Developing a Word2Vec Model

Fahmina Kabir*
fakabiradj@gmail.com
Israt Tasnim*
tasnim.lamisha@gmail.com

1. Introduction

We are aiming to develop a Word2Vec model from scratch using free books. Then we will try to analyze the model by observing similarity and distance among words from the obtained model. Word embedding is one of the most popular representations of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. Word2Vec is one of the most popular techniques to learn word embedding using a shallow neural network. Word2Vec is essential to solving Natural Language Processing problems, such as, sentiment analysis, survey responses, recommendation engines, and more.

One of the reasons that Natural Language Processing is a difficult problem to solve is the fact that, unlike human beings, computers can only understand numbers. We have to represent words in a numeric format that is understandable by the computers. Word embedding refers to the numeric representations of words. Word2Vec is a popular word embedding technique to represent words in forms of vectors which solves this problem.

This task can be challenging, as we are aiming to develop our own corpus from books. So, we have to find books which are free of copyright issues, which can be difficult. It might also be difficult to find books with a wide range of words from different domains.

*Author order has been decided alphabetically.

2. Related Works

Recently word embedding techniques are being widely used in NLP. Some of the works related to our project is described in the following section:

Considering [1], we can see that, this paper proves that smaller explanatory test datasets like, English Wikipedia and Simple English Wikipedia can provide enough semantic information to get equal or better accuracy than using large current event datasets. We know, vectorial representations obtained from large datasets such as Google News perform well on word similarity tasks. But, this research shows that small datasets can perform well too. They have used FW-CBOW, SW-CBOW etc. models and calculated cumulative scores of accuracy rate at different recall rate points. According to their research, FW-CBOW outperformed other models.

Considering [2], Word2vec has been populated with features for text classification tasks such as sentiment analysis. Recently, there are many available word2vec models such as GoogleNews-vectors-negative300 and word2vec-twitter-model that help researchers doing sentiment analysis easier. In this paper, authors proposed a new method to solve text classification tasks. They have extracted features of tweets (verb or adjective), then run them with available word2vec models before classifying the tweets with machine-learning technologies. This aimed to provide an effective method for new researchers about sentiment analysis.

Studying [3], in this paper, they have performed an empirical analysis of Word2Vec by comparing its output to WordNet, a well-known, human-curated lexical database. They have shown in this paper that, Word2Vec captures far more of certain relations than others: favoring synonyms, hyponyms and hypernyms ahead of holonyms and meronyms across the Reuters corpus. They have used the Word2Vec model that Google trained on the 100 billion word tokens in its Google news corpus. They have used NLTK, Gensim for preprocessing their dataset. They have shown that for some word w, they have found the probability that a neighbor at a given cosine distance is a synonym, a hypernym, a holonym or a meronym.

Most of these works mainly focused on their final results, so they mostly used tokenization of the documents as the preprocessing method. We are aiming to apply a few more processing techniques.

3. Project Objective(s)

As mentioned before, we are developing a Word2Vec model using books which are free of copyright issues. Then we are going to analyze this model in terms of cosine similarity and cosine distance. We are following a structured approach for developing this model. The flow chart of the entire process is given in the figure below:

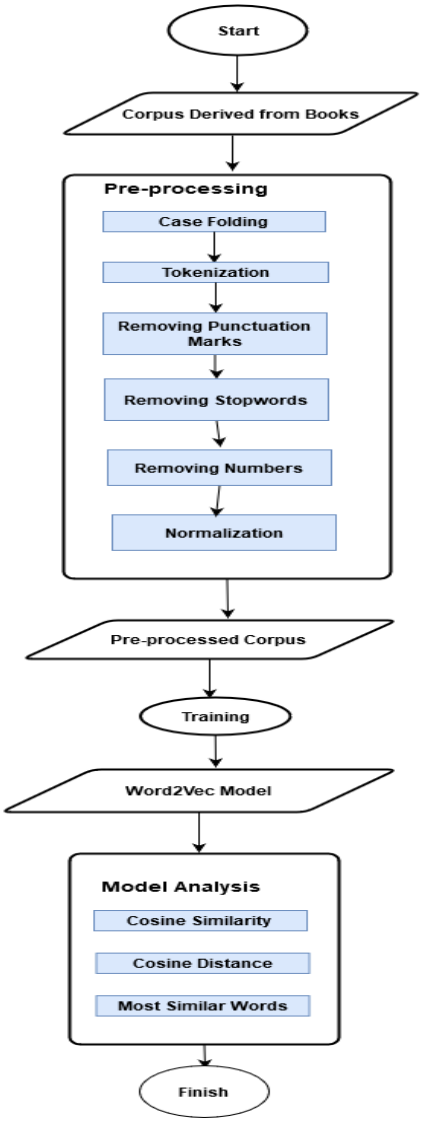

Figure:1 Flow Chart of Proposed Model

These steps are further explained in the following sections.

4. Methodologies / Model

Word2Vec is an unsupervised system for determining the semantic distance between words. It does not label particular semantic relationships between words, Instead, it assigns a number between 0 and 1, indicating the semantic distance between two words. It can generate ranked lists showing which words are closer and which words are further away in a semantic model. We have considered similarity and distance to analyze this model in terms of semantic resemblance. To visualize the model, we have tried to display words similar to our chosen words. We have followed the steps given below:

4.1 Corpus Derived from Books

We have a function which reads all the text files(books) in specified folder and saves the file's content into an item of a list. Then returns the list which contains the contents of the books. Then we have another function which takes the derived list as an input and returns a list of list of words or a list of sentences(corpus). This list and information about this list is stored in a JSON file.

4.2 Preprocessing

Preprocessing is the process of converting unstructured data into structured data as needed, for further text mining processes. Our preprocessing phase contains 6 steps. Which are: i) Case Folding ii) Tokenization iii) Removing Punctuation Marks iv) Removing Stopwords v) Removing Numbers vi) Normalization.

Case folding is the process of matching a case in a document. It is needed to convert all text in a document into a standard form (usually lowercase). We have converted all words of our corpus into lowercase. Tokenization is the process of dividing text in the form of sentences, words, phrases, symbols or other meaningful elements called tokens. Punctuation marks, stopwords and numbers in the corpus are not important to our analysis, that is why we have removed them from our corpus. We have used lemmatization for our dataset normalization. The goal of lemmatization is to normalize words to their common base form.

4.3 Training Model

The vector space representation of the words provides a projection where words with similar meanings are locally clustered within the space. We have used the word embedding algorithm Word2Vec to train our model. We have set min_count to 1. A value of 1 for min_count specifies to include those words in the Word2Vec model that appear at least

once in the corpus. We have used CBOW word embedding technique. The vocabulary count was 91050. Then we analyzed the model.

Some important imported libraries are described below:

OS

OS exports functions from posix or nt, example: unlink, stat etc. POSIX means "Portable Operating System Interface for uni-X". So programs that import os will have better portability among various platforms. We have used the listdir() method which is described as listdir(path=None). It returns the list of names of files in the directory specified in path.

JSON

The full form of JSON is "JavaScript Object Notation". It is a subset of JavaScript syntax used as a lightweight data interchange format. A JSON object in python is nothing but a dictionary. We have used load() which simply translates JSON object into python object.

gensim.models

gensim library includes functionalities to compute similarities of a document with a corpus of documents. A part of this library is models, which basically provides us with some models to retrieve semantically similar documents, strings. We have used Word2Vec. It takes a text as input and outputs a list of vectors, which contains feature vectors for the words in the given text.

NLTK

NLTK’s full form is "Natural Language ToolKit". It helps to work with human language data. NLTK is a huge library. We had to import stopwords from NLTK. It basically gives us a set of English stopwords. Words similar to a, about, and, might, mostly, to, those, we are considered stopwords. These are considered useless words in computer programs. We have used sent_tokenize(), RegexpTokenizer(), WordNetLemmatizer().

Word2Vec

We have used Word2Vec models to process our dataset. Word2Vec is a neural network that has 2 layers. It vectorizes texts, which means it processes given text into a set of vectors, typically known as features vectors. Feature vectors represent words in that given text. It takes a text corpus as input and outputs features vectors. It turns texts into numerical forms.

Word2vec is used to group the vectors of similar words together in vector space. It detects similarities between words mathematically. Word2vec creates vectors that are distributed numerical representations of word features, features such as the context of individual words.

The output of the Word2vec is a vocabulary in which each item has a vector attached to it.

5. Experiments

5.1 Dataset

We have used books which are free of copyright issues. Firstly, we have downloaded 51 books in pdf format. We have downloaded the books from openlibra.com which is a free online bookstore. We have tried to download books from different categories such as: Graphic Designing, Agile Software Development, Art and Design, Business, Chess, Cinema, Comics, Computer Programming, Database System, Education, Geography, History, Language, Music, Philosophy, Robotics, Web Development etc. so that we get a wide range of words. Then we have converted the pdf files into text format. We have used books which have 2 types of licenses. These are:

i) **Open Access**: Open access (OA) refers to free, unrestricted online access to research outputs such as journal articles and books. OA content is open to all, with no access fees.

ii) **CC-BY:** A CC (Creative Commons) license is used when an author wants to give other people the right to share, use, and build upon the work that they (the author) have created. Others can copy, distribute, display and edit the author's work if they credit the author properly.

5.2 Model Analysis

Text Visualization is an important part of text analysis and text mining. We have tried to visualize the model using a limited number of words. We didn’t use all the words to visualize because using all the words makes the figure clumsy.

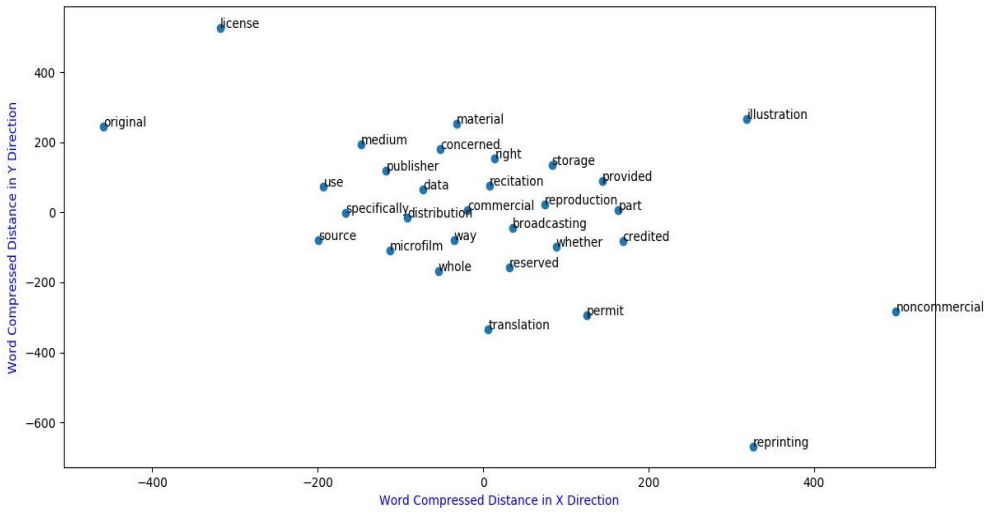

Figure:2 Visualization of 30 words from the model

Cosine Similarity is the calculation of the similarity between two n-dimensional vectors by looking for a cosine value from the angle between the two and is often used to compare documents in text mining. The cosine distance is then defined as,

Cosine Distance = 1 − Cosine Similarity

Euclidean distance depends on a vector's magnitude whereas cosine similarity depends on the angle between the vectors. The angle measure is more resilient to variations of occurrence counts between terms that are semantically similar, whereas the magnitude of vectors is influenced by occurrence counts and heterogeneity of word neighborhood. That is why we have used cosine distance instead of Euclidean distance

We have selected three pairs of words to analyze our model. We have calculated cosine similarity and cosine distance for each pair of words. They are given in the following table.

Pair of Words	computer-virtual	window-frame	month-week
Cosine Similarity	0.6872372	0.8051437	0.9313405
Cosine Distance	0.3127627	0.1948562	0.0686594

Semantic similarity is a concept that can measure the similarity of meaning in the context of short texts. Text that is compared can be in the form of words, short sentences, and a document. In each of the selected pairs, the words are semantically similar, so the cosine similarity of the words of each pair should be high and the cosine distance of the words of each pair should be low. Here, we can see from the above table 1 that, the cosine similarity of the words of each pair are pretty high and the cosine distance of the words of each pair are low.

The visualization of most similar words of computer, window and month is shown below:

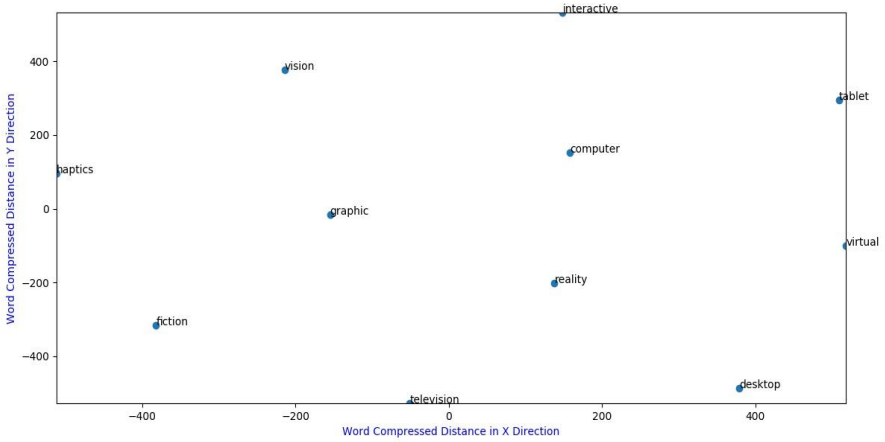

Figure:3 Visualization of most similar words of computer

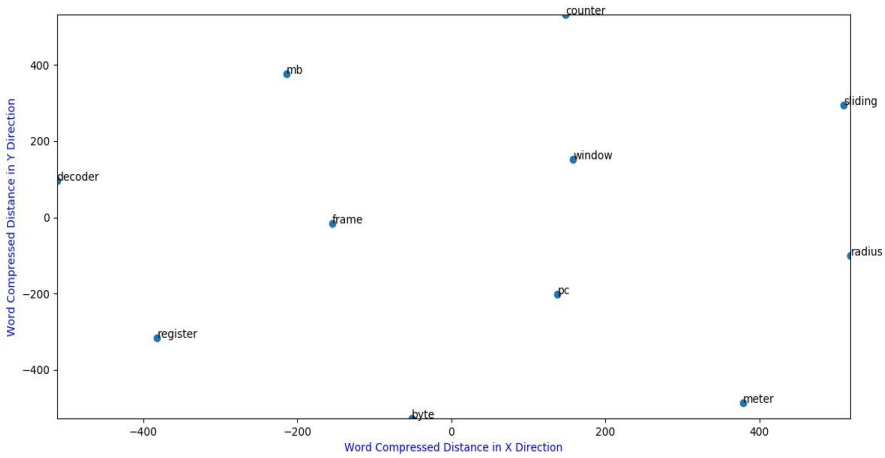

Figure:4 Visualization of most similar words of window

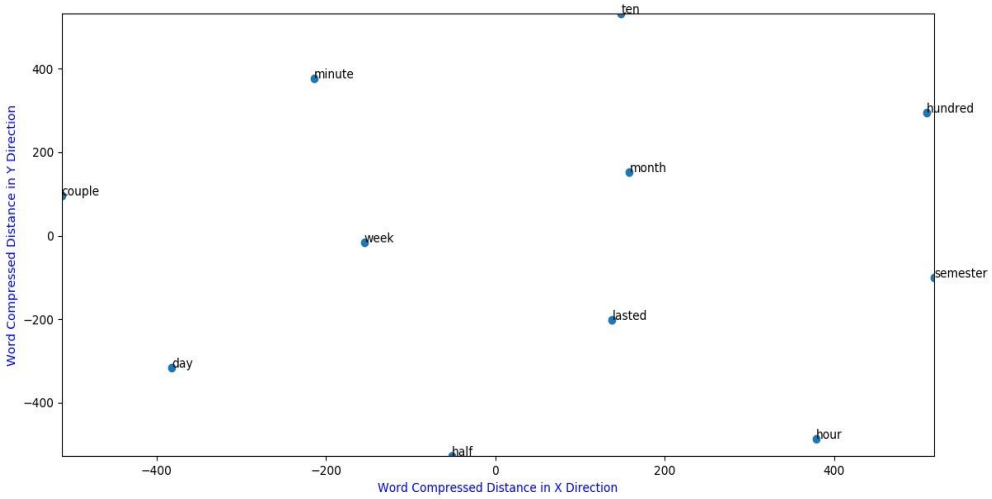

Figure:5 Visualization of most similar words of month

From the above figures, we can see that the words from selected pairs are close to each other.

6. Conclusion

We have implemented a Word2Vec word embedding model with Python's Gensim Library. We have done this by using open source books and built our Word2Vec model using the books as a corpus. We have analyzed our model by calculating cosine similarity and cosine distance and tried to visualize the model by displaying random words, most similar words to our selected words. In the future, we might try to increase the vocabulary count. But we must keep in mind that increasing vocabulary with uninformative words may cause time and space complexity.

7. References

[1] Jin, Lifeng, and William Schuler. "A comparison of word similarity performance using explanatory and non-explanatory texts." In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 990-994. 2015.

[2] NguyễnThịThúyHoài, LêThiệnNhậtQuang. “Twitter sentiment analysis with word2vec." In Proceedings of the 2019 International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 8, Issue 1, January 2019, ISSN: 2278 – 1323.

[3] Handler, Abram. "An empirical study of semantic similarity in WordNet and Word2Vec." (2014).