature of the CRISP-DM approach is that it can be modified according to the requirement of research or application. For the requirement of this research generalized CRISP-DM is modified accordingly Azevedo (2008). Below is the diagram of basic CRISP-DM and modified CRISP-DM according to this research in Figure 2

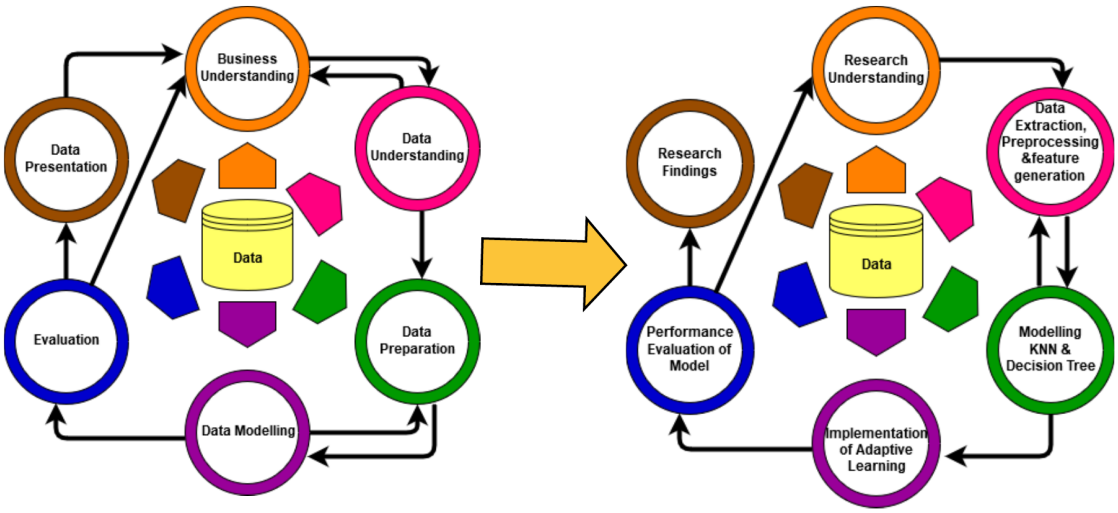

Figure 2: Scientific Methodology Used

primary change in business understanding according to the research understanding of this project is that the second stage of CRISP-DM is modified to data extraction, data cleaning and data preparation. The third stage is modified according to the model deployed in this project i.e. modeling K-Nearest Neighbor and Decision Tree. The fourth stage modified is the implementation of Adaptive learning after that fifth stage is modified to evaluation and results. The sixth stage and the last stage is modified to infer research findings of this project.

3.3 Architectural Design

For developing a robust, secure and better project for paraphrasing detection in Hindi Language, the architectural diagram is created for better accuracy, result and performance. Architecture diagram depicts the technology, techniques and tools used for this pro-

ject, how they are used and the flow of this project, for achieving the goal of paraphrase detection with better accuracy. This architecture follows the CRISP-DM methodology. Below is the depiction of the architecture followed in this project.

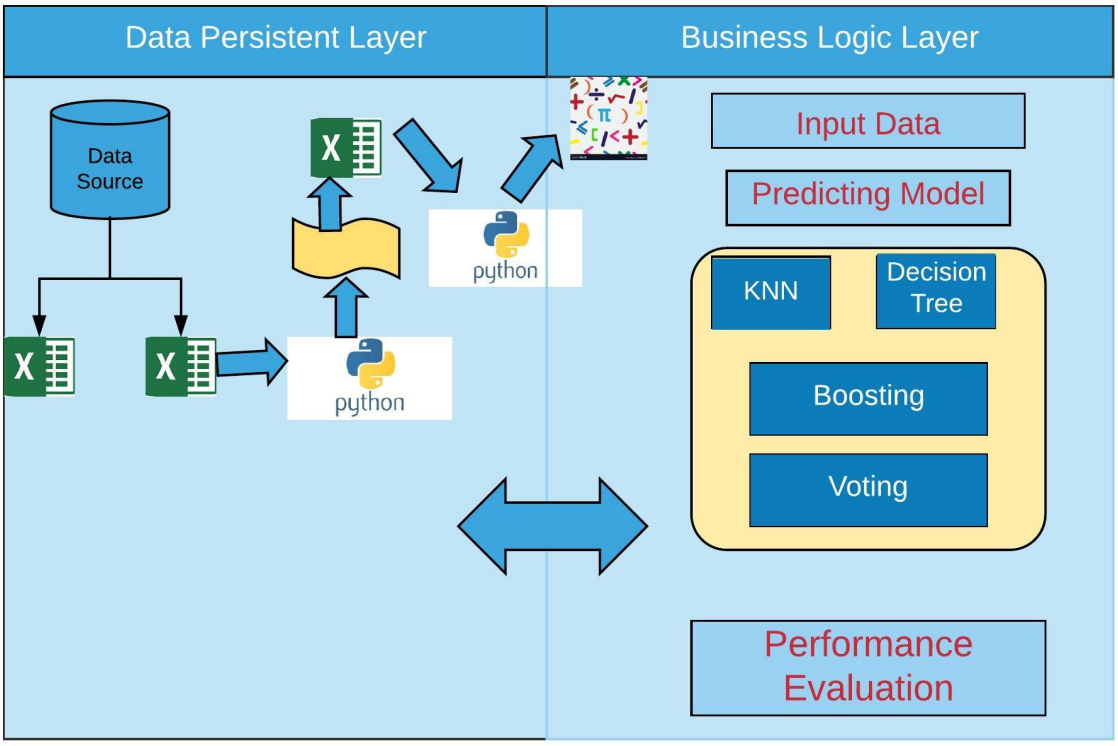

Figure 3: Scientific Methodology Used

This architecture diagram is following the the two layer structure i.e. Data Persistence Layer and Business Logic Layer. Data persistence Layer consists of Data source, Data extraction, Data cleaning, Data preprocessing and Data feature extraction. Another Layer, Business Logic Layer is about the implementation of machine learning techniques and evaluation of the results.

3.4 Outline of Scientific Methodology and Architectural Design

Therefore, this project is following the CRISP-DM approach as a methodology as per its best fit for this project implementation. Also, the architecture diagram of the project process flow is shown in Figure 3. It shows how this project has been developed. Additionally, data preprocessing and implementation of the model is elaborated in detail.

4 Data Preparation

4.1 Introduction

In this chapter, an explanation is provided about Data extraction, data cleaning, Data preprocess and data feature extraction for this project to implement machine learning techniques and evaluate the performance of models deployed. For deploying this project,

Python coding language has been used with the help of Jupyter Notebook in Anaconda environment. Reasons behind choosing Jupyter Notebook under Anaconda environment are as follows:

- Anaconda is an efficient application which easily provides all required packages and other tools within it.
- Anaconda provides package management which means it provides freedom to add and modify packages as per our requirement.
- Jupyter Notebook is the most efficient editor for python because it provides the functionality of writing code and checking the output of each single line at the same time. Also notebook has visualizations and in addition, it is easy to read and write the code.

4.2 Hindi Text Data Extraction

For starting a Machine Learning project, there is a need of dataset. For deploying this project successfully dataset has been requested from Amrita-CEN_DPIL@FIRE2016-Hindi. DPIL@FIRE2016 is detecting paraphrases in Indian languages organized by FIRE Forum for Information Retrieval Evaluation in 2016, in which dataset was available for Indian Languages such as Hindi, Marathi, Malayalam and Tamil. From that only, Hindi Language dataset was extracted for the research purpose of this project. Dataset contains three columns, from which two contains Hindi sentences and the third column is indicating if it is paraphrased or not.

4.3 Data Preprocessing

4.3.1 Text preprocessing

Text preprocessing is processed in multiple steps like importing text by encoding it, tokenization, stop words removal, Stemming, Lemmatization, handling synonyms and n-grams. Below is the process flow diagram for text preprocessing.

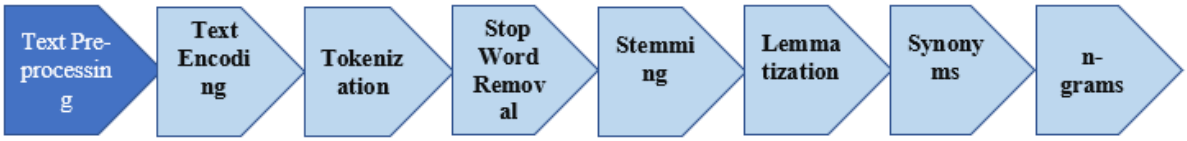

Figure 4: Process Flow for Text Pre-processing

Further explanation for text processing techniques implemented in this project is as follows:

- **Text Encoding:** In this process dataset is imported in Jupyter Notebook and after that UTF8 encoding which handle Hindi language scripting is used for encoding sentences.
- **Tokenization:** Tokenization is a process which tokenized sentences into words which in simple words means dividing the sentences into individual sequence of words which are termed as tokens. For performing tokenization NLTK library was imported in python.

- **Stop Word Removal:** There is a list of stop words for Hindi which contains 164 stop words, used in this project for removing stop words from sentences in dataset Yao and Ze-wen (2011). For e.g.

हैं, अंदर, अत, अदि, आप, इंहिं, इंहें|

Further explanation for text processing techniques implemented in this project is as follows:

- **Stemming:** In this project for stemming words into its basic structure needed a set of customized rules. Tokens of Hindi sentences are normalized in this process by removing Hindi suffixes character. For example:

'ौ','ै','ा','ी','ू','ो','े','ृ','ि','ु','ं','ॅ','कर','ाओ','िए','ाई','ाए','ने','नी','ना','ते','ीं','ती','ता','ाॅ','ां',' ोंं','ेंं','ाकर','ाइए','ाईं','ाया','ेगी','ेगा','ोगी','ोगे','ाने','ाना','ाते','ाती','ाता','तीं','ाओं','ाएं','ुओं' ,'ुएं','ुआं|

- **Lemmatization:** Further, in text preprocessing, Lemmatization of tokens are performed, which means taking the Hindi words and reducing it to its dictionary form.

- **Synonyms:** For obtaining synonyms of Hindi words, Hindi word Net has been used which is a lexical dictionary of Hindi words. This dictionary has 40K Synsets and it was developed at Centre of Indian Language Technology, Computer Science and Engineering Department in Indian Institute of Bombay by researchers. It was downloaded from the mentioned link in[1].

- **N-Grams:** N-gram is used for extracting features as n number of consecutive words in a sequence Deléger et al. (2013). In this project the value for N has been considered as 2 i.e. N = 2, which is also known as bigrams. Bigrams are used because the structure of the sentences in the dataset is small.

Steps in text preprocessing is concluded in below table:

4.3.2 Feature generation

Feature generation from the text is most challenging and tedious work in natural language processing and text analytics Seethamol and Manju (2018). There are several features which are difficult to extract from sentences and in case of Hindi text it has its own set of difficulties. Due to lack of proper packages and syntactic and semantic resources, it becomes more difficult to extract features from sentences. For the research purpose of this project, various specific features are extracted from sentences for making the machine understand the lexical and syntactic behavior of the sentence. Features extracted from the sentences after preprocessing in form of vectors, which are further used as input to pass through classifiers are further explained in detail in below section.

[1]http://www.cfilt.iitb.ac.in/wordnet/webhwn/downloaderInfo.php

Preprocessing	Input	Preprocessed
Tokenization	जानकारी के मुताबिक	'जानकारी', 'के', 'मुताबिक'
Stop words	जानकारी के मुताबिक	'जानकारी', 'मुताबिक'
Stemming	जंगलों	जंगल
Lemmatization	आरोपियों	आरोप
Synonyms	जानकारी के मुताबिक	ज्ञान अनुसार
N-Grams	जानकारी के मुताबिक	[जानकारी के] [के मुताबिक]

Figure 5: Text Preprocessing

Figure 6: Feature Generation process flow

- **Cosine Similarity:** The cosine similarity is computed between 2 vectors that are produced by using binary vectors of the phrases denoting the existence or maybe absence of the term. Cosine Similarity is calculated by multiplication of the vectors and then divided by the modulus of both the sentence vectors. It is denoted mathematically below:

$$similarity = \cos\theta = \frac{A.B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Higher the value more is going to be the similarity score for two phrases.

- **Word2Vec:** Word2Vec is a vector representation of words and it is a two-layer neural network which processes text. It has input as a text corpus and output as a set of vectors. Word2Vec helps neural deep nets to understand words by transforming it into numerical form.

- **Common IDF Score:** Numeric similarity vector is calculated by taking the sum of IDF score which was generated from common tokens of two sentences.

$$\Sigma \text{IDF score (common tokens)}$$

IDF is described as the inverse of document frequency. Inverse Document Frequency is used for identifying the importance of tokens in the corpus used in this project which is exemplified as:

$$idf_t = log \frac{N}{df_t}$$

In this project, document is referred to individual sentences. The first step in this feature generation is to create IDF for all tokens and keep it as a reference and then IDF Score of common tokens has been used as a feature for capturing the value of rarely occurring term.

- **Normalized IDF Score:** In this section, for normalizing IDF score, some part of IDF value of common tokens of two sentences is calculated for using it as a vector. In this way, we got a normalized score of IDF which ranges from 0 to 1 and it can be calculated from below formula.

$$\frac{\text{IDF of Common tokens}}{\text{Total IDF of Unique tokens (sentence 1, sentence 2)}}$$

- **Common N-Gram:** N-Gram is calculated in the text preprocessing step by taking the value of N = 2 (bigram) which was further used in feature generation by taking the common bigrams of two sentences. Further, it is used to obtain the significance of n-consecutive words simultaneously. It can be represented as follows:

- **Normalized Common N-Grams:** Similar to Normalized IDF score, Common N-Grams tokens is also normalized which were generated in the previous feature generation section. Like IDF Score it also ranges from 0 to 1 and it can be represented as follows:

- **POS Tagging:** It is a software which is used for assigning part of speech to each token in sentences. In this project, it is taking each token from Hindi sentence and assigning it part of speech.Zhang (2015)

4.4 Outline of Data Preparation

In this section, data is preprocessed in two steps i.e. text preprocessing and feature extraction. Feature extracted in this section is used further to train models in the next section.

$$\text{Bigrams (sentence 1)} \cap \text{Bigrams (sentence 2)}$$

$$\frac{\text{Common Bigrams}}{\text{Unique Tokens (sentence 1, sentence 2)}}$$

5 Implementation

5.1 Introduction

After preprocessing of data, evaluations are conducted using feature generated in section 4.3.2. Data generated in feature generation are used further to feed as a training data in the classifier to train the models using Python Scikit library.

5.2 Modeling of Pre-Processed data

In this project, three popular machine learning algorithms is used such as K-Nearest Neighbor, Decision Tree and Adaboost and further their respective performance is compared. The models used for this research purpose is a novel approach as per my knowledge, it has been never used before for paraphrasing detection in the Hindi language. Since the Adaboost is an ensemble learning algorithm, it is used for boosting the performance for both weak classifier K-Nearest Neighbor and Decision Tree. The Adaboost is implemented by training the model on the weak classifier and then the prediction of a weak classifier is boosted using Adaboost classifier and in last voted for the final decision by majority rule. For enhancing the performance of the model parameter tuning is also performed. Key parameters are as follows:

- **n_estimators:** n_estimator is used to specify the number of trees needed to build before voting for the final decision. Usually, performance is better if the number of trees is high, but it also increases the time complexity.
- **max_depth:** max_depth is the depth parameter of the tree which required to be tuned to enhance the performance of the model.
- **Learning rate:** Learning rate is defined as a decreasing function of time which need to be specified correctly. This parameter is tuned to control the amount of weight adjustment of our model with respect to loss gradient. In this project, Learning rate is set to 1 because low learning rate is better.

Best set of hyperparameters were selected for Adaboost using Grid search from Python Scikit library which leads to in following values: n_estimator = 400, max_depth = 2 and learning rate = 1. And trained the models on training data. Overall time took for training the model is less than 1 second with having RAM of 8 GB on quad-core processor machine.

- **Accuracy:** It is the most intuitive performance measure and it is calculated as the ratio of true predicted values over total values. The mathematical representation of Accuracy is shown below:

$$Accuracy = \frac{Number\ of\ correct\ instances}{Total\ number\ of\ instances}$$

- **Precision:** Precision is the ratio of true positive predicted observation to the sum of true positive observations and false positive observations. From this measure, we can articulate that the from labeled Paraphrase sentences ho many are actually paraphrased. This can be represented as follows:

$$Precision_P = \frac{Number\ of\ correct\ paraphrases}{Number\ of\ detected\ paraphrases}$$

- **Recall:** Recall measure is also known as sensitivity which is calculated as the ratio of true positive observations to the positive observations in actual class. It is represented as follows:

$$Recall_P = \frac{Number\ of\ correct\ paraphrases}{Number\ of\ reference\ paraphrases}$$

- **F-1 Score:** It is the weighted average of recall and Precision. Thus, the F-1 score considers both false positive observations and false negative observations. F-1 Score is generally considered more useful than accuracy especially in the case of uneven data.

  In all these mathematical representations of performance, measures include a notation p which is referred for paraphrasing class. Similarly, Accuracy, Recall, Precision and F-1 score can be measured for non-paraphrased class. Also, parameter macro average is used for the evaluation of performance measures, which is shown in below mathematical representation for all performance measures used in this project to evaluate our model Cordeiro et al. (2007).

  Macro-P and MacroRe is for macro Precision and macro Recall from which macro F-1 Score is computed.

$$F1 - score_P = \frac{2 \times Precision_p \times Recall_p}{Precision_p + Recall_p}$$

$$Macro - P = \frac{Precision_P + Precision_{NP} + Precision_{SP}}{Number\ of\ classes}$$

$$Macro - Re = \frac{Recall_P + Recall_{NP} + Recall_{SP}}{Number\ of\ classes}$$

$$Macro - F1\ score = \frac{2 \times Macro - P \times Macro - R}{Macro - P + Macro - R}$$

5.3 Deployment

This research does not have any real deployment yet. This Research will be deployed for presenting masters thesis only by presenting the report and performance evaluation achieved.

5.4 Design Models Used for Research

This section includes the detailed discussion about models used in this research.

5.4.1 K-Nearest Neighbor

K-Nearest Neighbor is used for both regression and classification problems, but it is mostly used for classification problems in business. KNN is usually used because it is easy to interpret the results and it takes less execution time Tan (2005). KNN does not make any assumption on the basis of data distribution. In this project, data is classified into two classes paraphrased and non-paraphrased and the parameter K has been set up to 5. In this algorithm, text is classified as paraphrased or non-paraphrased by majority voting of its neighbor.

The working algorithm of KNN for this research project is as follow:

1. Loading the Hindi text dataset
2. Initialized the value of K = 5
3. For obtaining the class which is getting predicted, iterated from 1 to end the number of data points in training dataset:
    - The computed distance of test dataset with each training dataset row. For computing distance Euclidean distance formula is used.
    - Sorted the computed distance in increasing order on the basis of distance values.
    - From the sorted array top k rows are obtained.

- From these rows, the most common class is obtained.
- Return the predicted class.

5.4.2 Decision Tree

A decision tree is always drawn upside down, it has roots on the top with branching downside. It is one of the most popular and power classification tools. It has an internal node which describes conditions, edges which slits the tree as branches on the basis of conditions and in last it has a leaf which represents the decision as per condition. Decision Tree makes it easier to represent the relationship between data attributes Apté et al. (1994). It is also known as CART (Classification And Regression Tree). The working algorithm of Decision Tree for this research project is as follow:

1. Computed entropy for Hindi dataset.
2. For every feature:
   - Computed entropy for each categorical data.
   - Took common token entropy for the current feature.
   - The computed gain for current feature.
3. Picked the highest gain feature.
4. Repeat until we reach a desired tree.

5.4.3 Adaptive Boosting

Adaptive boosting is also referred to as Adaboost in short and it is the first proposed boosting algorithm in 1996 by Freund and Schapire. It focuses on converting a weak set of classification problem into one strong algorithm Jeong et al. (2009) . The mathematical representation for Adaboost is as follow:

$$F(x) = sign(\sum_{m=1}^{M} \theta_m f_m(x)),$$

5.5 Outline of Implementation

Now models are implemented in 5, on the data with the help of Python in Jupyter Notebook. The performance of is evaluated in the next Section 6.

6 Result

6.1 Introduction

This section is about discussing the chosen performance evaluation metrics respective to the algorithm and further algorithms are also compared on the basis of performance evaluation metrics.

6.2 Performance Evaluation of Models

In this project, data is divided into training and testing set to compute the overall accuracy of the system. This research project is evaluated on four performance measure metrics which are Accuracy, Precision, recall and F-1 Score. The evaluation parameter value is computed for both respective algorithm K-Nearest Neighbors and decision tree and then the Adaboost algorithm is applied to both algorithm for boosting the evaluation parameter values. For our dataset, we have got the overall accuracy of 87.41% with F-1 Score 87.58% maximum for K-nearest neighbor after blending with the Adaptive Boosting and by following the proposed approach used in this project. With following the same approach and after blending the decision Tree classifier into Adaptive Boosting we got the accuracy of 87.03% and f1-score of 87.02%. From the above mentioned experimental results, we can see that there is a significant difference in the performance of K-nearest Neighbor and Decision after blending it with adaptive boosting. After that macro average was also computed for each performance metrics respective to their algorithm. Performance matrix summary detail can be found in Figure 7 and Figure 8.

Algorithm	Accuracy	Precision	Recall	F-1 Score
K-Nearest Neighbor	87.40%	87.32%	87.21%	87.58%
Decision Tree	87.03%	87.00%	86.99%	87.01%

Figure 7: Performance Metrics Score

Algorithm	Avg. Accuracy	Avg. Precision	Avg. Recall	Avg. F-1 Score
K-Nearest Neighbor	87.41%	87.58%	87.22%	87.32%
Decision Tree	87.04%	87.02%	86.99%	87.01%

Figure 8: Performance Metrics Average Score

Finally, Voting Classifier is used to vote the best algorithm performed by majority voting. We, used our Classifier(KNN, Decison Tree) to intialize soft voting. Voting classifier is given weight[1,1] , which means that the average for both is of similar weight for

calculating average probablity. For visualizing this probability wheightage, both classifier are fitted on the training set and graph is plotted for predicting class probablities. Green color is used to plot both classifier weightage probability bar and Blue color is used for plotting average voting probablity of both classifier. probability matrix of the votes for each algorithm. Graph is shown below in Figure 9 From array, we can conclude that decision tree have zero value for class 1. and highest value for class 2.

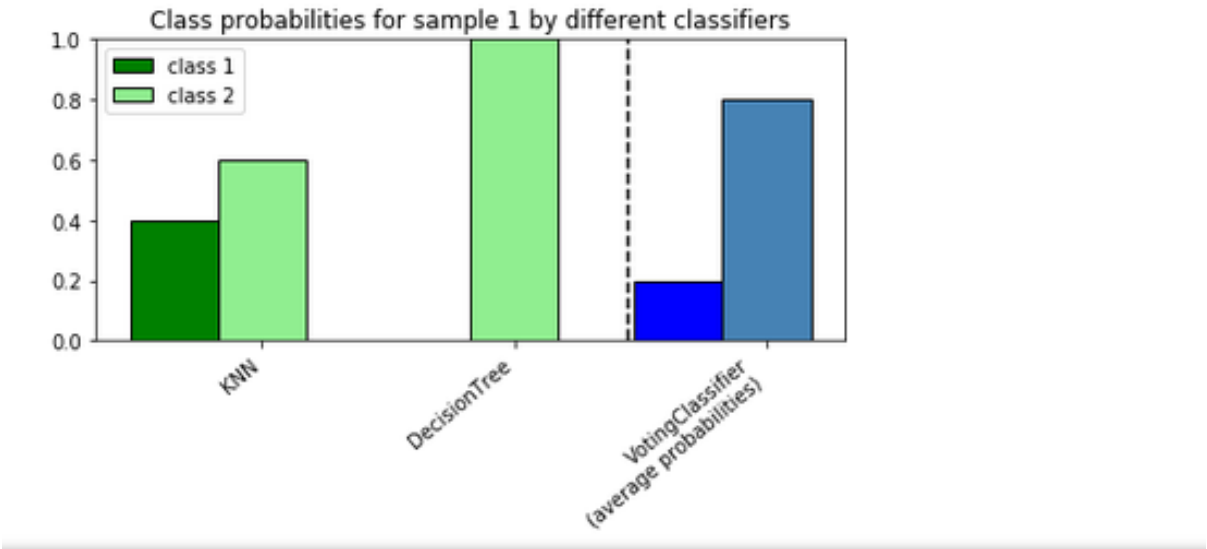

Figure 9: Class Probabilities for different classifier

6.3 Outline of Result

This section concluded that K-Nearest Neighbor performed better than Decision Tree with Adaptive Boosting on the Hindi dataset used in this project but the difference is significant. According to the Voting classifier result and the above graph referred to Figure 9, we can conclude that decision tree have zero value for class 1. and highest value for class 2.

7 Conclusion And Future Work

Looking back at the motivation of this research, the aim is to, successfully detect paraphrase in the Hindi language. So that it can help the people of the States of India which uses Hindi as an official language and will benefit Hindi language Scholars. As mentioned earlier that paraphrasing detection have many applications such as text summarization, question answering, plagiarism detection, information retrieval etc. In inclusion, this project can also contribute to knowledge, also into the data analysis industry. This research aim is to tackle the problem of paraphrasing detection in the Hindi language and it is achieved successfully by merging syntactic, lexical and semantic similarity measure and to boost the performance of a set of weak classifiers i.e. KNN and Decision Tree and combining them together for better accuracy. Accurate results can significantly help in developing uses of Hindi language in the industry by the development of applications and tools. This is the first research to perform KNN and Decision Tree blended with Adaptive Boosting and the results were pretty good but no better than the other research reviewed

in the literature survey. Our Model performed well on the Hindi dataset used for this research purpose with the accuracy of 87.41% and the f-1 score of 87.58% maximum. There is a significant difference between scores of KNN and Decision Tree. Limitation of this research is that, we faced two types of problems while processing data for paraphrasing detection which cannot be sorted with traditional similarity method. The first problem is the meaning problem, this problem comes in the lack of semantic analysis category. This problem is associated with the measuring of the similarity score of two sentences. The techniques which were used for measuring similarities such as lexical similarity, WordNet similarity and corpus-based technique only measure the semantic similarity between two words not between whole phrases. Due to which the similarity measure between two phrases is still an open question. The second problem is the word order problem, this address the problem of two same words implying a different meaning in two different sentences according to the circumstances. The remedy to this problem can improve the degree of similarity. The third problem is the availability of dataset for the Hindi language. If in the future, a large dataset and more resources for Hindi language will be available, the course of this research can be improved or completely change for the better.

Acknowledgement

I would like to convey my gratitude wholeheartedly to my thesis Supervisor and mentor, Dr. Soumyabrata Dev of the School of Computing at National College of Ireland, who always extended his knowledge whenever i needed any kind of guidance regarding my research subject. I would like to thank my parents, who were always there as a constant source of support throughout this research period. At last, I would also like to thank my friends who believed in me and were there whenever I needed any support.

References

Apté, C., Damerau, F. and Weiss, S. M. (1994). Automated learning of decision rules for text categorization, *ACM Trans. Inf. Syst.* **12**(3): 233–251.
URL: *http://doi.acm.org/10.1145/183422.183423*

Azevedo, Ana Isabel Rojo Loureno Santos, M. F. (2008). Kdd, semma and crisp-dm: a parallel overview.

Bhargava, R., Baoni, A., Jain, H. and Sharma, Y. (2016). Bits-pilani@dpil-fire2016: Paraphrase detection in hindi language using syntactic features of phrase, *CEUR Workshop Proceedings* **1737**: 239–243.

Bhargava, R., Sharma, G. and Sharma, Y. (2017). Deep paraphrase detection in indian languages, *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2017* pp. 1152–1159.

Blacoe, W. and Lapata, M. (2012). A comparison of vector-based representations for semantic composition, *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 546–556.
URL: *http://dl.acm.org/citation.cfm?id=2390948.2391011*

Brun, C. and Hagège, C. (2003). Normalization and paraphrasing using symbolic methods, *Proceedings of the Second International Workshop on Paraphrasing - Volume 16*, PARAPHRASE '03, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 41–48.
URL: *https://doi.org/10.3115/1118984.1118990*

Burrows, S., Potthast, M. and Stein, B. (2013). Paraphrase acquisition via crowdsourcing and machine learning, *Computer Communication Review* **43**(3).

Cordeiro, J., Dias, G. and Brazdil, P. (2007). A metric for paraphrase detection, *2007 International Multi-Conference on Computing in the Global Information Technology (ICCGI'07)*, pp. 7–7.

Deléger, L., Cartoni, B. and Zweigenbaum, P. (2013). *Paraphrase Detection in Monolingual Specialized/Lay Comparable Corpora*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 223–241.
URL: *https://doi.org/10.1007/978-3-642-20128-8*$_1$2

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

Eyecioglu, A. and Keller, B. (2018). Constructing a turkish corpus for paraphrase identification and semantic similarity, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **9623 LNCS**: 588–599.

Jeong, C., Cha, M. and Kim, H. (2009). Texture feature coding method for sar automatic target recognition with adaptive boosting.

Kulkarni, A., Paul, S., Kulkarni, M., Kumar, A. and Surtani, N. (2012). Semantic processing of compounds in indian languages, *24th International Conference on Computational Linguistics - Proceedings of COLING 2012: Technical Papers* pp. 1489–1502.

Lee, J. C. and Cheah, Y. (2016). Paraphrase detection using semantic relatedness based on synset shortest path in wordnet, *2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA)*, pp. 1–5.

Loukachevitch, N., Shevelev, A. and Mozharova, V. (2017). Testing features and methods in russian paraphrasing task, *Komp'juternaja Lingvistika i Intellektual'nye Tehnologii* **1**(16).

Loukachevitch, N., Shevelev, A., Mozharova, V., Dobrov, B. and Pavlov, A. (2018). Ruthes thesaurus in detecting russian paraphrases, *Communications in Computer and Information Science* **789**: 242–256.

Madnani, N. and Dorr, B. (2013). Generating targeted paraphrases for improved translation, *Computer Communication Review* **43**(3).

Nadali, A., Kakhky, E. N. and Nosratabadi, H. E. (2011). Evaluating the success level of data mining projects based on crisp-dm methodology by a fuzzy expert system, *2011 3rd International Conference on Electronics Computer Technology*, Vol. 6, pp. 161–165.

Nguyen-Son, H.-Q., Miyao, Y. and Echizen, I. (2015). Paraphrase detection based on identical phrase and similarword matching, pp. 504–512.

Prakash, A., Hasan, S. A., Lee, K., Datla, V. V., Qadir, A., Liu, J. and Farri, O. (2016). Neural paraphrase generation with stacked residual LSTM networks, *CoRR* **abs/1610.03098**.

Prayogo, A., Mubarok, M. and Adiwijaya (2018). On the structure of bayesian network for indonesian text document paraphrase identification, Vol. 971.

Redkar, H., Joshi, N., Singh, S., Kulkarni, I., Kulkarni, M. and Bhattacharyya, P. (2016). Samasa-karta: An online tool for producing compound words using indowordnet, *Proceedings of the 8th Global WordNet Conference, GWC 2016* pp. 322–329.

Saini, A. and Verma, A. (2018). Anuj@dpil-fire2016: A novel paraphrase detection method in hindi language using machine learning, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **10478 LNCS**: 141–152.

Sarkar, K. (2016). Ks-ju@dpil-fire2016: Detecting paraphrases in indian languages using multinomial logistic regression model, *CEUR Workshop Proceedings* **1737**: 250–255.

Seethamol, S. and Manju, K. (2018). Paraphrase identification using textual entailment recognition, Vol. 2018-January, pp. 1071–1074.

Shi, L. and Dong, Y. (2018). Chinese graduate students paraphrasing in english and chinese contexts, *Journal of English for Academic Purposes* **34**: 46–56.

Socher, R., Bengio, Y. and Manning, C. D. (2012). Deep learning for nlp (without magic), *Tutorial Abstracts of ACL 2012*, ACL '12, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 5–5.
URL: *http://dl.acm.org/citation.cfm?id=2390500.2390505*

Socher, R., Huang, E., Pennington, J., Ng, A. and Manning, C. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011* .

Tan, S. (2005). Neighbor-weighted k-nearest neighbor for unbalanced text corpus, *Expert Systems with Applications* **28**(4): 667 – 671.
URL: *http://www.sciencedirect.com/science/article/pii/S0957417404001708*

Tian, L., Ning, H., Kong, L., Chen, K., Qi, H. and Han, Z. (2018). Sentence paraphrase detection using classification models, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **10478 LNCS**: 166–181.

Wu, Y., Zhang, S., Zhang, Y., Bengio, Y. and Salakhutdinov, R. R. (2016). On multiplicative integration with recurrent neural networks, *in* D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon and R. Garnett (eds), *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pp. 2856–2864.
URL: *http://papers.nips.cc/paper/6215-on-multiplicative-integration-with-recurrent-neural-networks.pdf*

Yao, Z. and Ze-wen, C. (2011). Research on the construction and filter method of stop-word list in text preprocessing, *2011 Fourth International Conference on Intelligent Computation Technology and Automation*, Vol. 1, pp. 217–221.

Yin, W. and Schtze, H. (2015). Convolutional neural network for paraphrase identification.

Zhang, Yuan Li, C. B. R. D. K. (2015). Randomized greedy inference for joint segmentation, pos tagging and dependency parsing, pp. 42–52.
URL: *http://www.aclweb.org/anthology/N15-1005*