# Extraction Of Similar Semantic Sentence From Wikipedia Citation

**Deekshant Mamodia**
MT 19119
M.Tech CSE

**Prashant Pathak**
MT 19051
M.Tech CSE

**Anchit Gupta**
MT 19060
M.Tech CSE

## Abstract

**Text similarity is used to determine how 'close' two pieces of text are with respect to their surface closeness (lexical similarity) or with respect to their meaning (semantic similarity). Semantic text similarity is a very challenging problem in IR and NLP field. In this project, we try to get a semantically similar sentence in the cited document of the cited text, in the context of Wikipedia. We have explored two things in the project. First, we explored different embedding that converts text to vector and still holds the semantic meaning like the Word2Vec and Doc2Vec model. Second, we have explored many similarity and distance metric which help to calculate the results effectively. Apart from this, we have also designed an end to end application of the above task. We have developed an API and plugin so that our project becomes user friendly.**

*Keywords* **WMD, Word2Vec, Doc2Vec, SIF**

## 1 Introduction

Information Retrieval has a many different applications and among them, Text search is one of the important. Text similarity has many applications, for instance, the search engine uses text similar to find a similar document to the user's query. The similarity is a process by which we determine the relationship between text snippets. It can be used to find surface closeness (lexicons) or semantic closeness (meaning) of words (or sentences). Lexical similarity aims to determine whether two short texts are the same in terms of words or characters. Edit distance, lexical overlap, and largest common sub-string are few methods to find lexical similarity. Semantic similarity finds the similarity between text and document based on their meaning rather than by character matching. Semantic similarity is computed based on corpus-based and knowledge-based measures.

Our task in this project is to find a semantically similar sentence to the cite text in the cited document. This task is challenging because many factors come into play. The main factor is the cited document's nature, as it can be anything in the network ranging from text, image, java scripts, books, research archive, or maybe an empty document. This application can be a huge help to the user as it can reduce the information needs time. User doesn't have to visit the different cited link and read the document themself to get exact information instead he/she can get the exact information within seconds.Also this can be used to find irrelevant cited reference. If during finding the similarity we didn't find any similar document then we can easily say that the cited document is irrelevant to the sentence.

Next section deals with literature review for embedding and closeness measures. In this project, we have developed an end-to-end system for citation text retrieval. Figure 2 shows the system architecture. In the later section, all the components of the system have been explained in detail.

## 2 Literature Review

In this section we discuss previous work related to the different aspects of embedding generation and closeness measure. There are various machine learning and deep learning model that help us generate embedding form the text. In (Mikolov et al., 2013a) the authors proposed a way to convert words in to vectors. Later (Mikolov et al., 2013b) found that semantic and syntactic patterns can be reproduced using vector arithmetic. For example, the male/female relationship is automatically learned, and the induced vector representations, "King -

Man + Woman" results in a vector very close to "Queen.". In (Le and Mikolov, 2014) authors presented DOC2Vec model that help us to convert sentence in to vectors.This gives is a way to produce equal size embedding for variable size input.

(Huang et al., 2016) proposed a new method of finding the similarity between two texts named Word Mover's Distance. This method allows us to assess the "distance" between two documents in a meaningful way, even when they have no words in common. It uses word2vec vector embedding of words. It suggests that distances between embedded word vectors are to some degree semantically meaningful. The distance between two text documents A and B is calculated by the minimum cumulative distance that words from the text document A needs to travel to match exactly the point cloud of text document B. (Kenter and de Rijke, 2015) proposed a measures the similarity between two texts using a supervised machine learning approach. The training data for the supervised step consists of sentence pairs and associated labels that represent the semantic similarity between the two sentences. The author proposed a new normalized factor for each vector named Smooth Inverse Frequency.The SIF downgrades unimportant words such as but, just, etc., and keeps the information that contributes most to the semantics of the sentence.

## 3 Dataset Analysis

Wikipedia is a multilingual, web-based, free-content encyclopedia project supported by Wikimedia Foundation and based on a model of openly editable content. It provides free data dumps to user for research purpose. To implement the project we have used dataset of 16 GB (compressed). The dataset contains articles in the XML format. The dump also includes metadata such as the log files, revision history of each article, short abstract of each article with titles and URL of the article. This data is present in the compressed format. This contains the 6,023,753 articles in the English Wikipedia containing over 3.5 billion words. While processing it we got approx 45 lakhs of vocab size.

## 4 Baseline Implementation

In baseline implementation, we learned embedding for our words and sentences and then used different ways to compute similarity between vector.

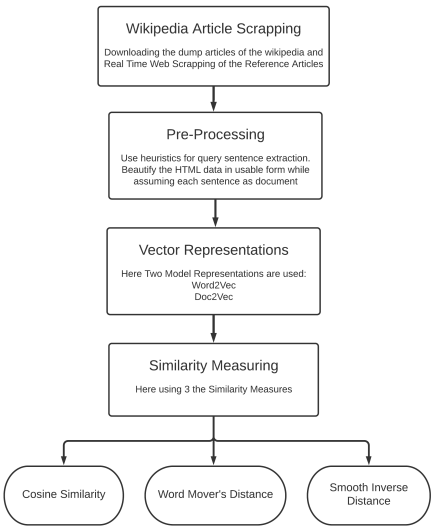


Figure 1: **Baseline implementation Flowchart**

First, we trained Doc2Vec(Le and Mikolov, 2014) model by taking each sentence of each document as a document and attach tags with each sentence. For finding the best match for each cited sentences, we scrap the Wikipedia pages at run time and made a Word2Vec and Doc2Vec representation of each reference document and Wikipedia page. Now to compute semantic similarity between embedding by calculating cosine similarity, Word Mover's Distance, Smooth Inverse frequency. Flowchart is shown in figure 1.

## 5 System Architecture

The proposed system is a complete end to end solution. The system architecture is divided into three parts which are shown in the figure 2.The front end is for the user to enter the text or citation number of which he/she wants to find the best match in the cited text.

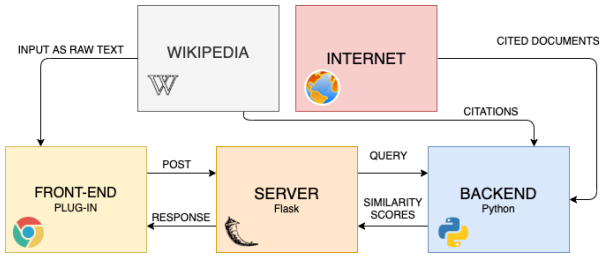


Figure 2: **System Architecture of Wikipedia Semantics Model**

The server takes the input from the user and sends it to the back-end of the system. The back-end part running the machine learning models

which are trained on the Wikipedia corpus. These algorithms use some similarity measures to compute text similarity and returns the result. Now we will explain each component in detail.

### 5.1 Backend Architecture

The Back-end architecture is consist of machine learning supervised and unsupervised models. These models takes pre-processed raw text and output the embedding of the text. These embeddings are used for measuring the similarity between query text and cited document text. The machine learning models are explained below in detail and architecture is shown in the figure 3. 4 shows data flow in the backend part.

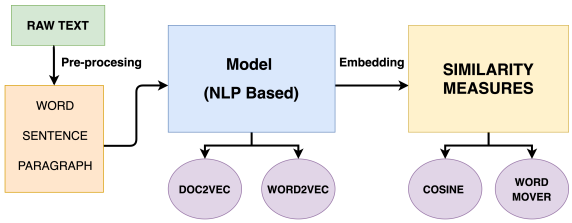


Figure 3: **The Back-end Architecture of the Wikipedia Semantics Model**

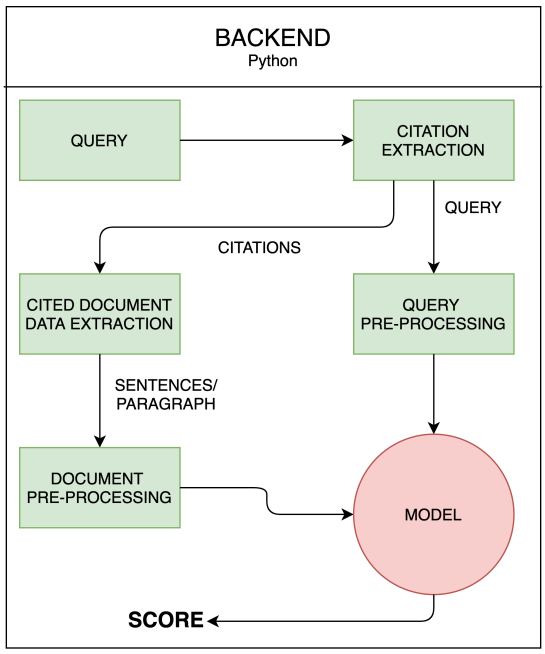


Figure 4: **The data flow in the backend.**

#### 5.1.1 Pre-processing of Text

The Wikipedia text and text of other cited are not in a good format so we cannot directly pass it to the models. To make text in a better format we first have to process the text. We have used the following as pre-processing steps.

- **Text to Lower-case:** The text are converted into lower form. Remove all non-alphanumeric characters from the tweet.
- **Tokenization of text:** In the training models, we need words or sentences to train our model. So we convert each text into the list of words and list of sentences.
- **Expand Clitics:** The text is written on Wikipedia is written by people so they may use a clitic form of the word. The clitic word combines when we remove punctuation from the text. So to preserve the meaning of clitics we use the contraction library which converts the word into the original form.
- **Remove Punctuation:** The punctuation is removed from the text as there is no use while training the models. But the dot is preserved to find the sentences in the paragraphs.
- **Lemmatization:** The lemmatization is the process of grouping together the inflected forms of a word and replaces it by dictionary form of a word. This will help to reduce the size of the vocabulary. We use the lemmatization on nouns.

#### 5.1.2 Embedding

The embeddings are the compact representation of the word or sentence and carry some kind of semantic meaning of the words or sentences. To measure the semantic similarity between texts we need embedding of each text. The text can vary from word to the paragraph. So to find embedding of each type of text we define three types of embeddings.

**Word2Vec for Words:** Word2Vec is a shallow, two-layer neural network which is trained to reconstruct linguistic contexts of words. It takes a large corpus of words as input and produce unique vector embedding with large dimensions for each word. For our project, we use the skip-gram model which helps to predict the target words from the context word. We gave input a list of words of text and got the embedding of each word and take the mean of all the embedding of each word. This way we got the embedding of each query text and text of the cited document.

**Doc2Vec for Senetnces:** The Doc2Vec model is based on the Word2Vec model. It is an unsupervised learning approach to learn document representation. We used the DBOW approach it uses paragraph vector to classify entire words in the document. In training, it samples a list of words, and then classifier classifies whether word belongs to the document in this way the vector representation of each word is learned. As input, we passed the sentence of the query text and text of the cited document and got fixed size embedding of each text.

**Word2Vec for Paragraph:** Now for finding the paragraph embedding, we use the same Word2Vec model but in a different way. First, we gave a list of words of one paragraph and model return the embedding of each word. Then we take the average of all the word embedding of the input words. This way got the embedding of each paragraph.

#### 5.1.3 Training of Models

We have implemented two models for the project. The two models are Word2Vec and Doc2Vec models. These models are trained on the pre-processed text of Wikipedia articles. A total of 98000 articles are used for training. The Word2Vec model is trained on the principle of skip-gram. The skip-gram is an unsupervised learning technique and used to most related words of a given word. This approach is used for large data-sets.

The Doc2Vec model is trained on the principle Distributed Bag of Words version of Paragraph Vector. It will classify whether word belongs to the document or not and in this way it learned.These models output is the embedding of each word or sentence in the training data.

#### 5.1.4 Similarity Measure Approaches

To measure the similarity between the query text and text of cited documents we have used two different methods which are discussed below:

- **Word Mover Distance:** To measure the topical similarity the word mover distance is used. WMD use word embeddings to calculate the distance so that it can calculate even though there is no common word. The assumption is that similar words should have similar vectors. It assumes a higher frequency of words it is more important. It transfers every word from text 1 to text 2 and choose minimum transportation cost to transport every word from text 1 to text 2.

- **Cosine Similarity:** Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The advantages of cosine similarity if two documents are far apart by the Euclidean distance, chances they may still close to each other.

- **Smooth Inverse Frequency:** This measure solves the problem of too much weight to the words that are quite irrelevant. It changes the word embedding of each word by using the formula a/(a + p(w)) where a is 0.001 and p(w) is the estimated frequency of the word in the corpus. SIF downgrades the unimportant words such as stopwords.

### 5.2 Front-end Architecture

The Front-end architecture is developed for the users to enter the text for which they want the best match citation text. The front-end is a chrome browser plugin which works on the Wikipedia articles. It works as first user have to select the query text, then click on plugin and submit it. Then it shows best suitable output for each citations.

### 5.3 Server-end Architecture

The server end is handled through an API developed with the help of Flask framework python. The API takes user query text as input and transfer the input to backend for further preprocessing . And as output receives the similarity scores with processed result from cited document and best is selected for delivery.

## 6 Analysis and Result

To analyze the different model and their embeddings with different approaches to the text similarity, We run our models on 9 different topics with different numbers of citations in the query text. We measured the R-Precision of each query. We extract the top matching results of each query from the different models on different text similarity measures. Now for ground-truth for each query, we take the results of the pre-trained Word2Vec model with WMD on Wikipedia on given queries as ground truth. The precision value of each query on a different models is shown below in the graph 5.

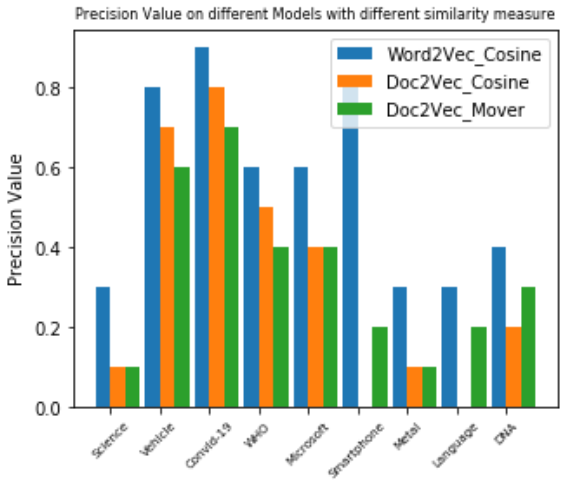


Figure 5: **Comparison of Different Model on Different Queries with Cosine and WMD Text Similarity Measures**

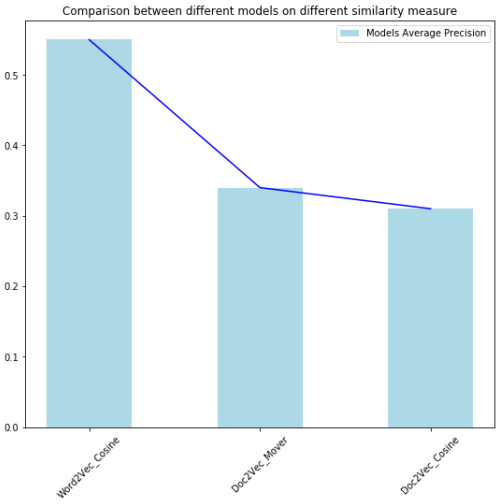


Figure 6: **Average Precision value of Word2Vec and Doc2Vec Model with Cosine and WMD**

The 6 graph shows that the Word2Vec model has high precision value on each query with cosine similarity as a text similarity measure. And the Doc2Vec model does not perform well with any text similarity measure. Below table also show same results.

Table 1: Model Performances

| Models | Average Precision |
|---|---|
| **Word2Vec Cosine** | 0.55 |
| **Doc2Vec WMD** | 0.34 |
| **Doc2Vec Cosine** | 0.31 |

So to analyze which model is best for finding the similar text. We calculate the Average Precision value of each model with all similarity measure on the query mention in figure 5. The average precision of each model is shown in graph 6.

## 7 Conclusion

From the result of this project we can infer that the type of embedding used plays an important role in the performance of the model. So to get a better precision value we can use better embedding like Universal Sentence Encoder (USC) proposed in (Cer et al., 2018) and BERT embedding proposed in (Devlin et al., 2018). Also other closeness measures such as LDA can be used. In future this project can be extended to work on these models for better results.

## References

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Gao Huang, Chuan Quo, Matt J. Kusner, Yu Sun, Kilian Q. Weinberger, and Fei Sha. 2016. Supervised word mover's distance. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4869–4877, Red Hook, NY, USA. Curran Associates Inc.

Tom Kenter and Maarten de Rijke. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, page 1411–1420, New York, NY, USA. Association for Computing Machinery.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.