VERB SENSE DISAMBIGUATION

-STUDY AND ANALYSIS

A Project Report

Submitted by:

PRANAY JAIN (15-1-4-035)

PRAMOD SINGH YADAV (15-1-4-101)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

at

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR,

SILCHAR, ASSAM (INDIA)-788010

MAY'2019

DECLARATION

I hereby declare that the project entitled “Verb Sense Disambiguation- Study and Analysis” submitted for the B. Tech. (ECE) degree is my original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

Signature of the Students

Place:

Date:

CERTIFICATE

This is to certify that the project titled “Verb Sense Disambiguation- Study and Analysis” is the bona fide work carried out by Pranay Jain (15-1-4-035) and Pramod Singh Yadav (15-1-4-101), students of B. Tech (ECE) of National Institute of Technology Silchar (An Institute of National Importance under MHRD, Govt. of India), Silchar, Assam, India during the academic year 2018-19, in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (Electronics and communication Engineering) and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

Name and Signature of the Supervisor

Place:

Date:

ABSTRACT

Many words in Indian languages as well as foreign languages have different meanings according to the context in which they are being used. Ambiguity of Noun and Verb are most common and can be removed by observing their surrounding terms in the context. In machine translation, it is very important that the machine is able to distinguish the exact sense of each word when a sentence is input into it. The exact sense of the word will depend upon the words along with which it is being used in a sentence. Most importantly, the sense of the verb will depend on the subject, i.e., noun with which it is used in that particular sentence. Our approaches hover around the same idea, i.e., to obtain a meaningful relationship between a word and its surrounding terms. This is a comparative study of different methods used to remove this ambiguity. In the first method, we are using different types of correlation techniques namely Pearson correlation, Spearman Correlation to find the level of relatedness between the verb and the noun with which it is used in that sentence. In the second approach, we classified the different forms of verbs according to the sense of that verb and calculated its co-occurrence frequency with the noun. In the third method, we mapped verb with document and noun with document and calculated their cosine similarity which gave us good results. This project helped us in understanding various techniques used in Natural Language Processing.

Acknowledgment

We feel immensely privileged to extend our heartfelt gratitude towards our guide, Dr. Koushik Guha for his invaluable guidance and influential mentorship exhibited during the course of this project. Owing to his technical advice, exemplary guidance, persistent monitoring and constant encouragement towards achieving perfection, it has been possible to take this project forward to this stage. We are obliged for the valuable information and the helping hand provided by him amidst his own busy schedule.

List of Figures

List of Tables

Table of Contents

Chapter 1 : Introduction

Word Sense Disambiguation (WSD), a subtask of Machine Translation is one of the most active research areas that originated in the 1950s. The ambiguity is present in almost all the natural languages spoken in the world which sometimes makes it difficult to get the correct meaning or sense of a word in context. The words (nouns or verbs) having multiple meanings are known as *polysemous words.* Word sense disambiguation is the process of removing the ambiguity of polysemous verb or polysemous noun present in the context by detecting the correct meanings of the word. In normal circumstances, the human brain is quite apt in understanding the sense of polysemous words, but in case of machines (computers), we need to devise a methodology that will help the computer to detect the correct meaning of the ambiguous words.The ambiguous word in the context can be a polysemous noun or polysemous verb. For example, in the phrase "*The bank down the street was robbed*", the word bank is a polysemous noun. In this example, the word bank means a financial institution, while in the "*the city Kolkata is on the bank of Ganges*", in this sentence the word bank is used as shore of a river. In the same way, for the example such as "*He gives his opinion on the matter*", the word give is a polysemous verb. For this example, the verb give is used as judgment while "*she gives him First Aid"* is used as apply. There are many polysemous verbs where meaning or sense of verb can be changed with respect to the surrounding terms specially the nouns. Removing the ambiguity of verb requires to find out the surrounding terms of verb as well as the surrounding terms of the noun that is used along with the verb. The terms in the neighbourhood of a verb are those with the highest cosine similarity value with noun. In this case, if too few terms are selected as neighbour terms of noun and verb then the relevant features might miss out and if too many terms are selected then irrelevant features may crop in.

1.1 Problem Statement

Most words in most languages can be used in several different ways so that their meaning is subtly or not so subtly modified by their context. Dictionaries, therefore, distinguish multiple senses of a word. To demonstrate the diversity of word senses, consider this selection form Webster's Collegiate Dictionary from the 30 senses listed for the verb *run* (intransitive):

The horse runs.
The ship runs before the wind.
The cat ran away.
The salmon run every year.
My horse ran last.
A breeze ran through the trees.
A vine runs over the porch.
The machine is running.
The colors run.
Blood runs in the veins.

The meaning of the predicate *run* is different in each of these examples: *the horse runs* in a different way than *the machine* or *the colors*—and *run away* and *run aground* are different yet, although all of these uses of *run* have a core meaning in common. The exact meaning of a predicate depends on the argument it operates upon. Predication creates new meanings in every context by combining the meaning of the argument and appropriately selected aspects of the meaning of the predicate. It is not the whole meaning of *run* that applies to *the vines running over the porch*, or *the blood running in the veins*, but only features that are relevant to the argument of the predication. There is no need to distinguish between the different senses of a word in a lexicon, and particularly the mental lexicon. The core meaning of each word in a language is well defined, but is modified in each context. Word senses emerge when words are used in certain special, typical contexts. Indeed, every context generates its own word sense. The differences between the contextual meanings of a word may be small or large, but they are always present. The decontextualized word meaning is nothing but an abstraction, though a very useful one. Specifically, in predication the meaning of the predicate is influenced by the argument of the predication.

Chapter 2 : Literature Survey

The exact meaning of a word in a sentence depends upon the context in which it is being used, i.e., the meaning of the word will depend upon the terms which are being used in that sentence. So we have to map the sense of the word with one or more of its surrounding terms.

2.1 Existing System

A lot of work has been done to remove the ambiguity of the words in a sentence. Most of the work has been focused on removing the ambiguity of the noun and very less work on removing the ambiguity of the verb. Lot of technologies have been developed like LSA (Latent Semantic Analysis) which helps us in converting a word into vectors as Mathematical calculations cannot be done on vectors. We need numbers or vectors to do the mathematical calculations. LSA serves this purpose. Same as that tf-idf and cosine similarity has been developed to do the calculations on LSA and find the closeness of two sentences. There are also mathematical formula for finding out the correlation between vectors through the means of Pearson and Spearman correlation coefficient.

2.2 Proposed System

We have to design such a system that machine itself tells us about the difference between the different meanings of the verb depending on its surrounding term and especially the noun used in that sentence. We have many features of natural language processing system which are already present there and abundantly in use like cosine similarity, tf-idf values, normalized frequency, correlation etc. We propose to use all these to design a system which would remove the ambiguity from the meaning of the verb. We have designed 3 methodologies to tackle the problem of removing the ambiguity of the verbs.

Chapter 3 : System Analysis and Design

3.1 Sample dataset

The dataset we have used consists of 267 statements and 2099 words in total. Here are some of the sentences present in the dataset. The vertical bar indicates end of a sentence.

```
The horse runs very fast |
My horse runs last |
The horse is running in the park |
The horse is running in the race-course |
The horse ran very fast in the race |
The rabbit is running in the garden |
The rabbit runs very fast |
The rabbit is running around in the house |
The kangaroo runs with a baby in its pouch |
The kangaroo is running in the forest |
The machine is running very smoothly |
The machine runs on electricity |
The machine is running in the factory |
The machine ran properly for many hours |
The machine runs on crude-oil |
The colors run |
These dyes and colors are guaranteed not to run |
blood runs in the veins |
blood runs from the heart to all parts of the body through artery |
The bus runs between railway station and airport |
The train runs between two states |
He runs a new program on the laptop |
The computer runs the instruction |
The film runs for 3 hours |
She runs 10 miles daily |
He runs daily through out the year |
The ship runs before the wind |
They run the tapes over and over again |
He never tires of running that video |
The apple runs large this years |
She gave him a black eye |
The draft gave me a cold |
He gave me a lot of trouble |
Our meeting gave much interesting information |
She gave the children lots of love |
I gave her my money |
```

Figure 3.1: Sample dataset of the Project

3.2 Latent Semantic Analysis

Latent semantic analysis (LSA) is a technique in natural language processing, in particular distributional semantics, of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. LSA generates a high-dimensional semantic space from the analysis of a large corpus of written text. LSA must be trained with a large corpus of written text. The raw data for LSA are meaningful passages and the set of words each contains. LSA knows only what it has been taught. If words are used that did not appear in the training corpus, or which are used differently than in the training corpus, LSA, not unlike a person, does not recognize them correctly or at all. LSA assumes that words that are close in meaning will occur in similar pieces of text (the distributional hypothesis). LSA thus makes the strong psychological claim that word meanings can be represented as vectors in a semantic space. But not only word meanings are represented as vectors in this space, documents are similarly represented as well. And new documents—sentences, paragraphs, essays, whole book chapters—can also be represented as vectors in this same space. This is what makes LSA so useful. It allows us to compare arbitrary word and sentence meanings, determine how related or unrelated they are, and what other words or sentences or documents are close to them in the semantic space. A matrix is constructed whose columns are words and whose rows are documents. The cells of the matrix are the frequencies with which each word occurred in each document.

267x637 cell

	1	2	3	4	5	6	7	8	9	10	11	12
1	[]	'$5000'	'10'	'3'	'30'	'4'	'70'	'Agra'	'Auden'	'Baltic-Rep...	'Conference'	'David'
2	1	0	0	0	0	0	0	0	0	0	0	0
3	2	0	0	0	0	0	0	0	0	0	0	0
4	3	0	0	0	0	0	0	0	0	0	0	0
5	4	0	0	0	0	0	0	0	0	0	0	0
6	5	0	0	0	0	0	0	0	0	0	0	0
7	6	0	0	0	0	0	0	0	0	0	0	0
8	7	0	0	0	0	0	0	0	0	0	0	0
9	8	0	0	0	0	0	0	0	0	0	0	0
10	9	0	0	0	0	0	0	0	0	0	0	0
11	10	0	0	0	0	0	0	0	0	0	0	0
12	11	0	0	0	0	0	0	0	0	0	0	0
13	12	0	0	0	0	0	0	0	0	0	0	0
14	13	0	0	0	0	0	0	0	0	0	0	0
15	14	0	0	0	0	0	0	0	0	0	0	0
16	15	0	0	0	0	0	0	0	0	0	0	0
17	16	0	0	0	0	0	0	0	0	0	0	0
18	17	0	0	0	0	0	0	0	0	0	0	0
19	18	0	0	0	0	0	0	0	0	0	0	0
20	19	0	0	0	0	0	0	0	0	0	0	0

Table 3.1: LSA matrix of the dataset.

3.3 Bag of Words

A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modelling, such as with machine learning algorithms. The bag-of-words model is a simplifying representation used in *natural language processing* and *information retrieval* (IR). In this model, a text (such as a sentence or a document) is represented as the *bag (multiset)* of its words, disregarding grammar and even word order but keeping *multiplicity*. The approach is very simple and flexible, and can be used in a myriad of ways for extracting features from documents. A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

- A vocabulary of known words.
- A measure of the presence of known words.

It is called a "*bag*" of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document. Bag of words is constructed

from dataset by removing punctuations, prepositions and same words. So our dataset is shortened to a set of verbs and its surrounding terms.
For example if a sentence in dataset looks like – *The horse is running in the park.* The terms added to Bag of Words will be {'*horse*', '*running*', '*park*'}.

The Bag of words obtained consists of 636 unique words derived from dataset.

1x636 cell

	225	226	227	228	229	230	231	232	233	234	235	236
1	feeling	figure	film	finally	finish	fire	first	fool	foot	for	forest	fort
2												

Table 3.2: Bag of Words Model of Dataset

3.3.1 Term Frequency and Normalized Term Frequency

Term Frequency also known as TF measures the number of times a term (word) occurs in a document. In reality each document will be of different size. On a large document the frequency of the terms will be much higher than the smaller ones. Hence we need to normalize the document based on its size. A simple trick is to divide the term frequency by the total number of terms in that document.

3.3.2 Term Frequency-Inverse Document Frequency

TF-IDF, which stands for term frequency—inverse document frequency**,** is a scoring measure widely used in information retrieval (IR) or summarization. TF-IDF is intended to reflect how relevant a term is in a given document. The intuition behind it is that if a word occurs *multiple times in a document*, we should boost its relevance as it should be more meaningful than other words that appear fewer times (TF). At the same time, if a word occurs many times in a document but also *along many other documents*, maybe it is because this word is just a frequent word; not because it was relevant or meaningful (IDF). We can come up with a more or less subjective definition driven by our intuition: a word's relevance is proportional to the amount of information that it gives about its context (a sentence, a document or a full dataset). That is, the most relevant words are those that would help us, as humans, to better understand a whole

document without reading it all. As pointed out, relevant words are not necessarily the most frequent words since stop words like “the”, “of” or “a” tend to occur very often in many documents. There is another caveat: if we want to summarize a document compared to a whole dataset about a specific topic (let’s say, movie reviews), there will be words (other than stop words, like *character* or *plot*), that could occur many times in the document as well as in many other documents. These words are not useful to summarize a document because they convey little discriminating power; they say very little about what the document contains compared to the other documents.
For each column in the matrix, a idf value is calculated using the formula-

IDF (word)= 1+log$_e$(total no. of documents/no. of documents with the term in it)

This IDF value for each word is then multiplied with the normalized term frequency to obtain tf-idf values for each word in a sentence. After this process we have a matrix full of words with their respective *tf-idf* values for each sentence.
Sample TF-IDF Values-

266x636 double

	38	39	40	41	42	43	44	45	46	47	48
1	0.2728	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0.1949	0	0	0	0	0	0	0	0	0	0
4	0.1949	0	0	0	0	0	0	0	0	0	0
5	0.1705	0	0	0	0	0	0	0	0	0	0
6	0.1949	0	0	0	0	0	0	0	0	0	0
7	0.2728	0	0	0	0	0	0	0	0	0	0
8	0.1705	0	0	0	0	0	0	0	0	0	0
9	0.1516	0	0	0	0	0	0	0	0.1549	0	0
10	0.1949	0	0	0	0	0	0	0	0	0	0
11	0.2273	0	0	0	0	0	0	0	0	0	0
12	0.2728	0	0	0	0	0	0	0	0	0	0
13	0.1949	0	0	0	0	0	0	0	0	0	0
14	0.1949	0	0	0	0	0	0	0	0	0	0
15	0.2728	0	0	0	0	0	0	0	0	0	0
16	0.4547	0	0	0	0	0	0	0	0	0	0
17	0	0	0.5434	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0

Table 3.3: Tf-idf table of the matrix

3.4 Singular Value Decomposition

The Singular-Value Decomposition, or SVD for short, is a matrix decomposition method for reducing a matrix to its constituent parts in order to make certain subsequent matrix calculations simpler. Singular value decomposition is a method of decomposing a matrix into three other matrices:

$$A=USV^T$$

Where:

- A is an m × n matrix
- U is an m × n orthogonal matrix
- S is an n × n diagonal matrix
- V is an n × n orthogonal matrix

Singular Value Decomposition is a matrix factorization method which is used in various domains of science and technology. Furthermore, due to recent great developments of machine learning, data mining and theoretical computer science, SVD has been found to be more and more important. It is not only a powerful tool and theory but also an art.

3.5 Cosine Similarity

The cosine similarity is a method of obtaining the closeness of two words. The cosine measure similarity is another similarity metric that depends on envisioning user preferences as points in space. Hold in mind the image of user preferences as points in an n-dimensional space. Now imagine two lines from the origin, or point (0, 0… 0), to each of these two points. When two users are similar, they'll have similar ratings, and so will be relatively close in space—at least, they'll be in roughly the same direction from the origin. The angle formed between these two lines will be relatively small. In contrast, when the two users are dissimilar, their points will be distant, and likely in different directions from the origin, forming a wide angle. This angle can be used as the basis for a similarity metric in the same way that the Euclidean distance was used to form a similarity metric. In this case, the cosine of the angle leads to a

similarity value. If you're rusty on trigonometry, all you need to remember to understand this is that the cosine value is always between –1 and 1: the cosine of a small angle is near 1, and the cosine of a large angle near 180 degrees is close to –1. This is good, because small angles should map to high similarity, near 1, and large angles should map to near –1. It is calculated by taking the dot product of the words by using the formula:

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

Equation 3.1: Cosine Similarity between variables A and B

Cosine similarity values of nouns and verbs (101) with their surrounding terms (464).

101x464 cell

	1	2	3	4	5	6	7	8	9	10	11	12
1	[]	'$5000'	'10'	'3'	'30'	'4'	'70'	'Agra'	'Baltic-Rep...	'David'	'GPS'	'God'
2	'Auden'	0	0	0	0	0	0	0	0	0	0	0
3	'Conference'	0	0	0	0	0	0	0	0	0	0	0
4	'Galileo'	0	0	0	0	0	0	0	0	0	0	0
5	'He'	0.1085	0	0	0	0.0381	0	0	0	0	0	0
6	'Hitler'	0	0	0	0	0	0	0	1	0	0	0
7	'I'	0	0	0	0	0	0	0	0	0	0	0
8	'It'	0	0	0	0	0	0	0	0	0	0	0
9	'Light'	0	0	0	0	0	0	0	0	0	0	0
10	'Lights'	0	0	0	0	0	0	0	0	0	0	0
11	'My'	0	0	0	0	0	0	0	0	0	0	0
12	'Put'	0	0	0	0	0	0	0	0	0	0	0
13	'Schoenberg'	0	0	0	0	0	0	0	0	0	0	0
14	'She'	0.1115	0.1605	0	0.1741	0	0	0	0	0	0.1306	0.0784
15	'Someone'	0	0	0	0	0	0	0	0	0	0	0
16	'Stock-prices'	0	0	0	0	0	0	0	0	0	0	0
17	'They'	0	0	0	0	0	0	0	0	0.4477	0	0
18	'We'	0	0	0	0	0	0	0.3444	0	0	0	0.2755
19	'Winner'	0	0	0	0	0	0	0	0	0	0	0
20	'You'	0	0	0	0	0	0	0	0	0	0	0
21	'accident'	0	0	0	0	0	0	0	0	0	0	0
22	'apple'	0	0	0	0	0	0	0	0	0	0	0

Table 3.4: Noun and Verb cosine similarity with surrounding terms

3.6 Methodologies

We have applied three different approaches to remove the ambiguity around the sense of the verb. All the three approaches are explained in the coming sections.

3.6.1 Method 1: Correlation Method

In this approach we are first calculating cosine similarity between noun and surrounding term and verb and surrounding term then we are using different correlation techniques to find the level of relatedness between noun and verb.

3.6.1.1 Process Flow

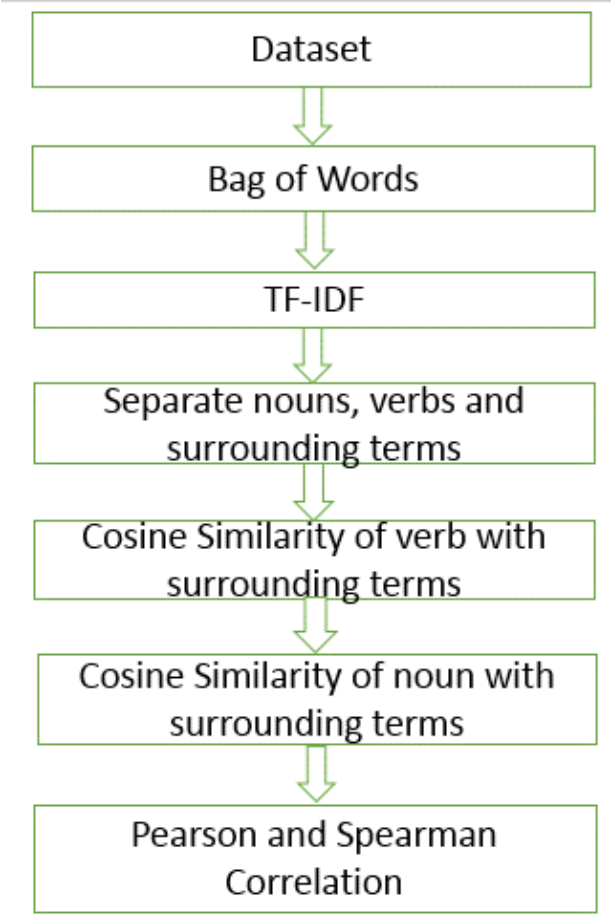

Figure 3.2: Process flow of correlation method

3.6.1.2 Bag of Words

The Bag of Words which we are having from the dataset of a total of 267 sentences contains unique 636 words. The words which do not affect the meaning of the noun or verb like “the”, “is”, “are”, etc. have been dropped from the database. The unique 636 words of the bag of word model is used to construct the *LSA matrix*. The dimension of the LSA matrix is *268*637* where 268 rows symbolizes the 267 documents present the dataset and 637 columns symbolizes the 636 unique words in dataset.

3.6.1.3 Term-Frequency

The cells of the LSA matrix are filled with the frequencies of each of 636 terms in the 267 sentences. After calculation of term frequencies, we will divide it by the total number of words present in that document. This gives us the normalized frequency of each word corresponding to all of the 267 sentences. For example, if document number 5 is “The machine runs on crude oil”, then here frequency of the word “machine” would be 1 and it will be divided by 6(number of words in sentence),giving us the normalized frequency for document number 5. The term frequency table is as shown below:

267x637 cell

	39	40	41	42	43	44	45	46	47	48	49
1	'The'	'Their'	'These'	'They'	'This'	'We'	'Winner'	'You'	'a'	'abdominal'	'accident'
2	1	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	0	0
10	1	0	0	0	0	0	0	0	1	0	0
11	1	0	0	0	0	0	0	0	0	0	0
12	1	0	0	0	0	0	0	0	0	0	0
13	1	0	0	0	0	0	0	0	0	0	0
14	1	0	0	0	0	0	0	0	0	0	0
15	1	0	0	0	0	0	0	0	0	0	0
16	1	0	0	0	0	0	0	0	0	0	0
17	1	0	0	0	0	0	0	0	0	0	0
18	0	0	1	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0

Table 3.5: Term frequency matrix of correlation method

3.6.1.4 Inverse Document Frequency

The formula for the idf is described as

IDF (word)= 1+log_e(total no. of documents/no. of documents with the term in it)

Using this formula we can calculate the idf values for each of the 636 words.

3.6.1.5 Term Frequency-Inverse document frequency

After calculation of the TF matrix whose dimension is 267*636 and the IDF matrix whose dimension is 636*636, we will multiply both the matrices to get the tf-idf matrix with dimensions 267*637.

3.6.1.6 Separation of Nouns, Verbs and Surrounding Terms

From a total of 636 unique words we will separate Nouns, Verbs and Surrounding Terms.

3.6.1.7 Cosine Similarity of Noun and Surrounding terms

After getting the tf-idf we can find the cosine similarity between a noun and its surrounding terms using the tf-idf matrix.

The formula for calculating the values of cosine similarity of noun with surrounding term is-

$$\text{Cos}(n,s) = \sum \text{tf-idf}(n_i) * \sum \text{tf-idf}(s_i) / (\sum n_i^{\,2})^{1/2} * (\sum s_i^2)^{1/2}$$

Where n = noun term.

s = surrounding term.

i = value ranging from 1 to 266.

After getting the cosine similarity values the terms with highest values are chosen.

Some values of the cosine similarity between the noun and its surrounding terms are shown below:

Surrounding Term	Cosine Similarity
'last'	0.450906457
'race-course'	0.358330308
'race'	0.31353902
'park'	0.253377791
'fast'	0.126740637
'very'	0.085139467

Table 3.6: Surrounding terms of Noun "Horse" with highest cosine similarity

Surrounding Term	Cosine Similarity
'crude-oil'	0.51883493
'electricity'	0.51883493
'smoothly'	0.432362441
'factory'	0.370596378
'properly'	0.370596378
'hours'	0.202143479
'many'	0.1962796
'very'	0.156540252

Table 3.7: Surrounding terms of Noun "Machine" with highest cosine similarity

Surrounding Term	Cosine Similarity
'veins'	0.607408665
'artery'	0.233618717
'body'	0.233618717
'heart'	0.233618717
'parts'	0.233618717
'all'	0.043189385

Table 3.8: Surrounding terms of Noun "Blood" with highest cosine similarity

3.6.1.8 Cosine Similarity of Verb and Surrounding Terms

After getting the tf-idf we can find the cosine similarity of the verb and its surrounding terms using the tf-idf matrix.

The formula for calculating the values of cosine similarity of verb with surrounding term is-

$$\text{Cos}(v,s) = \sum \text{tf-idf}(v_i) * \sum \text{tf-idf}(s_i) / \left(\sum v_i^{\,2}\right)^{1/2} * \left(\sum s_i^{2}\right)^{1/2}$$

Where v_i = i^{th} verb term

s_i = i^{th} surrounding term

i = value ranging from 1 to 266

After getting the cosine values of the verb with its surrounding terms the surrounding terms with highest cosine values are chosen.

The values for cosine similarity between the verb "run" and its surrounding terms is as shown below:

Surrounding Term	Cosine Similarity
'properly'	0.752576695
'race'	0.658504608
'hours'	0.410496379
'smoothly'	0.409250021
'many'	0.398588494
'factory'	0.350785733
'garden'	0.350785733
'race-course'	0.350785733
'tires'	0.350785733
'video'	0.350785733
'tapes'	0.335177515
'forest'	0.303001315
'guaranteed'	0.297935569
'crude-oil'	0.274597701

Table 3.9: Surrounding terms of Verb "Run" with highest cosine similarity

3.6.1.9 Correlation and Coefficient of Correlation

Correlation is a statistical technique that can show whether and how strongly pairs of random variables are related. For e.g., the height and weight of the person are correlated. The correlation coefficient is a statistical measure that calculates the strength of the relationship between the relative movements of two variables. The values range between -1.0 and 1.0.

3.6.1.10 Pearson Correlation

The Pearson product-moment correlation coefficient (or Pearson correlation coefficient, for short) is a measure of the strength of a linear association between two variables and is denoted by *r*. Basically, a Pearson product-moment correlation attempts to draw a line of best fit through the data of two variables, and the Pearson correlation coefficient, *r*, indicates how far away all these data points are to this line

of best fit (i.e., how well the data points fit this new model/line of best fit). The Pearson correlation coefficient, *r*, can take a range of values from +1 to -1.

- A value of 0 indicates that there is no association between the two variables.
- A value greater than 0 indicates a positive association; that is, as the value of one variable increases, so does the value of the other variable.
- A value less than 0 indicates a negative association; that is, as the value of one variable increases, the value of the other variable decreases.

This is shown in the diagram below:

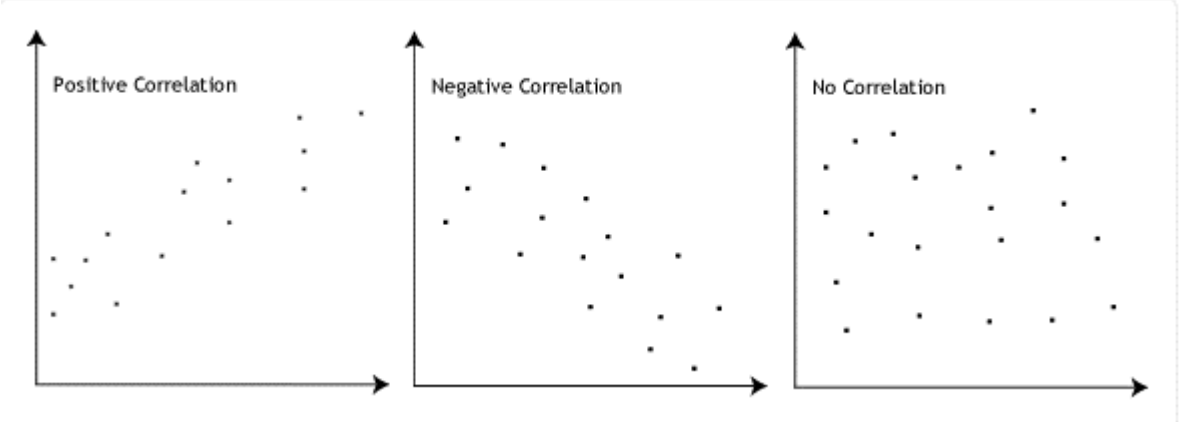

Figure 3.3: Graph of Pearson correlation Coefficient

Pearson Correlation Coefficient

The formula for Pearson correlation coefficient is

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[\, n\sum x^2 - (\sum x)^2 \,][\, n\sum y^2 - (\sum y)^2 \,]}}$$

Equation 3.2: Coefficient of Pearson Correlation

Where n= total no. of data entries and x and y are the data entries in the respective columns.

Determining strength of association based on the Pearson correlation coefficient

The stronger the association of the two variables, the closer the Pearson correlation coefficient, *r*, will be to either +1 or -1 depending on whether the relationship is positive or negative, respectively. Achieving a value of +1 or -1 means that all your data points are included on the line of best fit – there are no data points that show any variation away from this line. Values for *r* between +1 and -1 (for example, *r* = 0.8 or -0.4) indicate that there is variation around the line of best fit. The closer the value of *r* to 0 the greater the variation around the line of best fit. Different relationships and their correlation coefficients are shown in the diagram below:

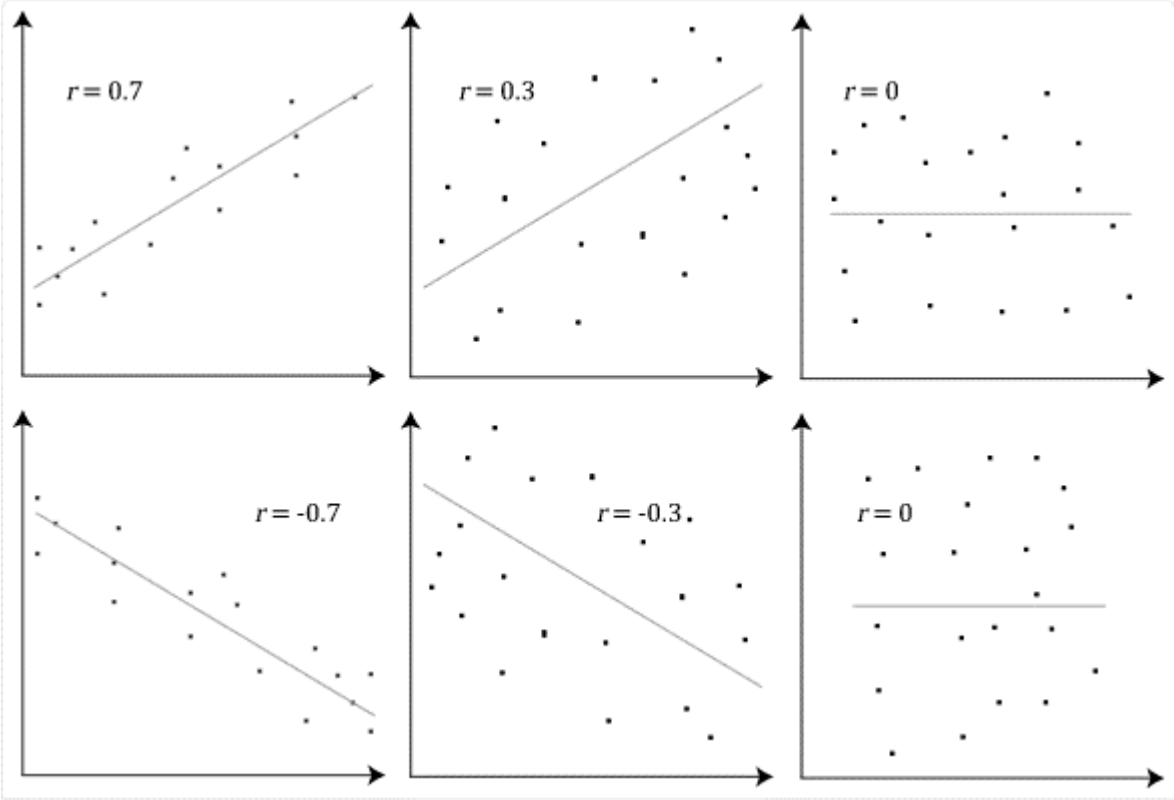

Figure 3.4: Graph of Values of Pearson Correlation Coefficient

After calculation of the cosine similarities between:

- Noun and its surrounding terms
- Verb and its surrounding terms

We calculate the Pearson correlation coefficient for these two cosine similarities.

Surrounding Terms	Horse(Cosine similarity)	Run (Cosine similarity)
Last	0.45	0.2468
race-course	0.358	0.3508
race	0.313	0.658
park	0.253	0.248
fast	0.1267	0.2662
very	0.08513	0.1788
Pearson Correlation=0.3325		

Table 3.10: Pearson Correlation Values between “Horse” and “Run”

Surrounding Terms	Blood(Cosine Similarity)	Run(Cosine Similarity)
veins	0.6074	0.2746
body	0.2336	0.1056
heart	0.2336	0.1056
parts	0.2336	0.1056
all	0.04	0.0195
Pearson Correlation=0.9162		

Table 3.11: Pearson Correlation Values between “Blood” and “Run”

3.6.1.11 Spearman Correlation

The Spearman's rank-order correlation is the nonparametric version of the Pearson product-moment correlation. Spearman's correlation coefficient, (ρ, also signified by r_s) measures the strength and direction of association between two ranked variables. Spearman's correlation measures the strength and direction of monotonic association between two variables. Monotonicity is "less restrictive" than that of a linear relationship. For example, the middle image above shows a relationship that is monotonic, but not linear.

A monotonic relationship is not strictly an assumption of Spearman's correlation. That is, you can run a Spearman's correlation on a non-monotonic relationship to determine if there is a monotonic component to the association.

The Spearman correlation coefficient, *r*, can take a range of values from +1 to -1. The value of spearman correlation coefficient tells us about the relation between two variables as follows –

- 1 for perfect positive monotonic relationship
- -1 for perfect negative monotonic relationship
- 0 for no correlation

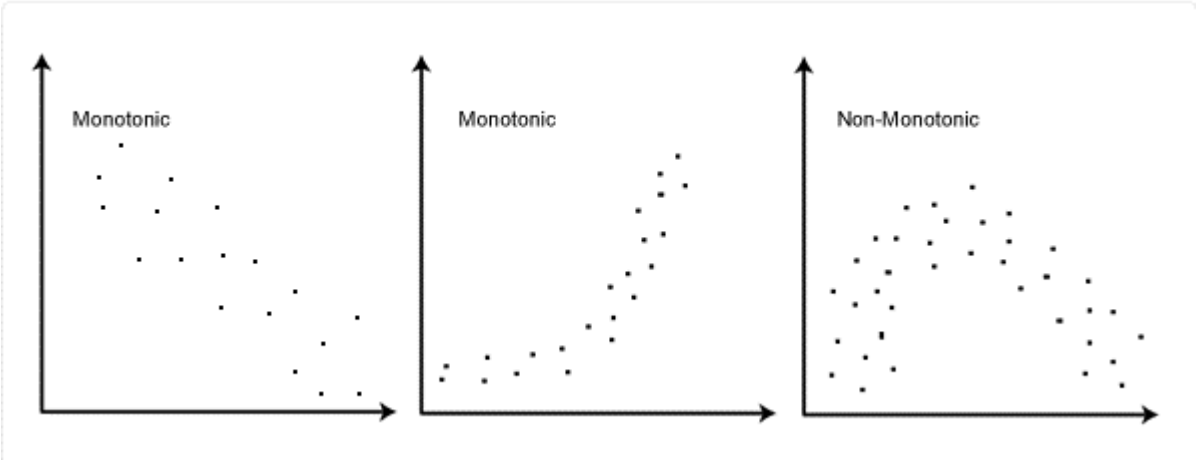

Figure 3.5: Graph of values of Spearman Coefficient Correlation values

How to rank data?

In some cases your data might already be ranked, but often you will find that you need to rank the data yourself. Thankfully, ranking data is not a difficult task and is easily achieved by working through your data in a table. The score with the highest value should be labelled "1" and the lowest score should be labelled "10" (if your data set has more than 10 cases then the lowest score will be how many cases you have). When you have two identical values in the data (called a "tie"), you need to take the average of the ranks that they would have otherwise occupied.

Spearman Correlation Coefficient

The formula for spearman correlation coefficient is

$$r_R = 1 - \frac{6\Sigma_i {d_i}^2}{n(n^2-1)}$$

Equation 3.3: Spearman Correlation formula

Where n= total no. of data entry pairs and d_i is the difference of respective rank of columns for the i^{th} element.

Surrounding Terms	Horse(Cosine similarity)	Run (Cosine similarity)
Last	0.45	0.2468
race-course	0.358	0.3508
race	0.313	0.658
park	0.253	0.248
fast	0.1267	0.2662
very	0.08513	0.1788
Spearman Correlation=0.31429		

Table 3.12: Spearman Correlation values between noun "Horse" and verb "Run"

Surrounding Terms	Blood(Cosine Similarity)	Run(Cosine Similarity)
veins	0.6074	0.2746
body	0.2336	0.1056
heart	0.2336	0.1056
parts	0.2336	0.1056
all	0.04	0.0195
Spearman Correlation=0.68825		

Table 3.13: Spearman Correlation values between noun "Blood" and verb "Run"

Surrounding Term	John	Broke
'angrily'	0.398473634	0.260714585
'batsman'	0.166030681	0.108631077
'cigarette'	0.38894454	0.254479859
'cigarettes'	0.284624024	0.186224703
'cycle'	0.199236817	0.130357292
'finally'	0.122369426	0.080064253
'glasses'	0.249046021	0.162946615
'highest'	0.166030681	0.108631077
Pearson Coefficient Value=0.9929		
Spearman Coefficient Value=0.98197		

Table 3.14: Spearman and Pearson Correlation values between noun "John" and verb "Broke"

3.6.2 Method 2: Classifier

Classification is the task of assigning a set of predefined categories to free-text. Text classifiers can be used to organize, structure, and categorize pretty much anything. For example, new articles can be organized by topics, support tickets can be organized by urgency, and chat conversations can be organized by language, brand mentions can be organized by sentiment, and so on.

As an example, take a look at the following text below:

"*The horse is running in the race-course."*

A classifier can take this text as an input, analyse its content, and then and automatically assign relevant meanings, such as here the meaning of running is in the context of moving.As another example, take a look at the following text below:

"*The machine is running."*

By analyzing the context in which the word running is used we can say that in this sentence the meaning of the verb is defined as working i.e. the machine is working.

3.6.2.1 Process Flow

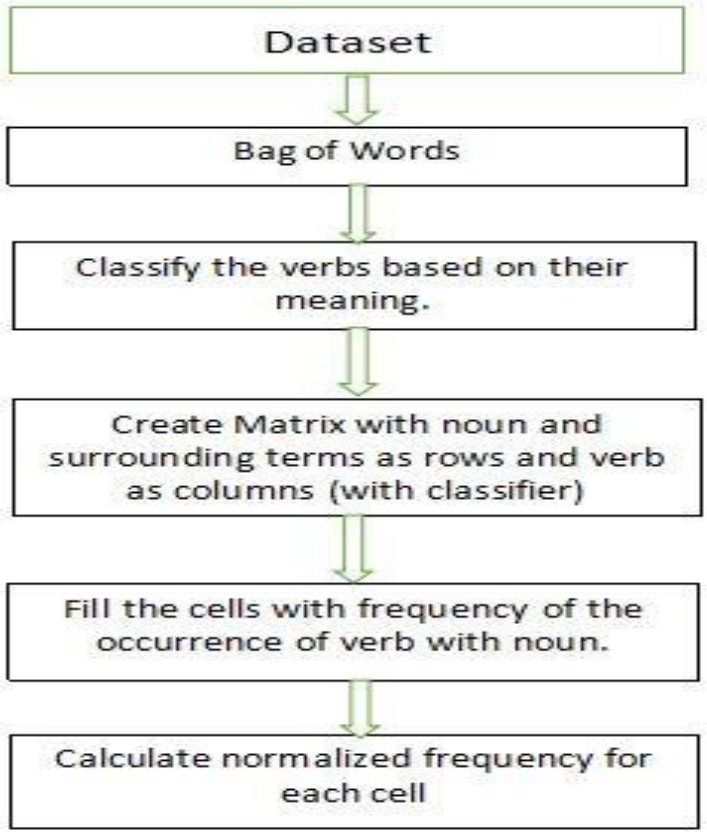

Figure 3.6: Process flow for Classifier method

This method as experiment was applied to the verb run and its different meaning in the context of its surrounding terms. The verb run was classified into 9 classes depending on the meaning of the verb when we look at its surrounding terms-

- Moving
- Working
- Flowing
- Diffusion
- Executing
- Continue

- Playing
- Sailing
- occurring

The matrix is constructed considering the 9 meanings of the ambiguous verb run. The rows are defined by the noun and the surrounding terms and the columns are the classifications of the verb run. The verb run is here represented in 4 forms;

- Run
- Running
- Ran
- Runs

The matrix constructed is as shown below:

	Verb Run																		
	Moving				Flowing				Working				Diffussion				Executing		
	runs	run	ran	running	runs	run	ran	running	runs	run	ran	running	runs	run	ran	running	runs	run	ran
Horse	2	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
fast	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
last	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
park	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
race-cours	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
race	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rabbit	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
garden	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
house	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
kangaroo	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
baby	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pouch	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
forest	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
machine	0	0	0	0	0	0	0	0	2	0	1	2	0	0	0	0	0	0	0
smoothly	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
electricity	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
factory	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
properly	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
hours	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
crude-oil	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
colors	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
dyes	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
blood	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
veins	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
heart	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
body	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.15: Classifier matrix tf-idf values

After construction of the matrix the cells are filled with frequencies with which they are occurring with the respective noun and verb.

3.6.3 Method 3: Document Matrix

Term-document matrix is an important representation for text analytics. Each row of the matrix is a document vector, with one column for every term in the entire corpus. Naturally, some documents may not contain a given term, so this matrix is sparse. The value in each cell of the matrix is the term frequency. (This value is often a weighted term frequency, typically using tf-idf -- term frequency-inverse document frequency.)With the term-document matrix, you can compute the similarity of documents. In this, we will make two document matrices. The first one will be the Noun-Document tf-idf matrix where all the nouns (total=154) would be the rows and 267 documents would be the columns. The second matrix would be the Verb-Document tf-idf matrix where the rows will be the verbs (total =32) be and 267 documents would be the columns.

3.6.3.1 Process Flow

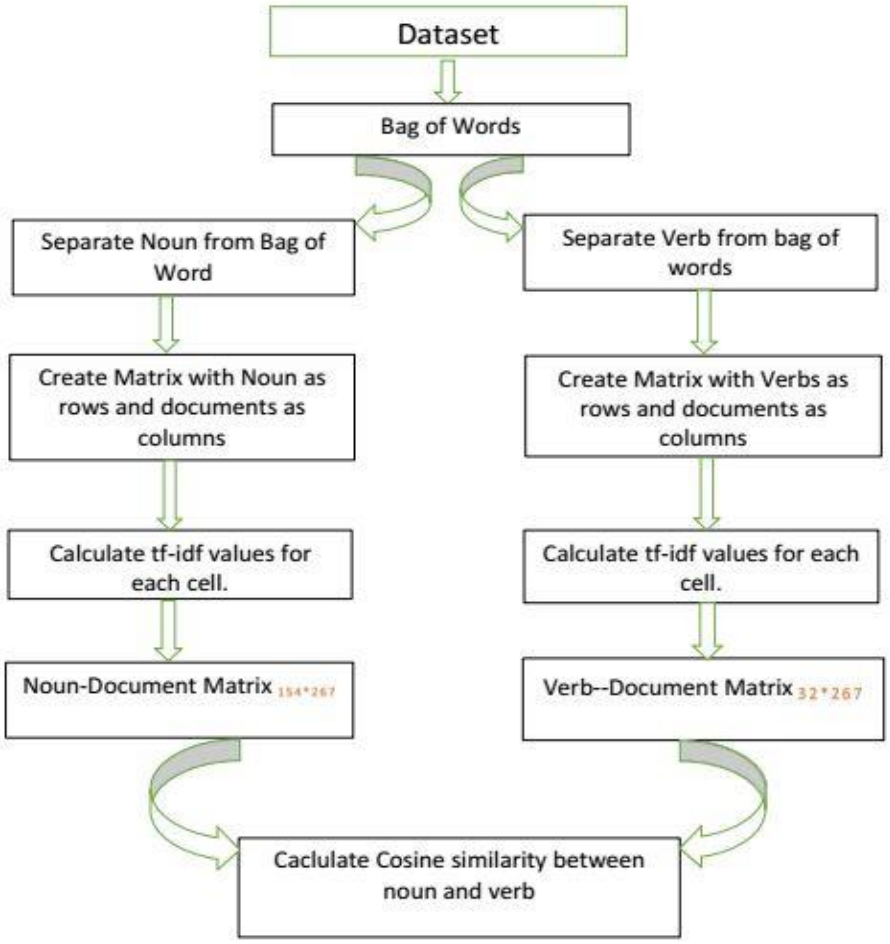

Figure 3.7: Process flow for document matrix method

3.6.3.2 Noun-Document tf-idf Matrix

The cells of the matrix would be filled with the tf-idf values. Each cell would contain the number of the appearance of the noun in each sentence divided by the total number of words in the sentences (known as normalized frequency).

After calculation of the idf values of each noun using the formula:

IDF (word) = 1+log$_e$ (total no. of documents/no. of documents with the term in it)

Each row of the matrix would be multiplied with respective idf values.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
horse	0.995562149	1.244452686	0.711115821	0.711115821	0.622226343	0	0	0	0	0	0	0	0	0	0	0	
rabbit	0	0	0	0	0	0.78409091	1.097727274	0.686079546	0	0	0	0	0	0	0	0	
kangaroo	0	0	0	0	0	0	0	0	0.6549	0.842014	0	0	0	0	0	0	
machine	0	0	0	0	0	0	0	0	0	0	0.829635	0.995562	0.711116	0.711116	0.995562	0	
colors	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.9647	0.
dyes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.73
blood	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
artery	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
bus	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
police	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
computer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
film	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ship	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
apple	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
draft	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
car	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
church	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
park	0	0	0.842014497	0	0	0	0	0	0	0	0	0	0	0	0	0	
supermar	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
roof	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
lawyers	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
prisoner	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
life	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
children	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
money	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
party	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
God	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ice	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 3.16: Tf-idf matrix of Noun for document matrix method

3.6.3.3 Verb-Document tf-idf Matrix

The cells of the matrix would be filled with the tf-idf values. Each cell would contain the number of the appearance of the verb in each sentence divided by the total number of words in the sentences (known as normalized frequency).

After calculation of the idf values of each verb using the formula:

IDF (word)= 1+loge(total no. of documents/no. of documents with the term in it)

Each row of the matrix would be multiplied with respective idf values.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
run	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.829545457	0.609848486
runs	0.739375	0.924219	0	0	0	0	0.739375	0	0.410764	0	0	0.739375	0	0	0.739375	0	0
ran	0	0	0	0	0.736763	0	0	0	0	0	0	0	0	0.842014	0	0	0
running	0	0	0.643972	0.643972	0	0.643972	0	0.563476	0	0.643972	0.751301	0	0.643972	0	0	0	0
give	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
gives	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
gave	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
broke	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
breaking	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
called	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
call	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
calls	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
know	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
knows	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
knew	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
known	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
put	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
took	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
taken	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
taking	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
takes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
take	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
made	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
make	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
making	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
makes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
drew	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
drawn	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.17: Tf-idf matrix of Verb for document matrix method.

3.6.3.4 Singular Value Decomposition (SVD)

We will find the svd of both the matrices Noun-Document tf-idf matrix as well as Verb-Document matrix.

	1	2	3	4	5	6	7	8	9	10	11	12	13	1
horse	0.995562149	1.244452686	0.711115821	0.711115821	0.622226343	4.06887E-17	5.69642E-17	3.56026E-17	5.51718E-33	5.57995E-33	5.10982E-49	6.13178E-49	4.37985E-49	4.37985E-4
rabbit	-2.75249E-17	9.16132E-17	-3.21384E-17	-1.96607E-17	8.95317E-18	0.78409091	1.097727274	0.686079546	3.30287E-17	1.32982E-17	-3.47418E-17	-4.16901E-17	-2.97787E-17	-2.97787E-1
kangaroo	-5.65628E-17	1.58759E-17	3.19936E-18	7.7801E-17	-4.77931E-17	-2.98748E-18	6.7485E-17	4.10521E-17	0.654900164	0.842014497	-1.07328E-17	-1.28793E-17	-9.1995E-18	-9.1995E-1
machine	-1.58151E-16	-1.99414E-17	-1.00928E-17	2.186E-16	1.23002E-16	1.38661E-16	1.94217E-16	4.26068E-16	-2.95335E-16	2.81291E-17	0.829635124	0.995562149	0.711115821	0.7111158
colors	-1.83391E-16	6.43721E-17	-7.95489E-18	-9.39219E-17	-3.15429E-16	-8.50309E-17	5.90533E-17	-1.06677E-16	4.03892E-16	1.72932E-16	1.09332E-16	4.06317E-18	-7.59649E-17	-4.756E-1
dyes	4.35512E-17	1.29974E-17	-5.28923E-18	-1.1888E-16	6.16997E-17	6.95752E-17	1.06848E-16	-7.95636E-17	4.22414E-17	-2.51623E-16	2.34784E-17	-1.48106E-17	-1.4981E-17	-1.42246E-1
blood	-3.06873E-17	1.29363E-17	5.2679E-19	-9.12457E-17	6.9179E-17	-8.56622E-17	-3.67209E-17	-8.46568E-17	2.38407E-16	-1.10966E-16	-3.67283E-16	-2.39522E-16	-1.68363E-16	-1.64571E-1
artery	-1.12831E-16	2.658E-17	4.40729E-19	1.38456E-16	-3.17898E-17	7.55557E-17	7.02632E-17	-2.15572E-16	2.02173E-17	1.19009E-17	5.05557E-16	5.91688E-16	1.53734E-16	1.42335E-1
bus	1.08866E-16	5.16926E-17	1.18562E-17	3.77924E-17	-3.41027E-16	3.41206E-17	-1.24324E-16	1.1034E-18	1.57616E-16	-1.92643E-16	1.50686E-16	-1.63608E-16	-1.01339E-16	-7.62104E-1
police	6.4625E-17	-3.87828E-17	4.89422E-18	1.72147E-16	-1.87224E-16	1.52506E-16	-1.34681E-16	2.92584E-17	1.43797E-16	-2.39247E-17	-5.6783E-16	6.07276E-16	8.92685E-16	-3.58664E-1
computer	7.94827E-17	4.22717E-18	-4.76947E-18	2.59264E-16	1.06715E-16	3.35936E-16	-5.26593E-17	1.29638E-16	1.78094E-15	-1.37117E-15	3.74E-17	5.21148E-17	1.06118E-16	-9.60481E-1
film	4.57082E-17	-3.37629E-17	1.80507E-18	1.23702E-16	-1.28872E-16	2.87296E-17	3.71742E-17	-8.70987E-18	4.76499E-17	2.79099E-17	1.64077E-17	-1.23553E-16	1.71984E-16	-6.33249E-1
ship	4.12848E-17	4.20678E-17	-5.09144E-18	6.32256E-17	-2.24063E-16	1.30447E-16	-2.41415E-16	2.15178E-16	-3.98292E-17	-1.72948E-16	-5.17779E-16	5.33988E-16	7.15308E-16	-6.09584E-1
apple	-7.02769E-17	-1.09049E-16	9.60078E-19	-1.15939E-16	4.85237E-16	-1.15244E-16	2.17877E-16	-2.11357E-16	-3.40528E-17	7.04114E-18	-3.35256E-17	-2.0831E-16	1.07824E-16	2.58668E-1
draft	5.27051E-17	-6.47324E-17	4.04191E-19	6.11051E-17	-1.92143E-17	3.67214E-17	-4.024E-17	2.95703E-17	-1.15421E-17	3.66883E-18	-5.71289E-17	2.6696E-16	-3.90937E-16	1.19424E-1
car	4.99248E-17	-9.93326E-17	-1.26113E-18	-1.72483E-16	3.22948E-16	-5.54832E-18	9.37873E-17	-1.24146E-16	2.93283E-18	-5.33948E-17	9.97251E-17	-9.27501E-17	-4.17274E-17	8.0158E-1
church	-1.29443E-17	-2.5492E-17	1.63822E-19	9.22517E-17	-3.2982E-17	7.05337E-17	4.39058E-18	-7.06911E-17	1.2248E-16	-1.66518E-16	6.3129E-17	8.97029E-17	5.90425E-17	7.68838E-1
park	1.17961E-16	1.38778E-17	0.842014497	-3.46945E-17	-6.93889E-18	-3.48206E-16	-4.87489E-16	-3.04681E-16	2.54949E-31	1.15783E-30	5.94383E-17	7.1326E-17	5.09471E-17	5.09471E-1
supermar	1.94289E-16	-1.38778E-17	5.55112E-17	3.46945E-17	4.16334E-17	4.48182E-16	6.27454E-16	3.92159E-16	-3.20504E-17	-3.95627E-17	-2.48789E-17	-2.98547E-17	-2.13248E-17	-2.13248E-1
roof	-6.92683E-18	2.38362E-17	1.84756E-18	3.38399E-17	-6.35924E-17	8.47732E-17	-7.9437E-17	-4.18815E-17	-3.18022E-17	5.28408E-18	-1.0623E-16	2.21645E-16	3.76674E-17	2.89031E-1
lawyers	8.40367E-18	5.05718E-17	-3.61412E-19	9.99364E-17	-2.30127E-16	2.47244E-17	-1.02644E-16	1.38139E-16	5.07528E-18	-6.65747E-19	-3.68075E-16	4.06683E-16	3.27381E-16	-6.10887E-1
prisoner	-2.07235E-17	-5.6877E-17	-3.61412E-19	-3.51491E-17	1.85758E-16	-8.95601E-17	1.79531E-16	-1.82729E-16	5.07528E-18	-6.65747E-19	4.4006E-16	-5.11223E-16	-1.16372E-16	3.46087E-1
life	1.16405E-17	3.22524E-18	3.65686E-18	5.41701E-17	-1.51383E-16	-9.1324E-17	4.69414E-17	4.42115E-17	1.46377E-15	-8.81557E-16	-1.66883E-17	1.18551E-17	9.31235E-17	-6.9466E-1
children	3.17939E-17	1.19654E-17	7.55426E-18	-4.45742E-17	-1.65977E-17	2.12002E-18	4.99888E-17	1.61156E-16	5.5984E-15	-3.25942E-15	-1.05271E-17	-7.6863E-17	-4.24906E-16	-3.42345E-1
money	2.0357E-17	1.06107E-17	3.15277E-20	4.25473E-17	-1.04989E-16	-3.15462E-17	1.5699E-17	2.06553E-17	-9.5369E-18	-1.65248E-18	1.12416E-16	-6.44279E-18	-3.28329E-16	9.48927E-1
party	5.75648E-19	-3.44001E-19	-4.70989E-23	-1.79843E-17	2.03211E-17	6.66813E-18	6.94924E-18	-1.87394E-17	-1.76888E-22	-9.12165E-22	5.38196E-17	-9.39931E-18	-9.44175E-17	3.41451E-1
God	4.16781E-18	2.13817E-18	-1.01895E-32	-2.65689E-17	1.94196E-17	-6.13201E-17	2.77672E-17	2.56526E-17	5.70219E-31	2.79475E-31	2.99842E-17	-3.80951E-17	-1.37914E-16	1.72699E-1
ice	2.38023E-17	1.31123E-17	3.26376E-19	-1.18583E-16	7.34703E-17	6.48121E-17	-1.33519E-16	1.26845E-16	-5.92968E-18	9.24529E-19	-3.44692E-16	2.16888E-16	4.15505E-16	-8.1838E-1

Table 3.18: Tf-idf matrix of Verb for document matrix method after SVD calculation

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
run	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.829545457	0.609848486	0	0
runs	0.739375	0.924219	-4.2E-20	-4.2E-20	8.4E-18	-4.2E-20	0.739375	-3.7E-20	0.410764	-4.2E-20	-4.9E-20	0.739375	-4.2E-20	9.6E-18	0.739375	-7.4734E-17	7.22224E-35	0.739375	0.284375
ran	-7.3E-18	-9.1E-18	5.02E-17	5.02E-17	0.736763	5.02E-17	-7.3E-18	4.39E-17	-4.1E-18	5.02E-17	5.85E-17	-7.3E-18	5.02E-17	0.842014	-7.3E-18	6.69749E-34	-1.01177E-34	-7.3E-18	-2.8E-18
running	8.34E-21	1.04E-20	0.643972	0.643972	-1.4E-17	0.643972	8.34E-21	0.563476	4.64E-21	0.643972	0.751301	8.34E-21	0.643972	-1.6E-17	8.34E-21	1.18026E-49	4.41953E-35	8.34E-21	3.21E-21
give	6.16E-19	7.7E-19	4E-17	4E-17	-8.5E-17	4E-17	6.16E-19	3.5E-17	3.42E-19	4E-17	4.67E-17	6.16E-19	4E-17	-9.7E-17	6.16E-19	-8.42633E-35	-2.54796E-35	6.16E-19	2.37E-19
gives	-3.3E-19	-4.1E-19	-2.7E-17	-2.7E-17	-8.8E-17	-2.7E-17	-3.3E-19	-2.3E-17	-1.8E-19	-2.7E-17	-3.1E-17	-3.3E-19	-2.7E-17	-1E-16	-3.3E-19	4.53092E-35	1.13273E-35	-3.3E-19	-1.3E-19
gave	-8.5E-19	-1.1E-18	-4.3E-17	-4.3E-17	1.25E-17	-4.3E-17	-8.5E-19	-3.7E-17	-4.7E-19	-4.3E-17	-5E-17	-8.5E-19	-4.3E-17	1.43E-17	-8.5E-19	1.15636E-34	2.89089E-35	-8.5E-19	-3.3E-19
broke	1.39E-17	2.78E-17	6.2E-17	6.2E-17	-7.2E-15	6.2E-17	4.42E-33	5.42E-17	1.39E-17	6.2E-17	7.23E-17	4.42E-33	6.2E-17	-8.3E-15	4.42E-33	3.20868E-19	3.08585E-19	4.42E-33	6.94E-18
breaking	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
called	4.98E-36	-2.7E-35	1.11E-35	1.01E-34	0	1.1E-35	4.94E-36	-8.6E-36	8.11E-36	-2.9E-35	4.89E-36	4.94E-36	1.1E-35	-1.2E-51	4.94E-36	-3.07994E-52	-1.53997E-52	4.94E-36	2.59E-37
call	-1.5E-35	-7.4E-35	-4E-35	-1.5E-34	0	-4E-35	-1.5E-35	-3.4E-35	-8.4E-36	3.96E-34	-4.6E-35	-1.5E-35	-4E-35	6.05E-51	-1.5E-35	1.51233E-51	7.56164E-52	-1.5E-35	-5.8E-36
calls	3.87E-51	1.89E-50	1.01E-50	3.9E-50	0	1.01E-50	3.87E-51	8.69E-51	2.15E-51	-1E-49	1.18E-50	3.87E-51	1.01E-50	-1.5E-66	3.87E-51	-3.84775E-67	-1.92387E-67	3.87E-51	1.49E-51
know	1.03E-35	5.02E-35	2.68E-35	1.03E-34	0	2.68E-35	1.03E-35	2.31E-35	5.7E-36	-2.7E-34	3.13E-35	1.03E-35	2.68E-35	-4.1E-51	1.03E-35	-1.02105E-51	-5.10523E-52	1.03E-35	3.95E-36
knows	4.13E-36	2.02E-35	1.08E-35	4.16E-35	0	1.08E-35	4.13E-36	9.29E-36	2.3E-36	-1.1E-34	1.26E-35	4.13E-36	1.08E-35	-1.6E-51	4.13E-36	-4.11225E-52	-2.05612E-52	4.13E-36	1.59E-36
knew	6.21E-35	3.04E-34	1.62E-34	6.25E-34	0	1.62E-34	6.21E-35	1.39E-34	3.45E-35	-1.6E-33	1.89E-34	6.21E-35	1.62E-34	-2.5E-50	6.21E-35	-6.17506E-51	-3.08753E-51	6.21E-35	2.39E-35
known	7.35E-51	3.6E-50	1.92E-50	7.4E-50	0	1.92E-50	7.35E-51	1.65E-50	4.08E-51	-1.9E-49	2.24E-50	7.35E-51	1.92E-50	-2.9E-66	7.35E-51	-7.31224E-67	-3.65612E-67	7.35E-51	2.83E-51
put	2.02E-67	9.86E-67	5.27E-67	2.03E-66	0	5.27E-67	2.02E-67	4.53E-67	1.12E-67	-5.3E-66	6.15E-67	2.02E-67	5.27E-67	-8E-83	2.02E-67	-2.00547E-83	-1.00274E-83	2.02E-67	7.76E-68
took	5.39E-53	2.63E-52	1.41E-52	5.42E-52	0	1.41E-52	5.39E-53	1.21E-52	2.99E-53	-1.4E-51	1.64E-52	5.39E-53	1.41E-52	-2.1E-68	5.39E-53	-5.35601E-69	-2.67801E-69	5.39E-53	2.07E-53
taken	1.23E-51	6.01E-51	3.21E-51	1.24E-50	0	3.21E-51	1.23E-51	2.76E-51	6.82E-52	-3.2E-50	3.75E-51	1.23E-51	3.21E-51	-4.9E-67	1.23E-51	-1.22167E-67	-6.10837E-68	1.23E-51	4.72E-52
taking	-1.1E-50	-5.5E-50	-2.9E-50	-1.1E-49	0	-2.9E-50	-1.1E-50	-2.5E-50	-6.2E-51	2.93E-49	-3.4E-50	-1.1E-50	-2.9E-50	4.47E-66	-1.1E-50	1.11696E-66	5.58479E-67	-1.1E-50	-4.3E-51
takes	-6.6E-67	-3.2E-66	-1.7E-66	-6.7E-66	0	-1.7E-66	-6.6E-67	-1.5E-66	-3.7E-67	1.73E-65	-2E-66	-6.6E-67	-1.7E-66	2.64E-82	-6.6E-67	6.58789E-83	3.29394E-83	-6.6E-67	-2.5E-67
take	3.01E-35	9.63E-35	4.81E-35	2.89E-34	0	4.81E-35	3.01E-35	6.02E-35	1.5E-35	-5.8E-34	7.22E-35	3.01E-35	4.81E-35	-1.1E-50	3.01E-35	-2.67276E-51	-1.33638E-51	3.01E-35	6.02E-36
made	9.71E-17	1.67E-16	9.39E-19	3.61E-18	7.35E-15	9.39E-19	9.71E-17	-1.1E-17	4.86E-17	9.39E-19	1.1E-18	9.71E-17	9.39E-19	8.4E-15	9.71E-17	3.28692E-19	9.09001E-19	9.71E-17	4.86E-17
make	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
making	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
makes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
drew	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
drawn	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.19: Tf-idf matrix of Verb for document matrix method after SVD calculation

3.6.3.5 Cosine similarity

We will calculate cosine similarity between Noun and Verb.

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

Equation 4. Cosine similarity between variables A and B.

Here A and B are the respective rows of noun and verb matrices and

n takes the value from 1 to 267.

	horse	rabbit	kangaroo	machine	colors	dyes	blood	artery	bus	police	computer	film	ship	apple
run	-2.8851E-49	1.75079E-18	5.10882E-16	2.60951E-17	0.942154994	0.297935569	3.60831E-16	6.55552E-16	-2.02791E-16	4.51609E-17	-1.10506E-16	2.6172E-17	-2.65375E-18	1.05996
runs	0.352998086	0.19917066	0.093659267	0.284941758	-4.99113E-17	2.09248E-16	0.191466547	0.1056145	0.090859534	-8.44624E-17	0.223449341	0.228831418	0.173741765	0.22883
ran	0.206466889	2.53763E-17	-2.81996E-18	0.278902197	-1.17579E-16	4.0885E-17	-4.78775E-17	1.67574E-16	-2.44452E-16	-1.36883E-16	-2.15073E-17	-1.2615E-16	-1.53627E-16	3.03335
running	0.251394319	0.320878649	0.276893496	0.30694426	-9.4394E-18	-1.74179E-16	-1.86001E-16	5.40863E-16	-1.6163E-17	1.29891E-16	-1.40968E-16	1.16834E-16	9.80878E-17	-1.13182
give	2.57267E-18	3.74422E-17	-1.5849E-17	-1.38691E-17	-2.11317E-17	2.95579E-17	-2.21845E-17	5.81414E-18	-2.95814E-17	-1.30997E-16	-3.33616E-17	-9.3914E-18	-2.68871E-17	-3.98496
gives	-5.00576E-17	-2.6067E-17	-2.23934E-17	-6.47469E-17	3.13667E-32	-2.97498E-32	-2.45801E-19	-1.35586E-19	6.92355E-18	-3.31546E-18	-5.49363E-17	-2.9377E-19	-5.40656E-17	-2.05386
gave	-1.82355E-17	-3.32869E-17	-2.78438E-17	-2.7269E-17	-6.62415E-18	1.04303E-17	-1.40239E-17	6.31461E-17	1.38656E-16	3.81583E-19	-1.56831E-16	1.14574E-16	6.55255E-17	-1.47908
broke	-7.74274E-16	2.33332E-17	-4.93683E-18	-1.07174E-15	-1.00818E-17	6.52895E-18	-8.58212E-18	1.84859E-17	0.06340819	-5.47234E-17	-7.35288E-17	-1.8268E-16	-5.52725E-17	-1.89989
breaking	3.86651E-81	-3.01623E-33	1.81169E-33	4.33818E-33	6.85814E-33	8.29373E-33	5.61166E-33	1.97216E-32	8.05462E-32	8.15898E-17	-9.59676E-18	-1.1619E-30	-1.19624E-17	-1.24874
called	1.05865E-35	5.48341E-18	2.10951E-17	4.08595E-17	-1.63302E-17	4.04051E-17	-1.77951E-17	1.38566E-16	-1.90334E-16	0.248619835	-2.06413E-16	1.0454E-15	1.65596E-16	-1.16632
call	-8.5556E-35	-1.4903E-17	1.9257E-17	-7.51971E-18	-5.74668E-18	6.17227E-17	2.29903E-17	-1.56801E-17	-2.61151E-17	1.1129E-16	-1.56416E-17	1.365E-16	0.335349456	-3.84647
calls	2.25922E-50	1.95942E-33	1.96169E-32	5.86981E-33	-6.49775E-33	-9.63385E-33	-1.21027E-32	-1.86042E-33	-7.14288E-33	2.04232E-18	-3.67511E-17	-1.2291E-31	-4.36022E-17	1.48128
know	3.18988E-35	-3.67783E-33	1.51543E-18	1.89967E-19	-1.01906E-18	2.58898E-18	-1.34928E-18	-1.51196E-18	1.77708E-18	-3.12691E-17	1.28986E-17	-1.6847E-17	1.10282E-16	-3.39028
knows	1.83649E-35	1.09423E-18	2.42559E-18	-2.49734E-18	3.83692E-18	1.22623E-18	-2.79208E-18	-2.02032E-17	5.954E-17	5.87091E-17	-2.32701E-17	-4.6129E-16	-5.26974E-17	-1.76379
knew	5.10982E-34	-4.49899E-33	-2.86898E-33	-5.35024E-33	-1.89841E-33	2.19777E-32	6.49508E-33	-2.00703E-32	-7.28835E-18	8.22481E-18	-1.80255E-17	2.75454E-32	1.11434E-16	-1.99463
known	4.08874E-50	-1.16628E-33	7.9494E-33	3.19214E-33	-2.37957E-34	5.91346E-33	-3.16024E-33	-4.67455E-34	-2.55647E-17	9.31789E-18	-6.93441E-18	-8.9048E-32	3.64092E-18	1.51361
put	8.88572E-67	4.2939E-34	4.77439E-34	1.86217E-34	3.05674E-34	1.77832E-33	4.25172E-34	-1.83834E-33	3.68131E-18	-3.04415E-18	6.87335E-18	-4.7596E-32	-5.2899E-18	-1.70522
took	3.60533E-52	-3.17788E-19	2.70995E-18	9.80902E-19	-2.14552E-18	9.88697E-19	-7.18125E-19	6.69686E-18	-2.13005E-17	1.03432E-17	-2.28939E-16	-8.8501E-17	-4.173E-17	-3.77073
taken	1.21025E-50	-2.86781E-33	9.42887E-33	9.81106E-33	2.37222E-33	-8.52969E-33	5.5095E-33	-1.28052E-32	-1.458E-16	2.48467E-17	-3.19399E-17	4.76761E-31	-1.57931E-17	1.95211
taking	-9.68199E-50	-8.01319E-34	-1.10555E-33	-1.45028E-33	5.03895E-34	8.04294E-33	4.12258E-33	-3.49091E-33	4.6516E-18	2.86174E-18	-2.90215E-18	2.67951E-32	2.46458E-17	1.60178
takes	2.97904E-52	5.20382E-18	-1.20264E-17	-1.49421E-17	-3.91244E-18	4.00052E-18	-4.68527E-18	4.18528E-17	0.256642075	4.30352E-17	-6.39741E-16	-8.8017E-16	-3.32164E-17	-1.73965
take	1.11681E-34	-2.52156E-32	-2.0614E-17	-1.7544E-18	-1.06616E-17	3.18993E-17	3.54526E-18	6.54223E-17	2.00057E-18	-1.07189E-16	3.91807E-17	1.87677E-16	7.67746E-16	-2.77641
made	8.83038E-16	2.20866E-17	-8.51294E-17	1.1521E-15	-1.78567E-17	8.62092E-18	1.30594E-17	8.18574E-17	2.83806E-17	2.13868E-18	0.100402663	-4.6319E-17	-1.28731E-17	1.69672
make	-1.52869E-82	-1.14806E-33	1.95321E-32	8.27147E-34	2.01773E-33	2.48042E-32	4.48624E-33	-7.12324E-32	-2.49338E-32	-5.01989E-17	9.19086E-17	-4.2486E-32	3.85259E-17	3.18174
making	-7.88918E-82	-5.09672E-34	6.59903E-33	1.83672E-33	-6.74615E-34	1.36153E-32	1.57291E-32	3.77637E-32	2.93469E-33	3.68093E-17	-9.82423E-17	5.06895E-32	-8.93706E-19	6.12135
makes	0	0	0	0	0	0	0	0	0	0	0	0	0	
drew	7.00945E-50	7.48351E-18	-2.52993E-17	-1.3832E-16	-1.03855E-16	5.54215E-17	0.265001017	-8.8709E-17	1.07904E-16	-3.62859E-18	-9.23184E-16	-1.3099E-16	4.40568E-17	-1.06205
drawn	6.06237E-51	1.38687E-18	-4.42071E-19	-8.87154E-19	-1.34821E-17	3.85129E-18	-1.04302E-17	4.29734E-17	-3.79885E-17	4.78988E-17	-5.1411E-16	-1.843E-16	-3.93558E-18	-1.63344

Table 3.20: Cosine similarity between noun and verb

Chapter 4 : Results

4.1 Method 1: Correlation

The values of the coefficient of the correlation for both Spearman and Pearson obtained were good for the noun "blood" and "verb" run as well as noun "John" and verb "broke". This kind of relation was observed because in our dataset some pairs of verb and noun occurred together in many sentences very frequently. This led to an increase in the values of their coefficient of correlation.

Surrounding Terms	Horse(Cosine similarity)	Run (Cosine similarity)
Last	0.45	0.2468
race-course	0.358	0.3508
race	0.313	0.658
park	0.253	0.248
fast	0.1267	0.2662
very	0.08513	0.1788
Pearson Correlation=0.3325		

Surrounding Terms	Blood(Cosine Similarity)	Run(Cosine Similarity)
veins	0.6074	0.2746
body	0.2336	0.1056
heart	0.2336	0.1056
parts	0.2336	0.1056
all	0.04	0.0195
Pearson Correlation=0.9162		

Table 4.1: Pearson correlation Coefficient Values for verb "Run" with Noun "Horse" and "Blood"

Surrounding Terms	Blood(Cosine Similarity)	Run(Cosine Similarity)
veins	0.6074	0.2746
body	0.2336	0.1056
heart	0.2336	0.1056
parts	0.2336	0.1056
all	0.04	0.0195
Spearman Correlation=0.68825		

Surrounding Terms	Horse(Cosine similarity)	Run (Cosine similarity)
Last	0.45	0.2468
race-course	0.358	0.3508
race	0.313	0.658
park	0.253	0.248
fast	0.1267	0.2662
very	0.08513	0.1788
Spearman Correlation=0.31429		

Table 4.2: Spearman correlation Coefficient Values for verb "Run" with noun "Horse" and "Blood"

Surrounding Term	John	Broke
'angrily'	0.398473634	0.260714585
'batsman'	0.166030681	0.108631077
'cigarette'	0.38894454	0.254479859
'cigarettes'	0.284624024	0.186224703
'cycle'	0.199236817	0.130357292
'finally'	0.122369426	0.080064253
'glasses'	0.249046021	0.162946615
'highest'	0.166030681	0.108631077
Pearson Coefficient Value=0.9929		
Spearman Coefficient Value=0.98197		

Table 4.3: Pearson and Spearman coefficient for verb "BROKE"

4.2 Method 2: Classifier Method

We made the noun-verb matrix based on the classification from the database. We used classes depending on senses available in Word Net for a single ambiguous verb. The matrix is formed using all the nouns of dataset with senses as class of an ambiguous verb. We did the frequency calculation for the matrix.

	Verb Run																		
	Moving				Flowing				Working				Diffussion				Executing		
	runs	run	ran	running	runs	run	ran	running	runs	run	ran	running	runs	run	ran	running	runs	run	ran
Horse	2	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
fast	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
last	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
park	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
race-cours	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
race	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rabbit	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
garden	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
house	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
kangaroo	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
baby	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pouch	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
forest	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
machine	0	0	0	0	0	0	0	0	2	0	1	2	0	0	0	0	0	0	0
smoothly	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
electricity	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
factory	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
properly	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
hours	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
crude-oil	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
colors	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
dyes	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
blood	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
veins	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
heart	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
body	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4.4: Classifier matrix term frequency calculation

There were several drawbacks with the classifier method such as,

1. There were problems in the calculation of tf-idf values because total number of documents keep changing depending on the verb.
2. We are taking different forms of tenses of the verb instead of using just using the single tense. Hence there is a further need for morphological analysis in this regard.
3. The cosine similarity values obtained were totally different other than 0 to 1.The were very high since we were using document sum directly instead of vector sum.

4.3 Method 3: Document Matrix

Most of the cosine similarity values for noun-verb matrix were nearly equal to zeroes indicating that these nouns and verbs were not used together in any sentences. Maximum of the rest had values ranging between 0 and 0.3 indicating a weak relationship.

Few of them had values between 0.4 and 0.6 which shows an average relationship. Only 9 pairs of noun and verb had the value of cosine similarity greater than 0.6 indicating a strong relationship.

	horse	rabbit	kangaroo	machine	colors	dyes	blood
run	-2.8851E-49	1.75079E-18	5.10882E-16	2.60951E-17	0.942154994	0.297935569	3.60831E-16
runs	0.352998086	0.19917066	0.093659267	0.284941758	-4.99113E-17	2.09248E-16	0.191466547
ran	0.206466889	2.53763E-17	-2.81996E-18	0.278902197	-1.17579E-16	4.0885E-17	-4.78775E-17
running	0.251394319	0.320878649	0.276893496	0.30694426	-9.4394E-18	-1.74179E-16	-1.86001E-16
give	2.57267E-18	3.74422E-17	-1.5849E-17	-1.38691E-17	-2.11317E-17	2.95579E-17	-2.21845E-17
gives	-5.00576E-17	-2.6067E-17	-2.23934E-17	-6.47469E-17	3.13667E-32	-2.97498E-32	-2.45801E-19
gave	-1.82355E-17	-3.32869E-17	-2.78438E-17	-2.7269E-17	-6.62415E-18	1.04303E-17	-1.40239E-17
broke	-7.74274E-16	2.33332E-17	-4.93683E-18	-1.07174E-15	-1.00818E-17	6.52895E-18	-8.58212E-18
breaking	3.86651E-81	-3.01623E-33	1.81169E-33	4.33818E-33	6.85814E-33	8.29373E-33	5.61166E-33
called	1.05865E-35	5.48341E-18	2.10951E-17	4.08595E-17	-1.63302E-17	4.04051E-17	-1.77951E-17
call	-8.5556E-35	-1.4903E-17	1.9257E-17	-7.51971E-18	-5.74668E-18	6.17227E-17	2.29903E-17
calls	2.25922E-50	1.95942E-33	1.96169E-32	5.86981E-33	-6.49775E-33	-9.63385E-33	-1.21027E-32
know	3.18988E-35	-3.67783E-33	1.51543E-18	1.89967E-19	-1.01906E-18	2.58898E-18	-1.34928E-18
knows	1.83649E-35	1.09423E-18	2.42559E-18	-2.49734E-18	3.83692E-18	1.22623E-18	-2.79208E-18
knew	5.10982E-34	-4.49899E-33	-2.86898E-33	-5.35024E-33	-1.89841E-33	2.19777E-32	6.49508E-33
known	4.08874E-50	-1.16628E-33	7.9494E-33	3.19214E-33	-2.37957E-34	5.91346E-33	-3.16024E-33
put	8.88572E-67	4.2939E-34	4.77439E-34	1.86217E-34	3.05674E-34	1.77832E-33	4.25172E-34

Table 4.5: Cosine similarity between Noun and Verb

Chapter 5 : Conclusion

We are trying to find out the relation between noun and verb. The meaning of the verb changes depending on the noun and its surrounding term. If the value of cosine similarity is greater than 0.5 then we can say that there is closeness between the ambiguous verb and the noun. So the sense of the verb can be predicted as the sense is given in the WordNet and also those terms which are present with the particular verb and its corresponding noun can be considered as surrounding terms.

If the dataset is increased by addition of more sentences using the ambiguous verb then we can expect a further increase in the accuracy of the result. It's a fact that if the dataset is increased we get higher mapping between the noun and verbs. Whenever a new test sentence will be added we can check it in the existing system and if it is not available then we can add into the sentences. If the dataset is containing different forms of verb then we can remove ambiguity in a better way to get better results. So we strongly recommend to use this methodology to a larger corpus of database.

Bibliography

[1] Dataset- https://wordnet.princeton.edu/

[2] https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/

[3] Kintsch, Walker. (2001), ”Predication”, Cognitive Science, 25, 173-202.

[4] B Kievit-Kylar, GKachergis, MNJones, ”Naturalistic Word-Concept Pair Learning with Semantic Spaces”, Cognitive Science,2013.

[5] George A. Miller. Word net: A Lexical Database for English Communication, ACM, 38(11):39-41, 1995.

[6]Roberto Navigli, ”Word sense disambiguation: A survey”. ACM Computing Surveys. 41(2), 2009.

[7]Pal, A. R.,Saha, D. “Word sense disambiguation: A survey”. International Journal of Control Theory and Computer Modeling (IJCTCM) Vol.5, No.3, July 2015.

[8]Jorge-Botana, Guillermo, et al. "Visualizing polysemy using LSA and the predication algorithm." Journal of the American Society for Information Science and Technology 61.8(2010): 1706-1724.

[9] George A. Miller. Word net: A Lexical Database for English Communication, ACM,38(11):39-41, 1995.

[10]Leong, C. W., and R. Mihalcea., “Measuring the semantic relatedness between words andimages”, Proceedings of IWCS-9, Oxford, UK: Special Interest Group on Computational Semantics of the Association for Computational Linguistics, 2011.

[11] Kintsch, W. (2001),” Predication”,Cognitive Science, 25, 173-202.

[12]Kievit-Kylar, GKachergis, MNJones,”Naturalistic Word-Concept Pair Learining with Semantic Spaces”,Cognitive Science,2013.

[13]IDE, N. AND V´ERONIS, J., “Word sense disambiguation: The state of the art”,

Computational Linguistics, 24, 1, 1–40, 1998.

[14]Roberto Navigli,” Word sense disambiguation: A survey”. ACM Computing Surveys.41(2), 2009.