Creation of Dataset, Pre-processing and Extraction of Features for Paraphrase Detection in a Low Resourced Language : Kannada

Anagha H M
Computer Science and Engineering
PES University
Bangalore,India
anu.hm0520@gmail.com

Karthik Sairam
Computer Science and Engineering
PES University
Bangalore,India
karthik.sairam2001@gmail.com

***Abstract*—Paraphrase detection is an Natural Language Processing classification problem. The main goal of paraphrase identification is to find if two phrases of different lengths are synonymous. This is done by finding the similarity between the given two paraphrases. This paper attempts to preprocess and extract the necessary features required to perform paraphrase detection in Kannada. This is the first few steps done before proceeding to paraphrase identification.**

Kannada is a South-Indian language and it is written using the Kannada script. Kannada is originated from the Dravidian Language.

***Index Terms*—NLP, tokenization, paraphrases, Kannada, pre-processing, token, dataset, stemming**

I. Introduction

Text processing has always been an important aspect of natural language processing. Much research has been carried out in this domain pertaining to English and many European languages, leaving behind many low resourced languages. One such language is Kannada, a Dravidian language spoken by the people of Karnataka, a southern state in the country of India. Working towards developing a paraphrase detection system in Kannada requires the building of the relevant dataset and the related learning models from scratch. Of these, building the dataset, the pre-processing and the extraction of relevant features are of utmost importance and act as the basis of further developing a fully working robust model for paraphrase detection.

Paraphrase Identification is used to find whether the given to sentence are synonyms. The general method of performing this is to use NLP. To start with NLP, we need a dataset which has to be cleaned and features need to be extracted.

Dataset is a file that contains records. It contains two components- rows and columns. For paraphrase identification we need two main columns. Each column containing the sentences or phrases.

Preprocessing of the dataset is a very key step that needs to be performed to get accurate results. Preprocessing of the data includes cleaning the datasets of unnecessary symbols, numbers and making the dataset ready for the next step of processing, tokenization.

Tokenization is a fundamental step in NLP. It is the process of seperating a sentence into smaller units called tokens. Tokens are not necessarily the work=ds in the sentence. It is the smallest meaningful unit of the sentence. they can be words, subwords or even characters. Subwords refer to a group of letters that don't exactly have a meaning. Character tokens refer to a single alphabet.

Stemming is the process of reducing all the tokens available into its root word. Stemmers are language-specific. Stemming and lemmatizing go hand in hand. Lemmatizing requires a much larger vocabulary when compared to stemming.

After all the above-mentioned work is done, the features are extracted and sent to a neural network , or for a simpler approach, it is directly sent into a classifier. The above mentioned methods can be implemented irrespective of the language being processed.

II. Literature Survey

A lot of work has been done in paraphrase detection, but mainly all of them are done for English or other Indian Language. Considerable work has not been done in a low resourced language like Kannada.

We can see in paper [1], details about the Cosine Similarity, Jaccard Similarity and Levenshtein Ratio as three features to be extracted for the purpose of Language Independent paraphrase detection. The three features of the sentences are extracted and are passed into a Probabilistic Neural Network(PNN). PNNs are widely used for classification and pattern recognition problems.

Paper [2] proposed a survey on the different techniques available for the detection of paraphrases in Indian Languages. It gives us insight on all the different NLP approaches that can be used. It also talks about the different similarity metrics that can be used, including Sumo-metric, n-gram overlap and BLEU measure.

This paper [3] proposed a method which extracts six major features from the sentences and performs similarity checks based on the extracted features. It sheds some light on the different corpuses and the procedure to be followed when building a corpus.

In paper [4], we can learn about the Basic and Chunk phrased uni and bi directional methods. These were performed by extracting five features from the sentences.

Paper [5] defines paraphrases and explains the different algorithms that can be used. It also compares the different approaches and thus gives a better idea of which methods/ algorithm should be used for better results.

Tagging, pruning, stop word removing and stemming is done in paper [6]. It uses a semantic similarity algorithm approach to solve paraphrase detection in NLP.

Soundex core is used to encode homophones as the same representation. Paper [7] uses this method to improve their paraphrase detection. Error analysis and classifier comparison is done.

Tokenization, POS tagging, Token Match, Token count are some of the features extracted and fed into a classifier in paper [8]. SVM classifier is used.

An entirely different approach is taken in paper [9], the words are coverted into vectors and a matrix is created. Manhatten LSTM is the approach chosen and it's used to find sentence similarity. This also uses a Siamese Neural Network. It has a high accuracy and needs abundant data for execution.

In paper [10] topic phrases are removed, normalization is performes and boundary correction is done. This provides a Machine Learning based approach and a set of lexical, syntactic and pragmatic features. It also sets a benchmark for effectiveness on a clean corpus.

Semantic similarity measurement of English texts are performed and evaluation metrics are also given in paper [11].

Paper [12] uses lean paraphrase identification techniques for identifying paraphrases. It represents paraphrases as pairs. It uses both word and character based features.

Paper [13] uses a comparatively easier approach and uses an SVM to classify paraphrases. The inputs to the SVM here are tokens, stemmed words and stop-words.

Paper [14] has a greedy search approach, where dependencies are created for each sentences in the dependency tree. Classification is implemented in two major steps- search and learning method. It also has a fully automated dependency tree for the paraphrases by implementing 'maximise structural isomorphism'.

Words are converted into numerical vectors using approaches like Kate Autoender in paper [15]. Further, they use not one, but 5 classifiers to evaluate the results and compare them with other classifiers.

III. Proposed Methodology

The proposed methodology involves the following steps:

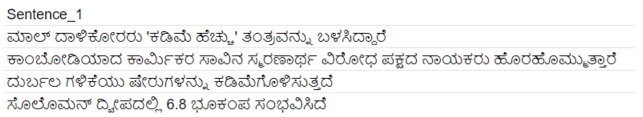
1) Creation of the Dataset
2) Cleaning of the Dataset
3) Pre-processing
4) Tokenization
5) Stemming

A. Creation of Dataset

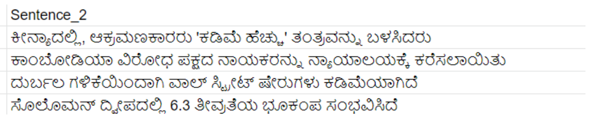
The first step includes the dataset in the Kannada language to be worked on. This is indeed challenging due to the lack of meaningful datasets in the Kannada language, and hence, a new dataset was created from scratch for this purpose of this project.
The dataset thus created has over 600 data points, and over 1200 sentences in the Kannada language. This creation constituted one of the major portion of the project process.
The data points are primarily in the Kannada language, but also seldom contain English Nouns like country names (e.g, U.S.A) and numerical data (e.g, 6%). Certain alphanumeric characters also exist, like the percentage symbol (%) and the dollar sign($).
In addition to this, the dataset thus created is structured as shown:

Sentence 1	Sentence 2
..	..

A sample of the dataset is shown below:



Sentence_1
ಮಾಲ್ ದಾಳಿಕೋರರು 'ಕಡಿಮೆ ಹೆಚ್ಚು' ತಂತ್ರವನ್ನು ಬಳಸಿದ್ದಾರೆ
ಕಾಂಬೋಡಿಯಾದ ಕಾರ್ಮಿಕರ ಸಾವಿನ ಸ್ಮರಣಾರ್ಥ ವಿರೋಧ ಪಕ್ಷದ ನಾಯಕರು ಹೊರಹೊಮ್ಮುತ್ತಾರೆ
ದುರ್ಬಲ ಗಳಿಕೆಯು ಷೇರುಗಳನ್ನು ಕಡಿಮೆಗೊಳಿಸುತ್ತದೆ
ಸೊಲೊಮನ್ ದ್ವೀಪದಲ್ಲಿ 6.8 ಭೂಕಂಪ ಸಂಭವಿಸಿದೆ



Sentence_2
ಕೀನ್ಯಾದಲ್ಲಿ, ಆಕ್ರಮಣಕಾರರು 'ಕಡಿಮೆ ಹೆಚ್ಚು' ತಂತ್ರವನ್ನು ಬಳಸಿದರು
ಕಾಂಬೋಡಿಯಾ ವಿರೋಧ ಪಕ್ಷದ ನಾಯಕರನ್ನು ನ್ಯಾಯಾಲಯಕ್ಕೆ ಕರೆಸಲಾಯಿತು
ದುರ್ಬಲ ಗಳಿಕೆಯಿಂದಾಗಿ ವಾಲ್ ಸ್ಟ್ರೀಟ್ ಷೇರುಗಳು ಕಡಿಮೆಯಾಗಿದೆ
ಸೊಲೊಮನ್ ದ್ವೀಪದಲ್ಲಿ 6.3 ತೀವ್ರತೆಯ ಭೂಕಂಪ ಸಂಭವಿಸಿದೆ

B. Cleaning of the Dataset

This involves the removal of unwanted characters and numerics from the data points in the dataset created. Given below is the list of characters classified as 'unwanted'

1) English Letters
2) Special Characters ($, %, ˆ, ., -, ,, "", ",)
3) Numerals (0-9)

Such unwanted characters are removed from all of the data points.

The dataset is then checked for inconsistencies, or NULL values. Since the dataset was handcrafted by the authors, no such NULL value exists in any data point. Post-cleaning, the sample of sentences are as shown below:

ಭಾರತೀಯ ಮಾಧ್ಯಮ ಕಾಮನ್‌ವೆಲ್ತ್ ಶೃಂಗಸಭೆ
ರಷ್ಯಾ ವಿಮಾನ ದುರಂತದಲ್ಲಿ ಸಾವಿನ ಸಂಖ್ಯೆ ಏರಿಕೆಯಾಗಿದೆ
ಮನೆ ತಂಗಿ ಹಗರಣದಲ್ಲಿ ಮಂದಿ ಬಂಧನ

The final step in completion of dataset cleaning is to export the dataset thus cleaned into a suitable format (CSV) without the loss of any data points, or part of any data point.

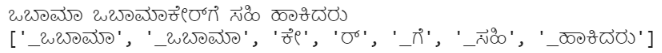
C. Tokenization

Any text processing task begins with splitting the sentences into its words, namely tokens. Tokenization is something more than just separating the words, and involves finding the smallest meaningful unit in a sentence. Such a unit is called a token.

The tokenization process is implemented using the INLTK library, which is toolkit for natural language processing in Indic Languages. The primary language of implementation of the program is in Python 3.7 and the softwares used are as follows:

1) Python 3.7
2) Pandas module
3) Numpy module
4) INLTK Library
5) Regex module

The below picture shows the cleaned sentence, and then the sentence after tokenization, for an example sentence in the Kannada language.



```
ಒಬಾಮಾ ಒಬಾಮಾಕೇರ್‌ಗೆ ಸಹಿ ಹಾಕಿದರು
['_ಒಬಾಮಾ', '_ಒಬಾಮಾ', 'ಕೇ', 'ರ್', '_ಗೆ', '_ಸಹಿ', '_ಹಾಕಿದರು']
```

D. Stemming

The next part of the process is to stem the words available in each sentence. Stemming is necessary in documents having different forms of the same root word, while the sentences in this Kannada dataset do not require stemming. This is because the probability of the tokens of the same root word appearing in the same sentence is minimal, hence negligible.

IV. Outcomes

Originally the dataset has two columns and each column has a sentence in it. This is the most time consuming step of the entire process. This is created by using an English paraphrase dataset as a reference. After following an extensive and time consuming process, the dataset was created and was ready to be cleaned and preprocessed.

After cleaning the dataset of all the extra symbols, characters, numbers and whitespaces are removed. Thus presenting us with a cleaned dataset.

This is then passed into the tokenizer. The tokenizer splits these sentences into tokens. these tokens are stored as a list.

This is then passed to a stemmer. Thus reducing all the word into its root word.

After going through the above steps, we are left with a clean, preprocessed, tokenized and stemmed dataset.

V. Future scope

After following the preceding steps we arrive at a cleaned, preprocessed dataset. This opens up endless possibilities. Some of them are :

- The main reason for doing all the procedures, is to make the dataset fit for paraphrase identification. The main future scope of this is to perform paraphrase identification/detection on the preprocessed dataset. This can be done by passing the extracted features into a nueral network. The output of the neural network can be used input for a classifier which will classify based on the input given whether the two sentences given are paraphrases or not.
- This can also be extended as a plagiarism checker. Plagiarism checkers use the same concepts of paraphrase detection. But they also check to see how much is directly copied. This can be done by using the same plagiarism checker on a bigger scale.
- With the above knowledge, this can also be extended to make a paraphrase generator. This will simply create paraphrases given a sentence. This will require extensive training of a model. This also requires a big dataset for training.

Acknowledgment

We would like to thank Prof.Mamatha H R for her continued support, engagement and her invaluable guidance throughout this experience. We would also like to thank the PES Computer Science Department for this wondrous opportunity and constant support with our research endeavors

References

[1] Sarkar, Sandip Saha, Saurav Bentham, Jereemi Pakray, Dr. Partha & Gelbukh, Alexander. (2016). NLP-NITMZDPIL-FIRE2016:Language Independent Paraphrases Detection.

[2] Srivastava, Shruti & Govilkar, Sharvari. (2017). A Survey on Paraphrase Detection Techniques for Indian Regional Languages. International Journal of Computer Applications. 163. 42-47. 10.5120/ijca2017913757.

[3] Sánchez-Vega, F., Villatoro-Tello, E., Montes-y-Gómez, M. et al. Paraphrase plagiarism identification with character-level features. Pattern Anal Applic 22, 669–681 (2019). https://doi.org/10.1007/s10044-017-0674-z

[4] Koleva, N., Andrea Horbach, Alexis Palmer, Simon Ostermann and Manfred Pinkal. "Paraphrase Detection for Short Answer Scoring." (2014).

[5] Magnolini, Simone. (2014). A survey on paraphrase recognition. CEUR Workshop Proceedings. 1334. 33-41.

[6] Kanjirangat, Vani Gupta, Deepa. (2016). ASE@DPIL-FIRE2016: Hindi Paraphrase Detection using Natural Language Processing Techniques & Semantic Similarity Computations.

[7] Bhargava, Rupal & Baoni, Anushka Jain, Harshit & Sharma, Yashvardhan. (2016). BITS_PILANIDPIL-FIRE2016:Paraphrase Detection in Hindi Language using Syntactic Features of Phrase.

[8] P. Vigneshvaran, E. Jayabalan and A. V. Kathiravan, "An Eccentric Approach for Paraphrase Detection Using Semantic Matching and Support Vector Machine," 2014 International Conference on Intelligent Computing Applications, 2014, pp. 431-434, doi: 10.1109/ICICA.2014.94.

[9] A. A. Aziz, E. C. Diamal and R. Ilyas, "Paraphrase Detection Using Manhattan's Recurrent Neural Networks and Long Short-Term Memory," 2019 6th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), 2019, pp. 432-437, doi: 10.23919/EECSI48112.2019.8976951.

[10] Kuntal Dey, Ritvik Shrivastava , Saroj Kaushik ""A Paraphrase and Semantic Similarity Detection System for User Generated Short-Text Content on Microblogs" 2016

[11] M. Alian and A. Awajan, "Paraphrasing Identification Techniques in English and Arabic Texts," 2020 11th International Conference on Information and Communication Systems (ICICS), 2020, pp. 155-160, doi: 10.1109/ICICS49469.2020.239485.

[12] "Eyecioglu A., Keller B. (2018) Knowledge-lean Paraphrase Identification Using Character-Based Features. In: Filchenkov A., Pivovarova L., Žižka J. (eds) Artificial Intelligence and Natural Language. AINL 2017. Communications in Computer and Information Science, vol 789. Springer, Cham. https://doi.org/10.1007/978-3-319-71746-3_21".

[13] D. S. Bhole and S. S. Patil, "Detection of paraphrases for Devanagari languages using support vector machine," 2018 International Conference on Communication information and Computing Technology (ICCICT), 2018, pp. 1-5, doi: 10.1109/ICCICT.2018.8325880.

[14] D. Uribe, ”Recognition of Paraphrasing Pairs,” 2008 Electronics, Robotics and Automotive Mechanics Conference (CERMA ’08), 2008, pp. 50-55, doi: 10.1109/CERMA.2008.56.

[15] H. Shahmohammadi, M. Dezfoulian and M. Mansoorizadeh, ”An Extensive Comparison of Feature Extraction Methods for Paraphrase Detection,” 2018 8th International Conference on Computer and Knowledge Engineering (ICCKE), 2018, pp. 47-51, doi: 10.1109/ICCKE.2018.8566303.