Lexi Con-Text Understanding the Context of a Sentence

Raj Kotadia
Dept. of Information Technology
Vidyavardhini's College of
University of Mumbai
Mumbai, India

Akash Rajpurohit
Dept. of Information Technology
Vidyavardhini's College of
University of Mumbai
Mumbai, India

Abhay Tank
Dept. of Information Technology
Vidyavardhini's College of
University of Mumbai
Mumbai, India

***Abstract*—Natural Language Processing is one of the evolving fields given the ever growing amount of textual data and the need to analyze it. One of the important aspects of Natural Language Processing is computing the similarity between different sentences. Sentence similarity task has wide applications ranging from AI and chat bots to Plagiarism Analysis as well as for removal of duplicate entries. Although it has many applications the different approaches that exists to compare sentences are very naive, most to these methods compute similarity based on common keywords between them. Such a method tends to falter in most cases apart from having very less accuracy. It is essential to also consider the semantics of the sentences apart from matching keywords while making the comparison. We propose an approach for comparing the sentences by considering their underlying context. The comparison is made based on syntactic as well as semantic analysis of the sentences through the use of word embeddings to give better results.**

***Index Terms*—Natural Language Processing, sentence similarity, word embeddings, vectorization**

I. Introduction

Natural Language Processing, also known as NLP, is one the advancing fields in Computer Science whose applications range over a variety of use cases. NLP falls under the category of AI which is the ability of computers to understand human language. One of the prime domains of NLP is sentence similarity which is still in its evolving phase. The application in focus for this paper is the comparison between sentences to check for similarity between them. Computers generally require us humans to "talk" to them in terms of a programming language that is unambiguous, precise and highly structured, or through clearly defined voice commands in order to carry out a certain task. However,the human speech is generally not so precise; it is often tends to be ambiguous and the linguistic structure can depend on many complex parameters. Apart from that the textual data that prevails is semi-structured in nature, a computer cannot interpret it as it is. This textual data is to be converted into a form which enables the computers to understand it and make sense out of it.

Given the above complications, the methods available are naive as they only compare sentences based on the common keywords they contain. Such methods do not take the underlying meaning into consideration while comparing the sentences. That is they only make syntactic comparisons. It might so happen that the sentences in question contain similar words but have a completely different context. To counter these problems our proposed algorithm effectively helps in finding out the similarity among various sentences by enabling computers to have a deep sense of understanding about the sentences they are dealing with. The algorithm will measure the similarity based on syntactic analysis while considering the semantics of the sentence as well. The algorithm employs a method which makes use of tokenization, parts of speech tagging and word embeddings for an effective representation of words in vector space to extract a more precise meaning out of each of the words and then make the necessary comparison.

With recent advancements in AI it is possible to develop such models which helps in making a computer understand the natural language and extract information from it. The idea is to develop such an algorithm which takes a pair of sentences as input and then computes the similarity between them to produce the output in a percentage format.

II. Related Work

Multitudinous research work has been done in the field of NLP with fair degree of implementation. In this section, we study and understand the modules which were referred for our algorithm. These practices could be utilized and combined to generate an algorithm with other approaches, methodologies and objectives. The related work covers the articles and papers with different problem statements and their solution and modules which were relevant to our algorithm or had approaches which were statistically and tactically appropriate for workflow of current algorithm.

Modular approach was implemented in [1] where sentence breakdown was performed based on three layers.

1) Shallow Layer - Tokenization.
2) Syntactic Layer - Checking the grammar
3) Semantic Layer - Checking the Semantics

- In Shallow layer lexical analysis, stop words removal and name entity recognizing is performed. Here tokens are

created and only relevant information is considered for further text processing.
- In Syntactic layer after NER is generated, relations and directed graphs are created.
- In Frame layer, all the entities and actors and their roles in the sentence are recognized. This is termed as Actor annotation and verbs are considered as their actions.

Next paper was presented by Emil Brajkovic and Daniel Vasic [2] is based on a system for evaluating answers for a Tutoring System. Yahoo Question and Answer dataset was used for their research. This research conducted was based on sets. A sets of concepts, relations was extracted from a sentences or a paragraph. This sets were then combined to generate a new set called proposition. Collection of this propositions is called Domain Knowledge. Next new sets i.e. Questions and Answers were extracted form this propositions. For measuring similarity between sequences, Levenshtein distance algorithm was used but was infirm. Approach proposed by authors is consider distances between sentence structure using constituency based parse tree and contextual information using Word2Vec representation.The constituency parse tree was nothing but tree based representation of entities and their attributes. Also they focused on a custom tree called Knowledge tree and its performance against well known algorithms.

Paper [3] is the study of word embedding in vector space. Paper was proposed by the founders of Word2Vec and is about foundations of Word2Vec tool. Here continuous vector presentations can be computed using two architectures which are proposed in paper. Machine learning models were used to get Word Embeddings. The model is trained using 1.6 billion words dataset. The generated word embeddings are then used to perform linear algebra which helps in Natural language Processing. Continuous Bag of Words and Skip Gram are the two models which are used to generate Word Vectors. The implementation of these concepts are shown in [4].

Another paper [4] present the significance of Natural Language Toolkit for the course of Computational Linguistics and for logical research in the same field. It focuses on usage of NLTK toolkit for natural language processing. NLTK is a python based library that provides multiple tools and modules such as corpus which are nothing but linguistic corpora, tools for processing strings, tagging part-of-speech, ML, parsing and chunking, semantic interpretation, estimation of probability etc. NLTK provides a huge documentation for users as well as instructors with details of each and every method or module. Also community utilizing NLTK library has a active user base.

A summary generation mechanism from multiple documents as sources using similarity between sentences [5]. This approach was taken by the authors to eliminate the redundant sentences in the context of their meaning. Since the summarization totally was dependent on graph, edges of the graph depicts how similar the sentences are. Authors differentiate between similarity algorithm Cosine Similarity and their proposed similarity algorithm where rather than considering words in global context they check the most expected sub-sequences. For similarity measurement and summarization they use this methodology to find the most subsequent pairs of words.

To gain more insight about the impact of grammar and spell checking on similarity computation, [6] present two modules, Spell Checker and Grammar Checker. Spell checker checks two things, whether the word exists or not and if miss-spelled then error detection by dictionary lookup and n-gram technique is used to find and correct errors. In grammar checking module tensed based grammar checking and correction is done. Spell correction is done on the basis of frequency of the prefix, word is suggested and considered.

Distributional semantics is a methodology for computing differences between linguistic items. [7] DSMs are used to group words based upon the company it keeps. The article is based upon Question and Answer with ranking the answers which is most relative to Question. Representation of DSMs are in vector space domain.

III. Implementation

Based on the above literature survey it becomes obvious that it is necessary for making the computer understand the context of different words so that the comparison task can then be easily carried out. The algorithm implementation revolves around preparing a dataset, preprocessing the inputs, preparing the word embeddings and then calculating the similarity.

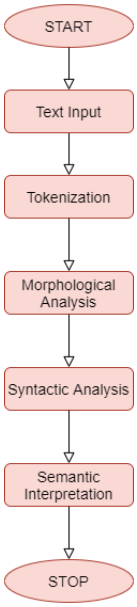

Fig. 1. Algorithm flowchart.

A. Dataset Preparation

This is the most important task which would essentially help in generating the base vocabulary upon which the word vectors can be created for the input sentences. For this purpose we used the Google News dataset that consisted of around 3 million words so that the word embeddings are generated in a better way. The model uses this data to form a basic vocabulary of words.

B. Text Input

Once the dataset is ready the next step is to collect the textual inputs for which the similarity is to be calculated and pass it on for further processing.

C. Tokenization

In order to get the individual word vectors it is necessary to tokenize the input sentences. Tokenization, in simple terms is breaking down a sentence into individual words. Many different Tokenization methods that can be utilized for this purpose. NLP [4] provides simple methods to make this task of tokenization easy.

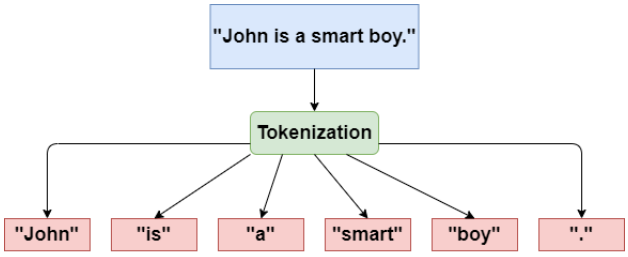

Fig. 2. Tokenization.

D. Morphological Analysis

Morphological analysis is referred as the process of obtaining grammatical information from word tokens, given their suffix information. Morphological analysis can be carried out in three ways: morpheme-based morphology (arrangement approach), lexeme-based morphology (process approach), and word-based morphology (paradigm approach). In this process we aim to find the base or dictionary form of the words which are generated in the previous step. These dictionary form of the word is also known as lemma. For example by performing morphological analysis on word ”caring” will give us the base word ”care” and not ”car”. NLTK [4] provides easy to use classes and methods that can be used for Morphological Analysis. The word is broken down into root word by recognizing the root word or stem and affixes as shown in Fig 3 .

E. Syntactic Analysis

The primary purpose of this step is to derive meaning from the text basically to understand the complete structure of the sentence in terms of Figures of Speech. Syntactic analysis may be defined as the process of analyzing the word tokens formed from the input sentences conforming to the rules of formal

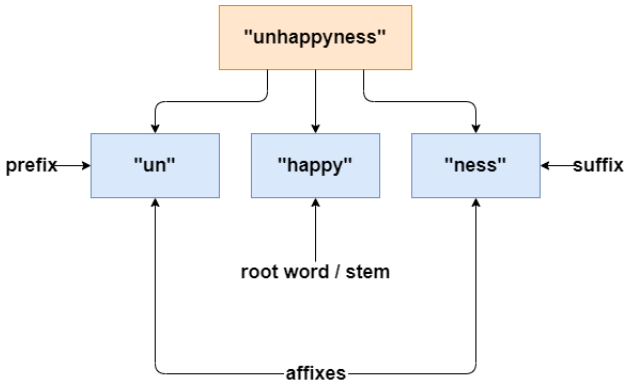

Fig. 3. Morphological Analysis.

grammar. A data structure , generally, in the form of parse tree or abstract syntax tree or other hierarchical structure.

F. Semantic Interpretation

Once the pre-processing task is completed and the tokens are all generated the next step is Semantic Interpretation which is basically deriving the context of the input sentences. It is difficult to obtain relevant information from textual input as computers do not tend to give favourable results when working with strings of data. So the better approach is to convert these strings into numerical form, specifically in to vectors, so that they can then be processed easily. This approach makes it easy to keep track of contextual information and also for further mathematical calculation.

The above mentioned steps can be generalized into the following approaches to achieve the desired tasks.

1) Word2Vec: This approach uses the steps outlined in fig. 1 but the main emphasis is given to Semantic Interpretation. There are several methods that can be employed for this task. ”One Hot Encoding” is one method that converts the individual words in the sentence into binary vectors. In this case, each word is mapped into a vector having a dimension equal to the number of words present in a sentence. Every dimension contains a ”0” except the one which represents the index of the word in the sentence. Consider the fig. 4 which provides the way of how textual data is encoded into binary vectors.

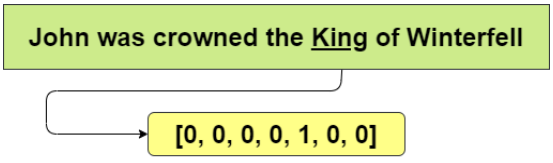

Fig. 4. One Hot Encoding.

However, this method has certain drawbacks. The size of these vectors goes on increasing with the increase in the length of the sentences. Also this method does not work well for semantics of the sentences.

A better approach is word2vec. Word2Vec is a two-layer neural network which is used to produce word embeddings. Given a blob of text, word2vec will generate a vocabulary of all the words present in the text blob and map them into vectors of some specific dimensions. Each dimension of the vector represents a certain attribute related to the word. Consider the fig. 6 which provides the example of how the word ”king” is encoded using word2vec. Here, each dimension of the vector encodes a certain quality that meaningfully represents the given word. This method helps in generating vectors from textual data which can then be easily handled by the machine and also preserves the contextual information in vector representation.

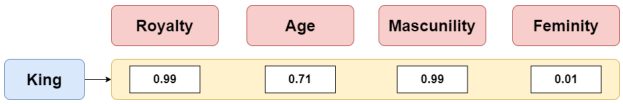

Fig. 5. Word Vector generated through Word2Vec.

These embeddings are used to represent the word in the vector space such that the words which share the similar context are located close to one another with respect to the corpus in the vector space. So words with similar meanings are close to one another in the vector space. The Word2Vec was developed by a team from Google which can trained any given dataset which acts as the corpus upon which the model generates a vocabulary of words wherein each word is mapped into its corresponding vector. For the purpose of this project, the Google News dataset was used to train the model and generate the corresponding word embeddings.

In this method, we follow the basic step of tokenization and stop words removal before feeding the input to the Word2Vec model. Once the word embeddings are generated a distance metric is used to calculate the similarity in terms of percentage. Various metrics were tried like Jaccard Similarity, Soft Inverse max but ”Cosine Similarity” provided better results. In this case, however, the stop words like ”not” also gets removed in the processing phase. In such cases the algorithm fails to produce the intended output. For example considering two sentences, ”John is a smart boy” and ”John is not a smart boy”; since ”not” is a stop word, it gets removed from the later sentence and before feeding the input to the model, both the inputs become same which is ”John smart boy”, leading to 100% similarity between them even though they are contextually different.

Later, we tried not removing the stop words. Employing this approach improved the accuracy of the algorithm. Since the aim is to analyse the contextual meaning of the sentence, stop words play a crucial part for analysis of sentimental context the inference of which can be derived from the above mentioned example.

2) Spacy: Another approach followed was by using Spacy. The reason for using Spacy was that it allowed us to predict linguistic attributes in the context such as Parts-of-Speech tags, Syntactic dependencies and Named entities. For example considering the statement ”John is intelligent boy”, the Spacy model correctly predicted ”John” as Proper noun, ”intelligent” as Adjective and ”boy” as noun.

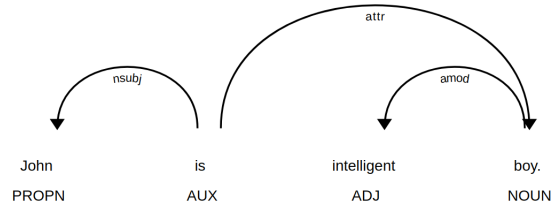

Fig. 6. Spacy.

In addition to the part-of-speech tags, it can also predict how the words are related like if a word is the subject or an object of the sentence.

Named entities are basically real world objects that are assigned a name. For example, a person, a city or any country. Consider the statement, ”Elon Musk is ready to hire developers without degree in U.S.A”. In this case, the model is correctly predicting ”Elon Musk” as person and ”U.S.A” as country.
In this approach, we convert the sentence into doc object. Doc object is nothing but collection of tokens. Tokens define the type of word, its attribute etc. Suppose we get the named entities from two sentences. First step would be to check the nouns in both sentences, if they are same, the chances are sentences are related. If nouns are same then we perform the morphological analysis of other tokens, that is getting the root word for each token. Then for each token we obtain its word embeddings and then calculate the cosine similarity to check how similar the tokens of both sentences are to each other. To compare tokens of sentences we can use syntactic dependencies as shown in fig 6 to determine the role and relation of tokens to named entity or nouns or to tokens of other sentence.

IV. Results and Discussions

The algorithm was implemented using both of the above mentioned approaches. The results differed from one another by a great margin which can be compared by the following results.

Sentence 1	Sentence 2	Similarity Percent
How can I be a good geologist?	What should I do to be a great geologist?	92.26
How I can speak English fluently?	How can I learn to speak English fluently?	97.64
What does manipulation mean?	What does manipulation means?	98.56
Why do rockets look white?	Why are rockets and boosters painted white?	89.23
How I can speak English fluently?	How can I learn speak English fluently?	97.64
What are some special cares for someone with nose that gets stuffy during the night?	How can I keep my nose from getting stuffy at night?	80.98
Should I buy car?	What keeps children active far from phone video games?	38.69

Fig. 7. Results by using Word2Vec.

As mentioned in the table above, we can see the similarity percentage that is calculated by word2vec algorithm for a set of sentences pairs. This percentage value ranges between 0 - 100% indicating how much similar are the two sentences syntactically and semantically.

Sentence 1	Sentence 2	Similarity Percent
How can I be a good geologist?	What should I do to be a great geologist?	94.58
How I can speak English fluently?	How can I learn to speak English fluently?	96.98
What does manipulation mean?	What does manipulation means?	100
Why do rockets look white?	Why are rockets and boosters painted white?	83.40
How I can speak English fluently?	How can I learn speak English fluently?	96.98
What are some special cares for someone with nose that gets stuffy during the night?	How can I keep my nose from getting stuffy at night?	92.86
Should I buy car?	What keeps children active far from phone video games?	52.29

Fig. 8. Results by using Spacy.

Both approaches are relatively accurate, however accuracy can be determined by adjusting the threshold values for both implementations. however, Word2vec provides relatively better results than Spacy to calculate the similarity having being trained on a larger corpus of text. The results generated work well with most of the cases leaving a few where negative words are taken into consideration.

Further, the results can be evaluated by setting a threshold on the percentage calculated to determine the output of the system. For example, any value below 65% is considered as ”false” or ”0” and any value above 65% is considered ”true” or ”1”. This metric helps us to generate crisp values for the output produced instead of fuzzy values.

V. Conclusion

The algorithm developed is capable of finding a similarity between two sentences in terms of percentage value. By following the steps mentioned in the implementation section, we were able to achieve an accuracy of 80-85% for single line sentence and 60-65% for multi line sentences.

For negative sentences, the algorithm produces a non satisfactory output as the word vector formed by the sentence does not hold the magnitude of the negative words since they get removed in the pre-processing step.

This algorithm can be used in a variety of use cases which can be listed as follows -

A. Descriptive Examination and Result Assessment.

The system compares the semantics of sentences, thus it can be implemented to assess Descriptive answers with the data fed to system from trusted sources. Also the sequencing of points of an answer can be tracked to correctly evaluate procedural answer. The generated confidence for each answer can be further reviewed by Higher Authority Examiners and final result can be generated.

B. Plagiarism Analysis.

Many Plagiarism assessment systems fail to check for the originality in paper as they compare the sentences based on the syntactic comparison or words repeated. Since our system understands the meaning of sentences it will be more effective in checking the Plagiarism of Content with the fed data.

C. AI and Chat Bots.

Response of AI and Chat bots can be improved by feeding them more refined data or by making them more capable of understanding the Query or Request by the user.

D. Removal of duplicate entries.

There are many situations when people try to add or ask similar kind of questions on any forum. In such cases manual checking has to be done for removal of such duplicate entries. With the help of our algorithm those tasks can be automated as it will identify the similar questions and mark them as duplicate.

The future aim is to increase the accuracy further more for larger sentences and make the algorithm more performant by increasing the corpus size to cover more words.

References

[1] R. Ferreira, R. D. Lins, F. Freitas, B. Avila, S. J. Simske and M. Riss, ”A New Sentence Similarity Method Based on a Three-Layer Sentence Representation”, 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Warsaw, 2014, pp. 110-117.

[2] E. Brajković and D. Vasić, ”Tree and word embedding based sentence similarity for evaluation of good answers in intelligent tutoring system”, 2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, 2017, pp. 1-5.

[3] Tomas Mikolov, Efficient estimation of word representations in vector space, 2013.

[4] M. Lobur, A. Romanyuk and M. Romanyshyn, ”Using NLTK for educational and scientific purposes”, 2011 11th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), Polyana-Svalyava, 2011, pp. 426-428.

[5] Kamal Sarkar, Khushbu Saraf and Avishikta Ghosh, ”Improving Graph Based Multidocument Text Summarization Using an Enhanced Sentence Similarity Measure”, 2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS), 2015

[6] Shashi Pal Singh, Ajai Kumar, Lenali Singh, Mahesh Bhargava, Kritika Goyal, Bhanu Sharma, ”Frequency based Spell Checking and Rule based Grammar Checking”, 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016

[7] Rutal S. Mahajan, Mukesh A. Zaveri, ”Novel Answer Ranking Approach in Question Answering System using Compositional Distributional Semantic Model”, 2018 2018 3rd International Conference for Convergence in Technology (I2CT), 2018