

# PHRASING SIMILARITY EXTRACTION

ROZEENA<sup>1</sup>, MOHD WASIM<sup>2</sup>

<sup>1</sup>MAHATMA GANDHI MISSION'S COLLEGE OF ENGINEERING AND TECHNOLOGY, NOIDA

<sup>2</sup>Division of Research and Development, Lovely Professional University, Punjab

## ABSTRACT

This project is based on word2vec model and use it for extraction of similarity from the large datasets. At this time, bulky datasets are problem and it is difficult to maintain them, most of the time redundant data make datasets messy but to improve it here we have word2vec model where we can extract similar words or documents and cluster them. Suppose we have large dataset of news articles where many of them have similar headlines which make it little messy and hard to handle it because of redundancy. With the help of word2vec extraction of similar headlines is easy and cosine similarity will help in to find out percentage of similarity between them.

## INTRODUCTION

Natural Language processing(NLP) is a combinational techniques of languages and computation, it gives complete representation of human(natural) language and information engineering. Computers analyze the large amount of natural language data help in like speech recognition, voice text messaging, spell check, etc. In NLP processing computers identify the appropriate word, phrase and in response give result like human.[2][7]

NLP started in 1950s and many research has already proposed, many evolution has already taken place like auto-check, auto-correct, descriptive analysis, machine translation, survey analytics, etc. In 1950s to 1960s NLP was encountered with semantic and syntactic problems and they took seven minutes to process single sentences. But, now as we can see this digital era is completely under controlled in NLP and it processing many large documents every single minute by the help of higher programming languages and also facing problems with them.[9]

Sometimes, unorganized data make task difficult to analyze them and these datasets decrease the performance of system. In this report we will discuss on one problem that is extraction of similarity. Redundancy(having of same data) is the major problem in dataset and extraction of similarity somewhere can reduce it. It also help in extract and cluster similar data in same dataset. Here, we used word2vec model in extraction of similarity.

Word2vec is a model to produce word embedding and it was introduced in 2013. It converts words into vectors and find out distance between them. Further, it represent less distance vectors together on graph. In this project, we are using news articles dataset which consist many similar contents and by the help of word2vec we extract them. [3][6]

## MOTIVATION

Currently there are many research is going on reducing the complexity of large dataset and NLP is one of the revolutionary change idea that deals with read, decipher and make sense of human language in a manner that is valuable in computer field. NLP is now top business choice as it has many applications like address customer pain by monitoring customer feedback using sentimental analysis where it can judge negative and

positive response, gather market intelligence from unstructured data and extract scaled informations for marketing, reduce customer frustration with hybrid bots where customer get response according to their chat. [2]

Here, with phrasing similarity extraction using word2vec we have a chance to reduce time complexity for large dataset and also can use for marketing purpose by analyzing its similar records. Words that are similar in contexts have similar embedding which measured by cosine similarity( check their angles and find distances between them ). Numbers of alternatives for phrasing similarity has already came up in NLP research field but word2vec has its own impact in this field as its execution power is fast and probability of correction is very high.[3][6]

## APPLICATIONS: EXTRACTION OF SIMILARITY

**Biomedical:** Semantic relatedness and similarity between biomedical terms can help in medical science for example diabetes and insulin are related terms while glucose and blood sugar are similar terms. Large datasets in medical field is common and similarity between biomedical terms can reduce the complexity in analyzing the results of patients.[8]

**Retrieving Information:** Retrieving similar information can reduce time consumption and also can reduce the computational complexity.

**Organized dataset:** Clustered similar informations together generate organized dataset which is easy to analyze.

## DATASET

The news dataset from DATA.GOV( the home of the U.S. Government's open data) consists 9 columns and 40lakh rows but we used upto 500 rows. It is comma separated values(csv) dataset. This dataset consists lot of articles some them are little similar.[1]

Name of elements used in columns

1. id
2. title
3. Publication
4. author
5. date
6. year
7. month
8. url
9. content

## PROPOSED METHODS

Here, we have some proposed methods which use in to preprocess the data.

## Word2vec

Word2vec creates vectors that are distributed numerical representations of word features and group the vectors of similar words together in vector space. It does so without human intervention. Here, we defined word2vec function or trained it for word2vec model by using csv file and convert into binary file.[3][6]

## Removal of Stopwords

Stopwords are common words generally use in language(like in english of, the, are) that complete the sentences. The removal of stopwords is important because processing of stopwords along with main words consume lot of time for example if you search “how to develop an application” on search engine and if it try to search every term like “how” “to” “develop” “an” “application” then it will load many pages related to “how” “to” “an” “develop” “application” but if stopwords are removed it will search pages which are related to “develop” and “application”. [4]

In this project removing of stopwords is necessary because further tokenizer will tokenize each word and if stopwords remains there then tokenizer will take long time to process. Stopwords generally are hindrance for tokenizer it low down the performance of system so, it is necessary to remove them and then extract similar data.

## Tokenizer

Tokenizer is the package use for tokenization. Tokenization is the process of tokenizing or splitting a string, text into a list of tokens. One can think of token as parts like a word is a token in a sentence, and a sentence is a token in a paragraph. Types of tokenization:

- Text into sentences tokenization
- Sentences into words tokenization
- Sentences using regular expressions tokenization

Example: Sentence into word tokenization

Before tokenization: ‘hello everyone, how are you’

After tokenization: “hello” “everyone” “how” “are” “you”[4]

## Cosine Similarity

Cosine similarity is use for the similarity and orthogonality of two vectors.

Formula for Cosine similarity:

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos \theta$$

$$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

Theta is the angle between two vectors and the main source of telling similarity between two words. If theta is zero then both vectors are similar to each other but if it 90 then both vectors are not related to each other and if theta is 180 then both vectors are opposite to each other.[4]

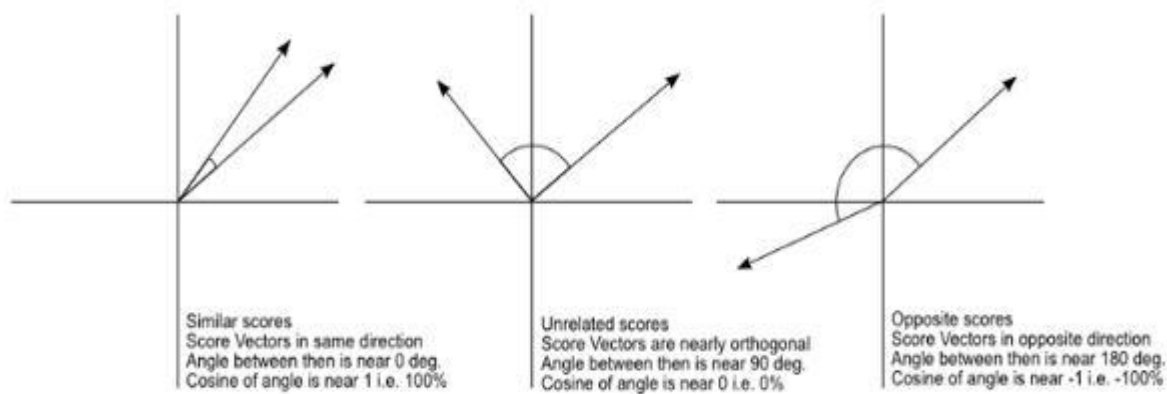


Fig. Cosine Similarity

## Overview of code

### 1. Train Word2vec model

```

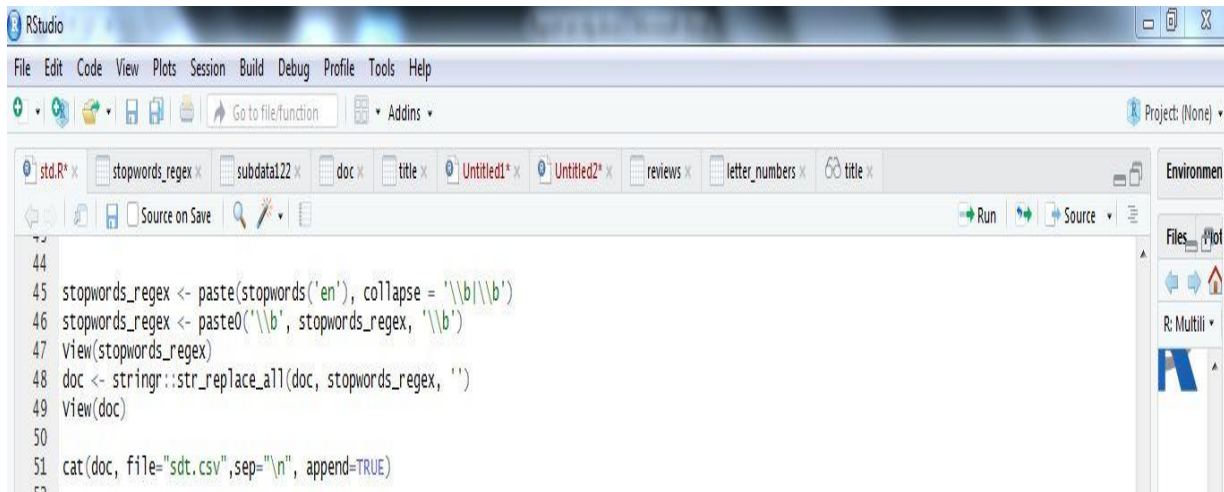
9- word2vec <- function(filename) {
10-   if (grepl('.csv', filename, fixed=T)) {
11-     binaryfilename <- gsub('.csv', '.bin', filename, fixed=T)
12-   }
13-   else {
14-     binaryfilename <- paste0(filename, '.bin')
15-   }
16- }
17-
18- # Train word2vec model.
19- if (!file.exists(binaryfilename)) {
20-   # Lowercase and setup ngrams.
21-   prepfilename <- 'temp_prep'
22-   prep_word2vec(origin=filename, destination=prepfilename, lowercase=T, bundle_ngrams=2)
23- }
24- # Train word2vec model.
25- model <- train_word2vec(prepfilename, binaryfilename, vectors=200, threads=4, window=12, iter=5, negative_samples=0)
26-
27- # Cleanup.
28- unlink(prepfilename)
29- } else {
30-   model <- read.vectors(binaryfilename)
31- }
32-
33- model
34- }
35- if (!file.exists('text8') && !file.exists('text8.zip')) {
36-   temp <- tempFile()
37-   download.file('http://mattmahoney.net/dc/text8.zip', temp)
38-   unzip(temp)
39-   unlink(temp)
40- }

```

Fig. Screenshots of code(training a word2vec model )

Here, we made functions which train the word2vec model. This function will convert file.csv into file.bin.

## 2. Removing of Stopwords



```

44
45 stopwords_regex <- paste(stopwords('en'), collapse = '\\b\\b')
46 stopwords_regex <- paste0('\\b', stopwords_regex, '\\b')
47 view(stopwords_regex)
48 doc <- stringr::str_replace_all(doc, stopwords_regex, ' ')
49 view(doc)
50
51 cat(doc, file="sdt.csv", sep="\n", append=TRUE)
52

```

**Fig. Removing of Stopwords**

Stopwords is in-built function which help in removing stopwords from dataset. After removing stopwords from dataset we have new dataset without stopwords. Further, in place of stopwords we are putting space there and now we have complete simple new dataset for tokenization so, we are going to write them in new file(sdt.csv).

## 3. Tokenization and Training

```

52
53 # Train word2vec model and explore.
54 model <- word2vec('sdt.csv')
55 model %>% closest_to("Republicans")
56

```

**Fig. Tokenization and training**

```

65 # Train word2vec model and explore
66 model <- word2vec('text8')
67 model %>% closest_to("computer")
68

```

**Fig. Tokenization and Training**

```

> model %>% closest_to("computer")
      word similarity to "computer"
1      computer                1.0000000
2     computers                0.7723743
3  personal_computer          0.6946393
4      computing              0.6938783
5      machines              0.6934333
6  mainframe_computer          0.6791429
7 multitasking_operating      0.6640214
8    microcomputer            0.6593373
9  personal_computers          0.6532452
10     hardware                0.6522595

```

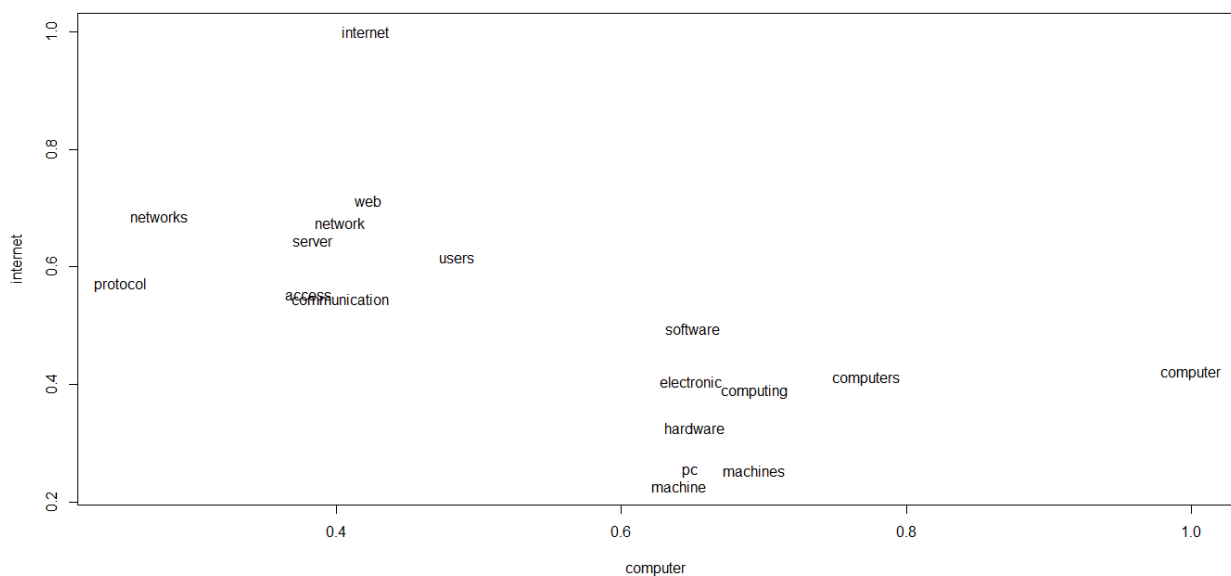
**Fig. Output of similar words**

Here, we trained and tokenized the data. In first screenshot we trained the model and in second we are testing the model with the word computer.

#### 4. Use of cosine similarity

```
69 # Plot similar terms to 'computer' and 'internet'.
70 computers <- model[[c("computer","internet"),average=F]]
71
72 # model[1:300,] here restricts to the 3000 most common words in the set.
73 computer <- model[1:300,] %>% cosinesimilarity(computers)
74
75 # Filter to the top 10 terms.
76 computer_and_internet <- computer_and_internet[
77   rank(-computer_and_internet[,1])<10 |
78   rank(-computer_and_internet[,2])<10,
79 ]
80
81 plot(computer_and_internet,type='n')
82 text(computer_and_internet,labels=rownames(computer_and_internet))
83 |
```

**Fig. Use of Cosine Similarity**



**Fig. Plotting**

Here we are using cosine similarity and comparing these two words (computer and internet) with other words from the dataset. After complete process we plotted them on graph.

## CONCLUSION

In our project we used word2vec, stopwords, tokenization and cosine similarity for large data processing. First we filtered the dataset by removing stopwords and then we tokenize the data. After that by the help of cosine similarity we found out distance between vectors and plotted them on graph. We also used many in-built packages which directly help us in this project and working on NLP project using R programming is good choice because of many in-built packages.



Word2vec is best choice to handle the large and complex data. It does not require human involvement, thus saving time, resources, it also reduce the computational complexity and produce results from big data by analyzing it that humans are unable to do.

NLP can give revolutionary change in data science such as data classification and clustering is now easy. There are lots of scopes in research field for NLP and its requirement in social media is mandatory because of producing excess amount of data on daily basis and human cannot handle it manually. But evolution in NLP is also needed.

## REFERENCES

1. <https://www.data.gov>
2. Fouad Nasser A Al Omran, Christoph Treude, "Choosing an NLP Library for Analyzing Software Documentation: A Systematic Literature Review and a Series of Experiments" 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)
3. Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, Kilian Q. Weinberger, "From Word Embeddings To Document Distances" Proceedings of the 32 nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s)
4. [https://medium.com/@paritosh\\_30025/natural-language-processing-text-data-vectorization-af2520529cf7](https://medium.com/@paritosh_30025/natural-language-processing-text-data-vectorization-af2520529cf7)
5. Yang Liu, Zhiyuan Liu, Tat-Seng Chua, Maosong Sun, "Topical Word Embeddings" 2015, Association for the Advancement of Artificial Intelligence (www.aaai.org).
6. Sahar Ghannay, Benoit Favre, Yannick Esteve and Nathalie Camelin, "Word Embeddings Evaluation and Combination"
7. <https://www.wonderflow.co/blog/natural-language-processing-examples>
8. Yongjun Zhu, Erjia Yan and Fei Wang, "Semantic relatedness and similarity of biomedical terms: examining the effects of recency, size, and section of biomedical publications on the performance of word2vec" Zhu et al. BMC Medical Informatics and Decision Making (2017) 17:95 DOI 10.1186/s12911-017-0498-1
9. Erik Cambria, Bebo White "IEEE Computational intelligence magazine | may 2014" Digital Object Identifier 10.1109/MCI.2014.2307227 Date of publication: 11 April 2014