

# **EFFICIENCY DEMONSTRATION OF EMBEDDING MODELS AND LIBRARIES FOR BENGALI WORD VECTOR REPRESENTATION**

**BY**

**Aminul Islam Bulbul  
ID: 183-15-12040**

**Saurav Das  
ID: 183-15-11851**

**AND**

**Tamanna Tasnim  
ID: 183-15-11858**

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science and Engineering.

Supervised By

**Dr. Sheak Rashed Haider Noori**  
Professor  
Department of CSE  
Daffodil International University

Co-Supervised By

**Dr. Md. Ismail Jabiullah**  
Professor  
Department of CSE  
Daffodil International University



**DAFFODIL INTERNATIONAL UNIVERSITY**

**DHAKA, BANGLADESH**

**DECEMBER 2022**

## APPROVAL

This Research Project titled “EFFICIENCY DEMONSTRATION OF EMBEDDING MODELS AND LIBRARIES FOR BENGALI WORD VECTOR REPRESENTATION” submitted by **Aminul Islam Bulbul(183-15-12040)**, **Saurav Das(183-15-11851)**, and **Tamanna Tasnim(183-15-11851)** to the Department of Computer Science and Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 29/01/2023.

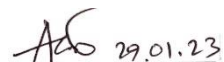
### BOARD OF EXAMINERS



**Dr. Touhid Bhuiyan**  
**Professor and Head**

Department of Computer Science and Engineering  
Faculty of Science & Information Technology  
Daffodil International University

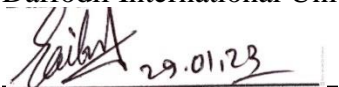
**Chairman**



**Arif Mahmud**  
**Assistant Professor**

Department of Computer Science and Engineering  
Faculty of Science & Information Technology  
Daffodil International University

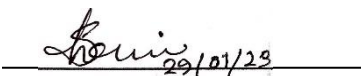
**Internal Examiner**



**Saiful Islam**  
**Assistant Professor**

Department of Computer Science and Engineering  
Faculty of Science & Information Technology  
Daffodil International University

**Internal Examiner**



**Dr. Shamim H Ripon**  
**Professor**

Department of Computer Science and Engineering  
Jahangirnagar University

**External Examiner**

## DECLARATION

We hereby declare that this research project has been done by us under the supervision of **Dr. Sheak Rashed Haider Noori, Professor, Department of CSE** Daffodil International University. We also declare that neither this research project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by: 

---

**Dr. Sheak Rashed Haider Noori**  
Professor  
Department of CSE  
Daffodil International University

Co-Supervised by:



---

**Dr. Md. Ismail Jabiullah**  
Professor  
Department of CSE  
Daffodil International University

Submitted by:



---

**Aminul Islam Bulbul**  
ID: 183-15-12040  
Department of CSE  
Daffodil International University



---

**Saurav Das**  
ID: 183-15-11851  
Department of CSE  
Daffodil International University



---

**Tamanna Tasnim**  
ID: 183-15-11858  
Department of CSE  
Daffodil International University

## ACKNOWLEDGEMENT

First, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the final year research project successfully.

We are grateful and wish our profound our indebtedness to **Dr. Sheak Rashed Haider Noori, Professor**, Department of CSE Daffodil International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of “*Deep Natural Language Processing*” to carry out this research project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

We would like to express our heartiest gratitude to **Dr. Touhid Bhuiyan, Professor, and Head**, Department of CSE, for his kind help to finish our project and to other faculty member and the staff of CSE department of Daffodil International University.

We would like to thank our entire course mate in Daffodil International University, who took part in this discuss while completing the course work.

Finally, we must acknowledge with due respect the constant support and patients of our parents.

## ABSTRACT

Word embedding demonstrates the magic in the field of NLP. Our goal is to find out the best embedding model. Finding out the best embedding model for specific tasks is difficult. Embedding demonstrates different results according to size and source of data set in various embedding tasks. The purpose of the study is to find out the performance it shows for different types of embedding tasks. Researchers have invented several embedding models after finding the magical performance of word-embedding in the field of NLP. In our paper we discussed CBOW, skip-gram and Glove models performance. The models do embedding by representing the word into vector forms. We collect 2.5 lakh Bengali newspapers articles from a renowned newspaper of BD. We trained the architectures CBOW and skip-gram, which is for word2vec and FastText models, dataset containing 20 million Bengali words. We use the same data set for training the Glove model. For collecting such a large amount of data, we build a web scraper by using Scrapy. Gensim, FastText and the python library has been used for training these three models consequently. For evaluating the models, we perform various word embedding tasks namely word analogy, semantic and syntactic prediction of words. Surprisingly they FastText perform in a better way for semantic and syntactic tasks than others. On the other hand, for analogy task, the performance was almost same for all the models except Glove.

## TABLE OF CONTENTS

<b>CONTENTS</b>	<b>PAGE</b>
Board of examiners	ii
Declaration	iii
Acknowledgements	iv
Abstract	v
Table of Contents	vi-xi
<b>CHAPTER 1: INTRODUCTION</b>	<b>1-7</b>
1.1 Introduction	1
1.2 Motivation	4
1.3 Rational study	4
1.4 Research Questions	6
1.5 Expected Outcome	6
1.6 Report Layout	7
<b>CHAPTER 2: LITERATURE REVIEW</b>	<b>8-13</b>
2.1 Literature Review	8
2.2 Challenges	13

<b>CHAPTER 3: RESEARCH METHODOLOGY</b>	<b>14-36</b>
3.1.1 Model’s Theoretical and Mathematical Base	14
3.1.2 Data Collection and Pre-Processing	23
3.2 Collecting Data	24
3.3.1 Data Preprocessing	24
3.3.2 Computer and Human Languages	25
3.3.3 One-hot Encoding	27
3.4.1 Word-embedding	28
3.4.2 CBOW	29
3.4.3 SkipGram	30
3.4.4 FastText	31
3.4.5 Glove	33
<b>CHAPTER 4: RESULTS AND EVALUATION</b>	<b>37-49</b>
4.1 Result of Different Models	37
<b>CHAPTER 5: IMPACT ON SOCIETY, ENVIRONMENT AND SUSTAINABILITY</b>	<b>50-53</b>
5.1 Impact on Society	50
5.2 Impact on Environment	51
5.3 Ethical Aspects	51
5.4 Sustainability Plan	

<b>CHAPTER 6: CONCLUSION AND FUTURE WORK</b>	<b>54-55</b>
6.1 Summary and Conclusion	54
6.2 Future Work	54
<b>REFERENCES</b>	<b>56-57</b>



## LIST OF FIGURES

FIGURES	PAGE NO
Figure 3.1.1.1: Euclidean Distance	15
Figure 3.1.2.2: Angle of The Reference Direction.	16
Figure 3.1.3.3: vector representation of our word	17
Figure 3.1.4.4: Multidimensional Binary Tree	21
Figure 3.1.5.5: Neural Network	22
Figure 3.1.6.6: Hidden layer	22
Figure 3.4.2.7: Architecture of CBOW model	29
Figure 3.4.3.8: Architecture of SkipGram model	31
Figure 3.4.5.1: Glove Model	35
Figure 3.4.5.2: Glove Model	35
Figure 4.1.1: Gensim Word2vec Cbow 100dim model	38
Figure 4.1.2: Gensim Word2vec Cbow 100dim model	38
Figure 4.1.3: Gensim Word2vec Cbow 300dim model	39
Figure 4.1.4: Gensim Word2vec Cbow 300dim model	39
Figure 4.1.5: Gensim Word2vec SkipGram 100dim model	40
Figure 4.1.6: Gensim Word2vec SkipGram 100dim model	40
Figure 4.1.7: Gensim Word2vec SkipGram 300dim model	41
Figure 4.1.8: Gensim Word2vec SkipGram 300dim model	41
Figure 4.1.9: Gensim Fasttext SkipGram 100dim model	42
Figure 4.1.10: Gensim Fasttext SkipGram 100dim model	42
Figure 4.1.11: Gensim Fasttext SkipGram 300dim model	43

Figure 4.1.12: Gensim Fasttext SkipGram 300dim model	43
Figure 4.1.13: Gensim Fasttext Cbow 100dim model	44
Figure 4.1.14: Gensim Fasttext Cbow 100dim model	44
Figure 4.1.15: Gensim Fasttext Cbow 300dim model	45
Figure 4.1.16: Gensim Fasttext Cbow 300dim model	45
Figure 4.1.17: Fasttext SkipGram 300dim model	46
Figure 4.1.18: Fasttext SkipGram 300dim model	46
Figure 4.1.19: Fasttext Cbow 100dim model	47
Figure 4.1.20: Fasttext Cbow 100dim model	47
Figure 4.1.21: Glove 300dim model	48
Figure 4.1.22: Glove 300dim model	48
Figure 4.1.23: Word in Embedding Space	49
Figure 5.4.1: Dataflow of The Scrapy of Spider	53

**LIST OF TABLES**

<b>TABLES</b>	<b>PAGE NO</b>
Table 3.4.4.1: FastText working methodology representing	32

# CHAPTER 1

## INTRODUCTION

### 1.1 Introduction

Every day we communicate with each other. There are a couple of ways to communicate. Language is one of them. We generate sentences to express ourselves. To do that we use languages. The formation of languages varies from country to country. The same message but with a different symbol. Symbol means letter. Letter is the fundamental part of languages. More than one letter makes a word, and more than one word makes a sentence. Humans can communicate with each other with these languages according to their knowledge. Bengali is one of the most spoken languages in all over the world. It is the fifth and seventh most pronounced language according to native and general speakers respectively [14]. Because of the complex architecture of the Bengali language, it is more difficult to work with than English. Since the language is morphologically rich and the composition is hard, a tiny change or modification can convert the entire meaning into a contrary meaning. [15]. Computer is a machine for solving several critical tasks. In the present time we are using computers for maximum of our work. We use voice commands through the computer or mobile. We type through the computer for messaging or any other writing purposes, namely email, thesis, newspaperese. On the internet a large portion of information is stored as text format. Every day we are using search engines and find our desired things from the internet through voice command or text, sometimes with pictures. The search engine then searches the keyword inside the internet and finds it. Since the computer doesn't understand the raw text that is why it is important to convert the text into a numeric format. Embedding is one of them but not the only way. There are bunch of other way to make it namely, one-hot encoding, count-based representation and embedding. Many research prove that word-embedding or the vector

representation demonstrates a greater result in the field of NLP tasks than the other methods [1].

Word embedding is the way of representation of the words. It is declared that semantic words should stay together in the vector space. The idea solves many unsolved problems in the field of NLP. The reason why word-embedding gets its popularity, and it is increasing every day. In English language many works already happen in embedding. Though Bangla language is the most speak language in all over the world but there is not such notable works for Bengali language in NLP. The users of the internet for Bengali people are increasing day by day. A large portion of internet users pronounced medium is Bengali. Hence it becomes a need to beautify the language and much work should happen in this sector. For getting a tremendous result in any kinds of nlp tasks for Bangla language it is important to increase the work in word embedding. Because the nlp task depends on word-embedding a lot. There are multiple word-embedding models that have been invented till now. Namely, Doc2vec, word2vec, Glove,[4] FastText and [5] Bert are the most common embedding models. [1][2][3][4][5]

In this study we tend to find out the best word embedding for Bengali language with our own dataset. Since Bengali language complexity makes the path more complicated than other languages hence very few attempts have been taken by some author in the era of embedding. [6] A neural lemmatizer for Bengali word embedding has been proposed by Abhushek et alin 2016 where they have used the wor2vec model.[7] Adnan Ahmed and Ruhul Amin have collected a large amount of Bengali data for document classification problems from 13 different Bengali newspapers in 2016. Where the vector representation of Bengali words is done by using word2vec model.[8] Ritu et al try to analyze the performance of various embedding models in 2018 for the Bengali language. They perform word2vec in two different platforms, gensim and TensorFlow for word clustering. FastText is another model that they have used. They trained their models with not only their own corpus, which they have collected from newspapers but also, they have added SUMonno [11] with it.[9] Nafiz et al finds out the performance by evaluating various embedding tasks. They perform Semantic Relatedness, Synonym Detection,

Antonym Detection, and Concept Categorization. Among them, In Concept Categorization they got the best result. They have used two fundamental architectures of word2vec model, SkipGram and CBOW.[10] They collected a huge amount of Bengali data from several renowned Bengali newspapers. They have collected 500000 articles and trained their models with 32 million of words. They choose almost every word-embedding model for training. They have use SkipGram, CBOW for both word2vec and FastText and python library for GLOVE model. They evaluate their models with two main embedding tasks, semantic analysis and word-analogy. Among the five models they have found FastText produced the best result for almost every task. We evaluate our models by finding the semantic, syntactic words and word analogy task. We perform two different architectures CBOW and SkipGram for these tasks for FastText, Word2vec and also for GLOVE. In word2vec CBOW predicts the targeted word after providing the context words on the other hand SkipGram produces the whole context after offering the middle word. FastText works in similar way but instead of considering the words as input it takes the character or n-gram of character as input. In the training time both models only consider the context words co-occurrence probability [3][4], do not consider both local and the global co-occurrence probability unlike Glove [3]. Finding out the best embedding models among them will play a significant role to increase the work in Bangla Natural Language Processing tasks. To do this we trained our model with a large corpus bearing 20 million of unique words. Since we want find-out the best dynamic word clustering model from our experimental models that is why we trained the model in different platforms. In our study, for clustering the word we trained word2vec in Gensim library, FastText in Gensim library, FastText in Facebook FastText Library, Glove with python library. There is no study remain in this field with such a verity. Only we are considering all most every library or platforms for each embedding models by providing different dimensions in the training period. What nobody did before.

## **1.2 Motivation**

Bengali is one of the most spoken languages in all over the world. Textual data of Bengali is increasing on the internet day by day. But unfortunately, there is not a that much notable works happen in nlp for Bengali language. So, researchers think that more work should happen on Bengali language in the field of nlp. You will find a very few attempts have been made for Bangla language on the internet before. Sentiment analysis, name entity recognition and text summarization are some common tasks in NLP. For working with any of this stuff, the term word-embedding come in place. The more appropriate result embedding will bring, the nlp task will be far better. Since there are numbers of embedding models remain, finding out the best one for a specific nlp task will be time consuming and reason of a perfect nlp task. But the problem is there are a few attempts that happen in this field which we will explain later in the literature review section. Word embedding means vector representation of the words. It tells that the semantic words should stay together in the multi-dimensional vector space. So, in this study we plan to find out the robust embedding model by training it into our own data set. Our aim is to not only find out the best word embedding model but also enrich the work in this sector as descendant's researcher can find it helpful. We hope that many more studies will happen in the field of nlp if we can ensure the availability of data and work resources.

## **1.3 Rational of the study**

Bengali is one of the most spoken languages in all over the world. Textual data of Bengali is increasing on the internet day by day. But unfortunately, there is not a that much notable works happen in nlp for Bengali language. So, researchers think that more work should happen on Bengali language in the field of nlp. You will find a very few attempts have been made for Bangla language on the internet before.

Sentiment analysis, name entity recognition and text summarization are some common tasks of NLP. For working with any of this stuff, the term word-embedding comes in place. Word embedding means vector representation of the words. It tells that the semantic words should stay together in the multi-dimensional vector space. So, in this study we plan to find out the robust embedding model by training it into our own data set. The more appropriate result embedding will bring, the nlp task will be far better. Since there are numbers of embedding models remain, finding out the best one for a specific nlp task will be time consuming and reason of a perfect nlp task. But the problem is there are a few attempts that happen in this field which we will explain later in the literature review section. Though a tiny number of works happen in this field but maximum of them by pre-trained data set. Which does not bring any significant change on nlp tasks. Some researchers did their study for analyzing embedding models, but which have some lacking.

We find out the reason why there is a small amount of study happens in NLP. Because of the resources of Bengali text data on the internet. It is very hard to get the textual data for your study.

In our study we tried to recover those problems. We have decided to make our vast amount of data, which is containing 300000 articles, public. As more study will be on this sector in Bengali language. There are multiple platforms remaining for implementing the models. We can train word2vec model in gensim package or in TensorFlow. FastText can be train in FastText official library, gensim also in TensorFlow. For Glove we can use Python library for glove or in gensim or directly by glove GitHub resource. In the past nobody did train their model for all platforms. Maximum platform used by M. Salehin et al [12].

So, we plan to find out the best embedding model and the best platform for training the model in various ways.

We consider the evaluation result of embedding models and their time complexity for different packages.



## 1.4 Research Questions

1. What is the procedure of collecting Bengali textual data?
2. How hard is it to collect the data?
3. How do we pre-process the data?
4. Which tool do we use to collect the data?
5. Why did we choose this specific study?
6. What are the methodologies we use?
7. What were the challenges?
8. How perfectly do the models work?
9. Did we get the desired output?
10. Why do we choose word2vec, FastText (SkipGram, Cbow) and glove?
11. What is the reason for choosing different platforms or libraries for training these models?
12. Does it bring any significant change in this field?

## 1.5 Expected Outcome

We perform different embedding models with our own Bengali corpus on different platforms. Our training embedding models are wor2vec in cbow and skipgram, for FastText we did the same and for glove model we use python library.

We trained the wor2vec model and FastText in the gensim library. for FastText we also trained it in FastText official library.

We tried to find out the best model by performing various evaluating methods.

## **1.6 Report Layout**

Here the first part of this paper contains the introduction, Motivation, Rationale of the study, questions about research, Expected Outcome and Report Layout consequently. The second part of this paper is containing the Terminology, literature review and the challenges. In the part, we discussed about the methodology where we talked about Data collection and preparation, Data pre-processing, Output generation, Results and evaluation. In fourth part we discussed about the result. We conclude the study by providing conclusion and the future working plan we are thinking.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Literature Review

Word embedding is the way of representation of the words. It is declared that semantic words should stay together in the vector space. The idea solves many unsolved problems in the field of NLP. The reason why word-embedding gets its popularity, and it is increasing every day.

In English language many works already happen in embedding. Though Bangla language is the most speak language in all over the world but there is not such notable works for Bengali language in NLP. The users of the internet for Bengali people are increasing day by day. A large portion of internet users pronounced medium is Bengali. Hence it becomes a need to beautify the language and much work should happen in this sector. For getting a tremendous result in any kinds of nlp tasks for Bangla language it is important to increase the work in word embedding. Because the nlp task depends on word-embedding a lot. There are multiple word-embedding models that have been invented till now. Doc2vec, word2vec, Glove, FastText and Bert are the most common embedding models. [1][2][3][4][5]

[1] Quoc V. Le, Tomas Mikolov. "Distributed Representations of Sentences and Documents".arXiv:1802.0683

[2] Mikolov, T., and J. Dean. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems (2013). arXiv:1310.4546v1.

Efficient Estimation of Word Representations in Vector Space. [2] Mikolov et al. proposed two architectures of wor2vec model for embedding, Continuous bag of words

and Continuous skip-gram model in shorten cbow and SkipGram. The training procedure is vice-versa for both models. Cbow predicts the targeted word after providing the context word skipgram does the opposite thing.

The architecture “cbow” follows the feedforward NNLM [paper] model. In the training time it does train itself continuously from one word to another word according to the window size given. It will predict the middle or target word if the context words are given. After predicting the word, it will go one word forward and do the same for finding the word co-occurrences probability based on given context. Every time the forward propagation happens, by inputting the context words one-hot encoded representation in a shallow neural network. The complexity of the training for this model is  $Q = N \times D + D \times \log_2(V)$ . In this paper they have mentioned that the weight matrix of input and the projection layer is shared for all words as like NNLM.

In skipgram same thing happens in a contrary way. Here skipgram predicts the context word instead of predicting the middle or current word. They have found it works better in some way than cbow but also it increases the time complexity. The training complexity then becomes  $Q = C \times (D + D \times \log_2(V))$ .

[3] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation." EMNLP. Vol. 14. 2014.

They proposed global vector or glove model. It is another process of word embedding. Glove basically introduced by Stamford university. Glove means global vector. Where word2vec model only looks at the local value of the words. One the other hand glove considers both the local and global representation of the words. The reason why the model is also known as the extended version of wor2vec. The whole process happens by generating a word-word co-occurrence matrix.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov. Enriching Word Vectors with Sub word Information. Transactions of the Association for Computational.

Here they found the model FastText. The model converts the words into a vector by using n-gram technique. FastText represents a single word as an n-gram. It is an advanced additional version of word2vec. The core difference between FastText and word2vec is, instead of working with words it works with n-gram of letters. When FastText gets a word as same size as  $n$  or  $>n$  it will consider it as a complete word and save it into the vocabulary such as দেশ, বাংলা. suppose we are splitting the word “বাংলাদেশ”, we get “দেশ”, and “বাংলা” which is a meaningful word in Bangla Language. This is one of the great successes of FastText’s in the land of word embedding. Let me clear it, suppose we have a corpus where the words “দেশ”, and “বাংলা” don’t exist but the word “বাংলাদেশ” is.

So, we see that FastText can make words which don't even exist in the corpus.

In this study we tend to find out the best word embedding for Bengali language with our own dataset.

Since Bengali language complexity makes the path more complicated than other languages hence very few attempts have been taken by some author in the era of embedding.

[6] A neural lemmatizer for Bengali word embedding has been proposed by Abhushek et al in 2016 where they have used word2vec model.

[7] Adnan Ahmed and Ruhul Amin have collected a large amount of Bengali data for document classification problems from 13 different Bengali newspapers in 2016. Where the vector representation of Bengali word done by using word2vec model.

[8] Ritu et al try to analyze the performance of various embedding models in 2018 for Bengali language. They perform word2vec in two different platforms, gensim and TensorFlow for word clustering. FastText is another model that they have used. They trained their models with not only their own corpus, which they have collected from newspapers but also they have added SUMonno [11] with it. [11] M. A. Al Mumin, A. A.

M. Shoeb, M. R. Selim, and M. Z. Iqbal, "Sumono: A representative modern Bengali corpus," *SUST Journal of Science and Technology*, vol. 21, pp. 78–86, 2014

[9] Nafiz et al finds out the performance by evaluating various embedding tasks. They perform Semantic Relatedness, Synonym Detection, Antonym Detection, and Concept Categorization. Among them, In Concept Categorization they got the best result. They have used two fundamental architectures of word2vec model, SkipGram and CBOW. They trained their models with a large dataset of 1.3 million of unique words. They have collected their data from various source of Bengali textual data, though the maximum of the data came from a single blog post. They examine their models with different sizes of dimension 300,100,64 and 32. SG provides the best performance in concept categorization with accuracy 91.02% and worse performance for antonym detection .[9]N. Sadman, A. Sadmanee, M. I. Tanveer, M. A. Amin and A. A. Ali, "Intrinsic Evaluation of Bangla Word Embeddings," *2019 International Conference on Bangla Speech and Language Processing (ICBSLP)*, 2019, pp. 1-5, doi: 10.1109/ICBSLP47725.2019.201506.

[10] Rifat Rahman has tried to find out the better embedding model but only by comparing between two architectures of word2vec model, SkipGram and CBOW. But conclude by mentioning that, he will do the same for the rest of the embedding models namely glove, FastText model in near future.

R. Rahman evaluates the models both extrinsic and intrinsic ways by fine tuning the word2vec models. For doing this he has collected the Bengali newspapers data from several newspapers by making a web crawler with beautiful soup. He has collected 243741 articles consisting of 4776708 sentences.

For intrinsic evaluation he calculates the purity and Normalized Mutual Information of k-mean clustering. On the other hand, for extrinsic evaluation, he has performed precision, Recall, F1 score matrix for measuring performance of SkipGram and cbow. They trained

the models in several ways by changing the vector's dimensionality and window size. He provides window size 2 and makes the vocab with 100,200,300,400 and 500 dimensional vectors of word differently. He did the same thing for window size 3 and 4. All these things did, both for extrinsic and intrinsic evaluation.

In the termination he found that wor2vec SkipGram provides the best result for both evaluation systems when vector dimension is 300 and window remains 3.

[10] R. Rahman, "Robust and Consistent Estimation of Word Embedding for Bangla Language by fine-tuning Word2Vec Model," 2020 23rd International Conference on Computer and Information Technology (ICCIT), 2020, pp. 1-6, doi: 10.1109/ICCIT51783.2020.9392738.

[12] A. A. A. Rafat and M. Salehin et al. collected a huge amount of Bengali data from several renowned Bengali newspapers. They have collected 500000 article and trained their models with 32 million of words. They choose almost every word-embedding model for training. They have use SkipGram, CBOW for both wor2vec and FastText and python library for GLOVE model. They evaluate their models with two main embedding tasks, semantic analysis and word-analogy. Among the five models they have found FastText produced the best result for almost every task.

[10] A. A. A. Rafat, M. Salehin, F. R. Khan, S. A. Hossain and S. Abujar, "Vector Representation of Bengali Word Using Various Word Embedding Model," 2019 8th International Conference System Modeling and Advancement in Research Trends (SMART), 2019, pp. 27-30, doi: 10.1109/SMART46866.2019.9117386.

[11] M. A. Al Mumin, A. A. M. Shoeb, M. R. Selim, and M. Z. Iqbal, "Sumono: A representative modern Bengali corpus," SUST Journal of Science and Technology, vol. 21, pp. 78–86, 2014

## 2.2 Challenges

The biggest for this study was collecting data. Collecting data for the Bengali language is more difficult than any other language, the reason why a few attempts have been made in the era of NLP. We consumed our 70 percent research time for data collection.

For collecting data, we chose scrapy as a web crawler which was a newborn tool for us. After collecting data another challenge was to pre-process. We did not find many resources for data preprocessing unlike English.

After preprocessing of data, we trained our models in google colab. Since we trained the models on different platforms or libraries, the resources were not as available for all the models. We have struggled to find out the glove implementation resources for the python library since the language has already been changed several times. We did not even find the proper guidance to train the glove model from scratch.

After training it was very difficult to name a model as the best embedding model for our data set since almost every has its own capability and they produce good results on that specific sector.



## **CHAPTER 3**

### **RESEARCH METHODOLOGY**

Before going to the working part, we want to discuss about some theoretical and mathematical concept. As we can understand the model's internal mechanism.

#### **3.1.1 Model's Theoretical and mathematical base**

1. Euclidean distance
2. Cosine similarity
3. SoftMax
4. Shallow neural network
5. Co-occurrence matrix
6. Window size = how many words it will consider at the training time
7. Logistic function (sigmoid)

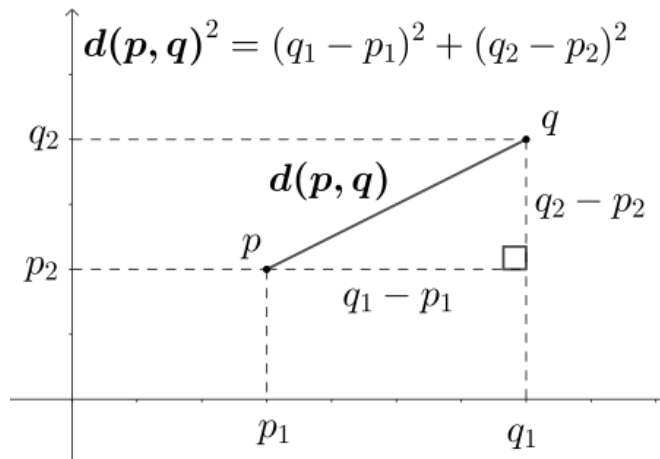


Figure 3.1.1.1: Euclidean Distance

The term Euclidean distance comes from the renowned Pythagorean theorem. It defines the distance between two or multiple points. We can find out the distance of points by using the Euclidean distance.

In computer science Euclidean distance is being used rapidly, especially in machine learning. We can find out the similarity between two objects by using this. Also, we can categorize our objects in a similar way.

Formulas of finding distance:

In above picture the distance between  $p$  and  $q$  is  $d(p, q) = |p - q|$  if we take it as a one-dimensional calculation. More appropriately

$$d(p, q) = \sqrt{(p - q)^2}$$

which will provide a positive value, always.

What happens if the dimension increases? Let's consider a two-dimensional space  $(p_1, q_1)$  and  $(p_2, q_2)$

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$$

and  $d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2}$  for  $n$  number of dimensions.

We also can get the distance of a polar coordinate system by using Euclidean distance. A polar coordinate is nothing but distance from a reference point and angle of the reference direction.

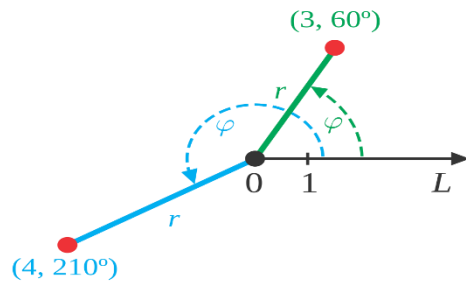


Figure 3.1.2.2: Angle of The Reference Direction.

Suppose the distance of a polar coordinate is  $p(r, \theta)$  and  $q(s, \phi)$  then the Euclidean distance will be

$$d(p, q) = \sqrt{r^2 + s^2 - 2rs \cos(\theta - \phi)}$$

This equation has been established by the law of cosine.

We can find the third side of a triangle when the two sides and angles are known. The equation of cosine law is  $a^2 = b^2 + c^2 - 2bc \cos \theta$

If the angle is 90 degrees, then the equation becomes  $a^2 = b^2 + c^2$  which is nothing but our prewrite of Pythagorean theorem.

We can get the equation in vector form, which will be like

(Here bold a indicates the vector form.

$$\|a\|^2 = \|b\|^2 + \|c\|^2 - 2\|a\| \|b\| \cos(a, b)$$

cosine similarity:

After knowing all these basic stuffs, we are able to understand the cosine similarity.

We use cosine similarity for finding the similarity of objects. Here objects mean the vector representation of any kind of data. It would be a numerical data and would be a text data's vector representation.

Basically, what cosine similarity does, it provides us the angular distance of two non-zero vectors. It has used Euclidean dot product for finding the values.

$$\text{Cosine similarity (P, Q)} = \cos(\theta) = \frac{P \cdot Q}{\|P\| \|Q\|} = \frac{\sum P_i Q_i}{\sqrt{\sum P_i^2} \sqrt{\sum Q_i^2}}$$

The value of cosine similarity always belongs to the interval -1 to 1.

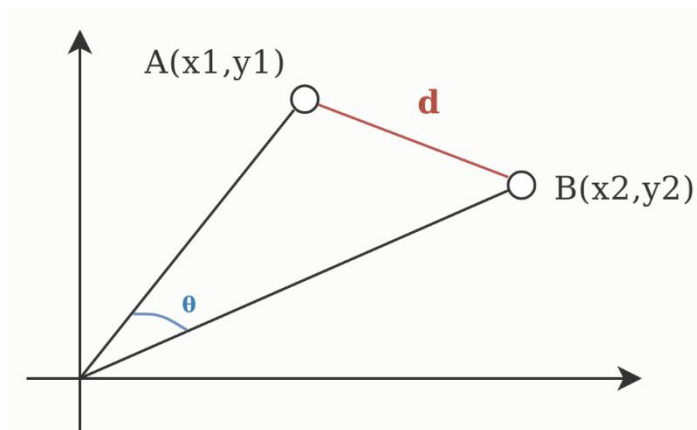


Figure 3.1.3.3: vector representation of our word

Here assume that the vector representation of our word A with value  $x_1$  and  $y_1$  and for B it's  $x_2$  and  $y_2$ .

Then we can write

vector A =  $[x_1, y_1]$  and B =  $[x_2, y_2]$

So, in this case the Euclidean distance between two vectors will be

$$\text{euclidean distance} = \sqrt{(x_1^2 + y_1^2) + (x_2^2 + y_2^2)}$$

And the cosine similarity will be,

$$\text{cosine similarity} = S_c = \cos(\theta) = \frac{x \cdot y}{\|x\| \|y\|} = \frac{x \cdot y}{\sqrt{x^2} \sqrt{y^2}}$$

Here the cosine similarity measures the similarity between two vector representations by calculating their cosine angle using above equation.

Let's play with cosine similarity and Euclidean distance with real example,

Student1 = female (1), student2 = male (0), supervisor = female(1)

Assume they all have some feature with vector representation.

Here the last value answers the question, does the person is male = 0, female = 1 or something else = 2.

Small hair, smoking, sex

Student1 [ 1.3,2.7, 1]

Student2 [ 9.7,9.9, 0]

Supervisor [ 1.3,1. 3?]

Suppose we want to predict the last value of the supervisor's sexuality against student1 and student2.

Prove 1: By using Euclidean distance,

Student1 [ 1.3, 2.7] and Supervisor [ 1.3, 1.3] is = 1.4

student2 [ 9.7, 9.9] and Supervisor [ 1.3, 1.3] is = 12.02

Prove 2: the cosine similarity will be

Student1 [ 1.3,2.7] and Supervisor [ 1.3, 1.3] is = 0.944

student2 [ 9.7,9.9] and Supervisor [ 1.3, 1.3] is = 0.999

According to the Prove 1 we can say that the supervisor is a female because the value is way more into the student1 or very close to the student1. Since the student1 is a female, the supervisor is also a female. On the other hand, student2 and the supervisor distance is more so they are not close to each other. You see it's true.

But the problem is when we took the cosine similarity for measuring the distance. We did get a bad performance. The question is why?

Well, the cosine similarity has worked with the angular distance where the magnitude of a vector is not a vital point and where the textual data come from. Actually, cosine similarity produces a valid value for text data.

The reason why in embedding cosine similarity takes place. In the skip-gram and CBOW model we measure the cosine similarity which I will explain later in this document in CBOW and skip gram part.

[16]

SoftMax:

SoftMax is a function. Which has been playing a vital role in the field of nlp. The function makes a value into a vector probability. In neural network there are input layer -><- hidden layer -><- output layer  
And in the training period both the forward and backward propagation happens until it gets an appropriate value.

The Softmax layer works as a transformer. It metamorphoses the output of the preceding layer. It changes it into a vector of probabilities.

$$p(y = j | x) = \frac{e^{x^T w_j}}{\sum_k e^{x^T w_k}}$$

Here the given vector is x and weighted vector is w

There are another kind of softmax which is known as Hierarchical softmax

Which is faster to evaluate because the time complexity become

$O(\log n)$  instead of  $O(n)$  of softmax.

It evaluates a multidimensional binary tree. It finds the targeted word or node by using the multiplication of probability of each edge till the last nodes.

After getting the last product value of the probability it has chosen the best one which is the large one.

Which is in this case I'm.

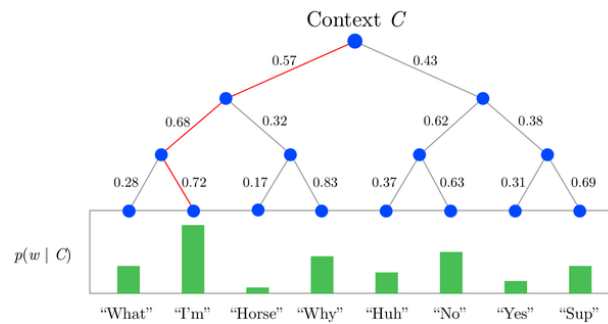


Figure 3.1.4.4: Multidimensional Binary Tree

### Shallow neural network

The shallow neural network is that kind of network which is very simple. When we think about a neural network, we imagine a huge number of hidden layers. Here shallow is different from a general neural network. Because it has only one or two hidden layers.

The skipgram or the CBOW models use that type of neural network actually.

How does it work,



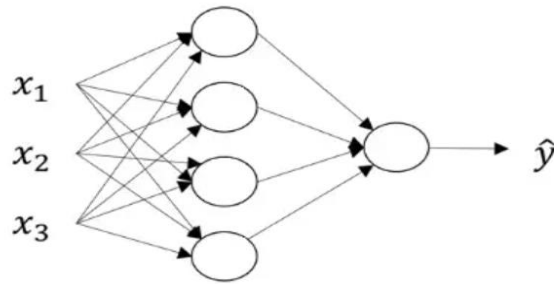


Figure 3.1.5.5: Neural Network

Here  $x_1$ ,  $x_2$ ,  $x_3$  are the inputs and  $\hat{y}$  are the output. In the middle the blank circle indicates the hidden layers.

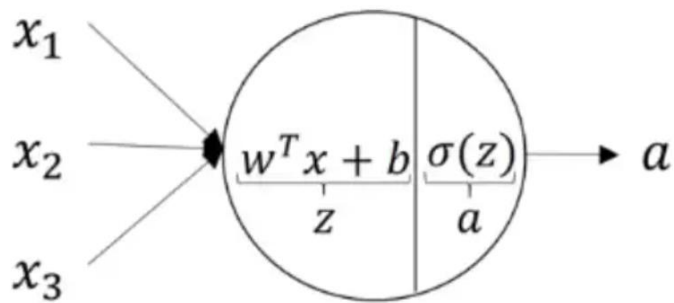


Figure 3.1.6.6: Hidden layer

Here,

$W$  = weight

$X$  = input and

$b$  = bias

The first half output is  $Z$

In second  $z$  become multiplied by an activation function. Here the picture shows the sigmoid activation function.

Mathematically it represents as,  $\sigma(x) = \frac{1}{1 + e^{-x}}$

In general, the range used for sigmoid function is 0 to 1.

Output =  $\sigma(x) * z = a$

Co-occurrence matrix:

Glove has used word co-occurrence matrix to determine the words occurrence together. After confirming the vector representation of the words, glove has made a matrix based on the word co-occurrences.

**Sigmoid:** function: [note: a logistic function or sigmoid function is a function which will map any real number into a proximity of 0 to 1]

### 3.1.2 Data collection and pre-processing

Though Bengali is one of the most spoken languages in all over the world. The textual data is not as rich as English. Also, there is no pre-trained dataset for Bengali according to our knowledge. The only places for getting textual data of Bengali language are, some blog posts and Bengali newspapers.

So, we planned to prepare our own data set. Collecting data was the biggest challenge for this study. To collect data, we made a web crawler. For implementing the web crawler, we have used scrapy since it works faster for crawling large amounts of data and complex websites. We have collected 250000 articles from a renowned Bengali newspaper of Bangladesh. Where maximum data was uncleaned, and some data was dislocating.

## **3.2 Collecting data**

Since our aim is to produce newspaper headlines for Bengali news that is why we choose the Bengali newspapers to collect our data.

For collecting data from newspapers, we have made a “web crawler” by using “SCRAPY”. Scrapy made for crawling the web pages. We choose scrapy because it is fast and secure. There are multiple functions that exist in scrapy.

A scrapy project contains,

Scrapy.cfg is the file which contains the functionality of a scrapy or it's the file which is the identification of “scrapy”. You cannot run your spider without this file. Spider is a file where we write our code for crawling the website. Another file name Pipeline, liable for handling database or you can store your crawl data into the database directly through the pipeline instead of storing it into your local disk first. Since we store our data into our local machine for security purpose there was no interaction with pipeline in this project directly.

### **3.3.1 Data preprocessing**

Preprocessing of Bengali data is a hardship work. Unlike others popular languages their id knows tools for pre-processing Bengali data. In the preprocessing task we use Regular expression, python library.

We tried to collect our data from the website more freshly without any massy information. Since the website has a complex structure, some unwanted miscellaneous data has arrived at crawling time.

Some unwanted html tag has arrived we have to remove it. There are some unwanted symbols like alpha-numeric numbers, unknown symbols, English word and letter,

numbers etc. has exist in our data. We have to remove it. To remove these kinds of miscellaneous information we used regular expression.

There were some none value inside our data and article we covert the none value with white space. We also replace some unwanted continuous data with white space.

We remove the Bengali number like

“১৭” তারিখ হতে টিকা দান কর্মসূচী.....

“১” মে জাতীয় .....

১৬ – ২০ ডিসেম্বর অব্দি বিজয় দিবস উপলক্ষে.....

which does not keep any significant change in the time of embedding. So, we remove the number data from our corpus.

We also remove the Bengali stop words, with sight help of sagor git repository. We convert our data into utf-8 format inside our scrapy tools at the time of downloading. We are also covert the data into json format at the same time.

*WebCrawler -> download data into utf-8 format -> remove html tag -> remove miscellaneous symbol -> remove Bengali numbers -> remove Bengali stop words -> data tokenization -> corpus*

After getting the clean data we start to tokenize it. We use ICU tokenizer for tokenize our data. Then we have prepared our corpus and make some copy of the corpus into different file format like .json, .text, .csv according to our need.

### **3.3.2 Computer and human languages.**

Basically, computers do not understand a single letter or word we speak or type. It does convert our text data into binary from 1 and 0. For letters it produces 1 and 0 directly but for voice first it converts the voice into letters.

Suppose if we type 'A', computer will convert it into the binary form of 'A' = 01000001. If someone says Bengali computer, then convert it into text form of 'Bengali' then into binary which is 01100010 01100101 01101110 01100111 01100001 01101100 01101001 00001010

Similarly for Bengali letters and language computers do the same process.

For আ the binary form will be = 11100000 10100110 10000110 00001010.

আল্লাহ -> 11100000 10100110 10000110 11100000 10100110 10110010 11100000  
10100111 10001101 11100000 10100110 10110010 11100000 10100110 10111110  
11100000 10100110 10111001 00001010

Well let's dive into the modern way of word representation

When a computer works with languages it is known as Natural Language Processing or NLP. It's a part of artificial intelligence or AI. The importance of NLP is increasing day by day. For this reason, Nowadays NLP has become a hot topic in the field of AI.

To solve the NLP problems the term "word embedding" has arrived.

Word embedding is the sector where the "vector representation" of a word has been discussed.

Vector representation means representing a word in vector form. By converting the human language into numerical value, we ensure the data is machine under stable.

A vector representation of a word is nothing but numbers. Word embedding is playing a significant role in the topic of NLP. Many NLP works are improving day by day because of vector representation of words in embedding. It shows that word embedding is bringing a more appreciated result in any NLP task than before.

There are multiple word embedding methods, most of the work has been done by word2vec. It has used two basic architectures for making the context into a vector: CBOW and SkipGram.

These two basic models actually the

Since the computer doesn't understand the raw text that is why it is important to convert the text into a numeric format. Embedding is one of them but not the only way. There are bunch of other way to make it namely,

- One-hot encoding
- Count-based representation and our today's topic
- Embedding

Before deep diving into embedding let's talk about one-hot encoding.

### 3.3.3 One-hot encoding

In one-hot encoding, it has made a sparse matrix for representing the word. Every element of the matrix remains zero except the targeted one. Which means we have to generate a matrix the same size as our corpus for representing each word. Assume we have a corpus size with 1 billion words then for presenting each word we have to generate a matrix, size of 1 billion rows with only a single non-zero cell.

Example:

Corpus = "soldiers" did well and they won the war"

Then convert into it a large vector ex: [.....]

In this case for representing the word "well" the one-hot encoding will make it like,

[0 0 1 0 0 0 0 0] that. Similarly, [0 0 0 1 0 0 0 0], [0 0 0 0 1 0 0 0], [0 0 0 0 0 1 0 0] for words and, they, won consequently.

Note: sparse mean remaining maximum number of zero into a matrix

### 3.4.1 Word-embedding

In embedding we intend to get the words as a numerical representation with a sound dimension and we expect the similar words will appear together in embedding space. The numeric representation of the word matrix should be a dense (a max nonzero value) matrix.

Word-embedding is a very renowned method for representing the words as number. It brings a tremendous result for any kinds of any nlp tasks. Embedding or vector representation of the words works magically in the field of nlp. Embedding models made by shallow neural networks. Shallow neural networks are neural networks which have one or two hidden layers [2]. Also embedding can be made by co-occurrence matrix [3].

Basically, embedding models work with a large amount of data. After training the model it creates a dictionary with words by representing them into  $n$  dimensional vectors. As, the contextually and globally related words remain together in the  $n$  dimensional vector space. That is why words with similar meanings remain together.

So, we can find out the semantic and syntactic words easily. Also, some word analogy task can be performed by the embedded encoded words. Finding out the nearest words and word analogy happened by measuring Euclidean distance and cosine similarity. Here is an example given below how semantic words remain together in  $n$  dimensional embedded space. The terms “NLP” and “Embedding” should stay almost together in the embedding space since they are related to each other in natural language.

Consider some false as vector dimension, suppose we get these two encoded representation for these two words where the dimension for NLP = [1 , .03 , 0.33 , 3 , 2, .....n ] and Embedding = [ 1 , 0.2 , 0.32 , 2.99, 2, ..... , n]

Look at the values, they are very much close to each other, and they will be together in embedded space. Which means the closer the dimension values will be the words will be closer to the vector space.

### 3.4.2 CBOW

The architecture “cbow” follows the feedforward NNLM [paper] model . In the training time it does train itself continuously from one word to another word according to the window size given. It will predict the middle or target word if the context words are given. After predicting the word, it will go one word forward and do the same for finding the word co-occurrences probability based on given context. Every time the forward propagation happens, by inputting the context words one-hot encoded representation in a shallow neural network. The complexity of the training for this model is  $Q = N \times D + D \times \log_2(V)$ . In this paper they have mentioned that the weight matrix of input and the projection layer is shared for all words as like NNLM. Continuous bag of word or CBOW works for word embedding technique. It is the extended version of bag of word. CBOW basically generates a targeted word after getting the input words. It means that in CBOW we input the whole words of a sentence then CBOW predicts the targeted word or that specific word.

Suppose our sentence is “সোনার বাংলা আমি তোমায় ভালবাসি”

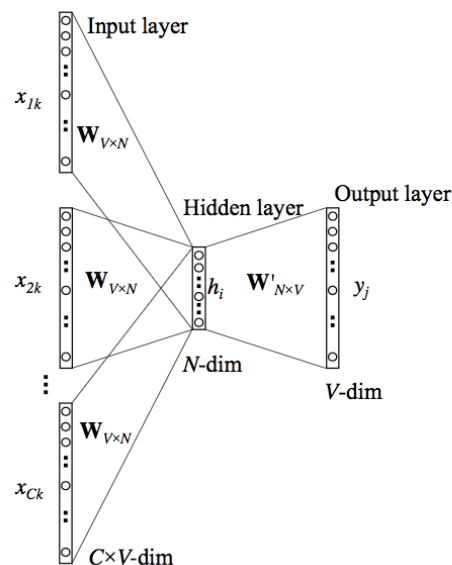


Figure 3.4.2.7: Architecture of CBOW model



If we tokenize it, the representation will be ,“সোনার” ,”বাংলা” , “আমি”, “তোমায়”, “ভালবাসি”

In CBOW if the input is: “সোনার” , “আমি”, “তোমায়”, “ভালবাসি” . Then the predicted output for CBOW will be “বাংলা”

When the windows size = 5

Assume the window = 3 . Now CBOW will take 2 words before and after of the target word as input for knowing the word “বাংলা” and its probability.

“সোনার”, “বাংলা”, “তোমায়”, “ভালবাসি” will be the input for output target word “আমি” .

In this way the whole corpus completes its training and finds out the vector representation of the word.

The Implementation of word2vec’s context window happens dynamically.

### 3.4.3 SkipGram

In SkipGram same thing happens in a contrary way. Here SkipGram predicts the context word instead of predicting the middle or current word. They have found it works better in some way than cbow but also it increases the time complexity. The training complexity then becomes  $Q = C \times (D + D \times \log_2(V))$ .

SkipGram follows the same methodology as CBOW but instead of generating a targeted word it produces the nearby words except the targeted one. Which means SkipGram takes a single word as input and output will be other nearby words of the sentence.

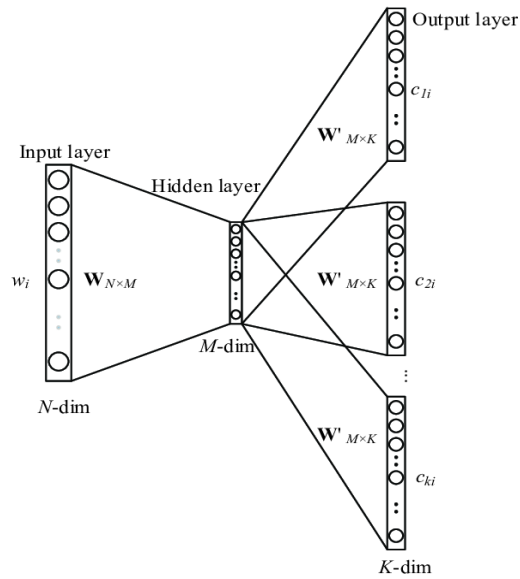


Figure 3.4.3.8: Architecture of SkipGram model

In SkipGram if the input is  $\rightarrow$  “বাংলা” then skipgram predict the output as  $\rightarrow$  “সোনার”, “আমি”, “তোমায়”, “ভালবাসি”

Note: Here in the input layer CBOW takes the one-hot representation of the context word for assuming the target word. On the other hand, SkipGram takes the target word’s one-hot representation as an input in order to get the surrounding words as output.

Where  $W$  is the weight matrix and  $N$  is the number of neurons.

### 3.4.4 FastText

Word representation and sentence classification is a complex task.

The FastText library can learn these things efficiently. In machine learning

Representation of a word is important. Miklovalt proposed a way of representing a word name word2vec. Basically, it converts the word into a vector. The FastText model

converts the words into a vector by using n-gram technique. FastText represents a single word as an n-gram. it is an advanced additional version of wor2vec.

FastText working methodology by representing a word as an n-gram

Word = army	n=2	<a, ar,rm,my,y>
Word = machine	n=3	<ma, mac, ach, chi, hin, ine, ne>
Word = embedding	n=4	<emb, embe, mbed, 32ed, eddi, ddin, ding, ing>

Table 3.4.4.1: FastText working methodology representing

The above table reveals the n-gram representation of a word by FastText when n = 2, 3 and 4 subsequently for the word's army, machine and embedding.

What is the core difference of FastText from wor2vec? The core difference of FastText model is instead of working with words it works with n-gram of letters. In the following example we will see that both CBOW and SkipGram for wor2vec are working with words. It will find out how many times the words army, machine and embedding have occurred together locally according to the window size. FastText methodology is almost the same but instead of finding out the word-word occurrences probability it will find out the letters co-occurrence probability.

Ex: when n =4

<emb, embe, mbed, bedd, eddi, ddin, ding, ing>

<emb, embe, mbed, bedd, eddi, ddin, ding, ing>

<emb, embe, mbed, bedd, eddi, ddin, ding, ing>

<emb, embe, mbed, bedd, eddi, ddin, ding, ing>

<emb, embe, mbed, bedd, eddi, ddin, ding, ing>

<emb, embe, mbed, bedd, eddi, ddin, ding, ing>

<emb, embe, mbed, bedd, eddi, ddin, ding, ing>

<emb, embe, mbed, bedd, eddi, ddin, ding, ing>

<emb, embe, mbed, bedd, eddi, ddin, ding, ing>

Suppose for the word বাংলাদেশ when  $n = 3$  FastText representation will be

<বাং, বাংলা, ংলাদে, লাদেশ, দেশ>

When FastText gets a word as same size as  $n$  or  $>n$  it will consider it as a complete word and save it into the vocabulary such as দেশ, বাংলা

Now look at the first example, after splitting the word “বাংলাদেশ” we get “দেশ”, and “বাংলা” which is a meaningful word in Bangla Language.

Here is one of the great successes of FastText’s in the land of word embedding.

Let me clear it, suppose we have a corpus where the words “দেশ”, and “বাংলা” don’t exist but the word “বাংলাদেশ” is.

So, we see that FastText can make words which don't even exist in the corpus.

Thanks to FastText for considering the root level training methodology as we can generate unknown words.

The first and last angular brackets indicate the inception and termination of the word.

### 3.4.5 Glove

Global vector or glove is another process of word embedding. Glove basically introduced by Stamford university.

Glove

Glove means global vector. In word2vec the model only looks at the local value of the words. One the other hand glove considers both the local and global representation of the words. That is why the model is also known as the extended version of wor2vec.

The whole process happens by generating a word-word co-occurrence matrix. The size of the matrix will be the same as the size of the corpus. Word-word means the matrix should have the same words in row and column, but a single word will occur only for a single time. That's mean if the size of the corpus word is X. Then after confirming the word occurrence for a single time the size of the corpus words will be  $X - T$  [here T indicates the amount of multiple word co-occurrences.]

example: corpus

I have my own little cat

You have little cat

You choose word-embedding

here,

Row = I You have choose my own little cat word-embedding.

Column = I You have choose my own little cat word-embedding

Look here we didn't take the words have, little, cat for multiple times in our matrix though it Occurs for multiple times in the corpus. Size of the matrix =  $13 * 13$  [ without reducing the double occurrence words] Size of the row =  $9 * 9$  [after reducing the double times occurrence words] Well after confirming the window size glove prepares a co-occurrence matrix according to its Occurrences probability in order to make a vector representation of the words.

Assume we are working with our above corpus. Then the co-occurrences matrix will be like, Word analogy means: when we get an answer by questing something such as: Dhaka is to Bangladesh Then what Deli is to India? [note: a logistic function or sigmoid function is a function which will map any real number into a proximity of 0 to 1]

	I	You	have	choose	my	own	little	cat	word-embedding
I	0	0	1	0	0	0	0	0	0
You	0	0	1	1	0	0	0	0	0
have	1	1	0	0	1	0	1	0	0
choose	0	1	0	0	0	0	0	0	1
my	0	0	1	0	0	1	0	0	0
own	0	0	0	0	1	0	1	0	0
little	0	0	1	0	0	1	0	2	0
cat	0	0	0	0	0	0	2	0	0
word-embedding	0	0	0	1	0	0	0	0	0

Figure 3.4.5.1: Glove Model

Where the window size = 1

- Window length 1 (more common: 5–10)
- Symmetric (irrelevant whether left or right context)
- Example corpus:
  - I like deep learning
  - I like NLP
  - I enjoy flying

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Figure 3.4.5.2: Glove Model

Now let us explain why they choose to collect both the local and global occurrences of the words in the whole corpus.

In local representation the model only considers the local context word for training the model or for getting the vector representation of the words. It does not care about the global probability of word co-occurrences together. Glove collects the global words occurrence probability. It will look the whole corpus at a time and find out how many times a word co-occurs together in all over the corpus

In above example (Stamford university cse224 lecture's example) glove collects not only the word, "deep" co-occurrences probability with the other words contextually but also it will find out the global co-occurred probability of the word by generating.

## CHAPTER 4

### RESULT AND EVALUATION

#### 4.1 Result of Different Models

we have got different types of results from different models. Among them FastText provides us a great result with a 100-dimensional vector in SkipGram. Though the time complexity for FastText is much bigger than any other models. Here gensim library for FastText showed us a satisfactory result with a less amount of time than facebook FastText library. According to the given snapshot you may see that we get the better result for syntactic analogy in word2vec than FastText. Glove provided us a great result in syntactic task hence it considers the global probability of words. For analogy task we get the best result from word2vec SkipGram. It has shown us the best result for analogy task.

**Evaluation:** We evaluate our models by finding the semantic, syntactic and word analogy task. In semantic task models should show the best matching words or word according to the given word. Because semantic words vector representation should be the same. They should stay together in the vector space. The idea is same as for finding out the syntactic word task. On the other hand, another evaluation method is nothing but summation and subtraction of vectors. Which is known as a word analogy task. In word analogy task we measure the cosine similarity. By finding out the cosine or the angle between two vectors and adding them with another vector we get a great result. We can get the relation between two words. We can find out another new word by performing the word analogy task

1. (রাজা + মেয়ে - ছেলে) the result should be “রানি”

2.(বাংলাদেশ + ইসলামাবাদ - ঢাকা) the result should be “পাকিস্তান” and we have got correct answer for majority of the models.



```

word2vec_gs_sg0_100.wv.most_similar(["কুয়াশা"] ,
[( 'বাতাস', 0.8055527210235596),
( 'শীত', 0.8016539812088013),
( 'রোদ', 0.7992282509803772),
( 'ঘন', 0.7736377716064453),
( 'কনকনে', 0.7713310122489929),
( 'ভ্যাপসা', 0.7668533325195312),
( 'বৃষ্টিও', 0.7581560611724854),
( 'ঝোড়ো', 0.7573686838150024),
( 'কুয়াশার', 0.7514734268188477),
( 'কুয়াশায়', 0.7490196228027344)]

word2vec_gs_sg0_100.wv.most_similar(["মেট্রোরেল"]
[( 'এক্সপ্রেসওয়ে', 0.8454931974411011),
( 'এলিভেটর', 0.8332984447479248),
( 'টানেল', 0.796747088432312),
( 'বিআরটি', 0.7688660621643066),
( 'মেট্রোরেলের', 0.7666299343109131),
( 'এক্সপ্রেসওয়ের', 0.7131284475326538),
( 'ফ্লাইওভার', 0.7084664702415466),
( 'এক্সপ্রেসওয়ে', 0.6966019868850708),
( 'রেললাইন', 0.6963473558425903),
( 'পদ্মাসেতু', 0.6934511661529541)]

```

Figure 4.1.1: Gensim Word2vec Cbow 100dim model

```

word2vec_gs_sg0_100.wv.most_similar(positive
=['বাংলাদেশ', 'ইসলামাবাদ'], negative=['ঢাকা'], topn=3)
[( 'পাকিস্তান', 0.7045680284500122),
( 'ভারত', 0.6577026844024658),
( 'ভারতও', 0.6489845514297485)]

word2vec_gs_sg0_100.wv.most_similar(positive
=['রাজা', 'মেয়ে'], negative=['ছেলে'], topn=3)
[( 'রানি', 0.7919169664382935),
( 'রানির', 0.7211582064628601),
( 'গায়িকা', 0.6812238097190857)]

```

Figure 4.1.2: Gensim Word2vec Cbow 100dim model

```

word2vec_gs_sg0_300.wv.most_similar(["কুয়াশা"] ,
[('শীত', 0.7571069598197937),
 ('রোদ', 0.7516466975212097),
 ('কনকনে', 0.7398843765258789),
 ('বাতাস', 0.7398055791854858),
 ('ঘন', 0.7060925960540771),
 ('কুয়াশায়', 0.7023773193359375),
 ('ভ্যাপসা', 0.6944897174835205),
 ('বোডো', 0.6934114694595337),
 ('কুয়াশার', 0.6839745044708252),
 ('স্রোত', 0.6683980822563171)]

word2vec_gs_sg0_300.wv.most_similar(["মেট্রোরেল"])
[('এক্সপ্রেসওয়ে', 0.7506396770477295),
 ('এলিভেটর', 0.7398675680160522),
 ('টানেল', 0.7287229299545288),
 ('পদ্মাসেতু', 0.6942799091339111),
 ('মেট্রোরেলের', 0.6862154603004456),
 ('ফ্লাইওভার', 0.6578311920166016),
 ('রেললাইন', 0.6275181770324707),
 ('বিআরটি', 0.6272804737091064),
 ('এক্সপ্রেসওয়ের', 0.6142202019691467),
 ('আন্ডারপাস', 0.6142109632492065)]

```

Figure 4.1.3: Gensim Word2vec Cbow 300dim model

```

word2vec_gs_sg0_300.wv.most_similar(positive
=['বাংলাদেশ', 'ইসলামাবাদ'],negative=['ঢাকা'], topn=3)
[('পাকিস্তান', 0.6200746297836304),
 ('সুইডেন', 0.5354422926902771),
 ('দেশটি', 0.5302954912185669)]

word2vec_gs_sg0_300.wv.most_similar(positive
=['রাজা', 'মেয়ে'], negative=['ছেলে'], topn=3)
[('রানি', 0.6271569728851318),
 ('রানির', 0.5631905198097229),
 ('গায়িকা', 0.543364405632019)]

```

Figure 4.1.4: Gensim Word2vec Cbow 300dim model

```

word2vec_gs_sg1_100.wv.most_similar(["কুয়াশা"] ,

[('ঘনকুয়াশা', 0.8485482931137085),
 ('ভ্যাপসা', 0.8257817625999451),
 ('গুঁড়িগুঁড়ি', 0.8187628388404846),
 ('বাতাসও', 0.7964993715286255),
 ('কুয়াশার', 0.7954361438751221),
 ('বজ্রপাতও', 0.7904238700866699),
 ('কুয়াশায়', 0.7824938297271729),
 ('ঘন', 0.7819479703903198),
 ('ঝিরিঝিরি', 0.7776706218719482),
 ('বৃষ্টিরও', 0.7746886014938354)]

word2vec_gs_sg1_100.wv.most_similar(["মেট্রোরেল"]

[('এলিভেটর', 0.8639034032821655),
 ('এক্সপ্রেসওয়ে', 0.8400706648826599),
 ('মেট্রোরেলের', 0.825241208076477),
 ('টানেল', 0.8094684481620789),
 ('এমআরটি', 0.7954422235488892),
 ('ফ্লাইওভার', 0.7847654819488525),
 ('বিআরটি', 0.7793248295783997),
 ('এক্সপ্রেসওয়ের', 0.7780054807662964),
 ('সেতু', 0.776928186416626),
 ('এক্সপ্রেসওয়ে', 0.7768104076385498)]

```

Figure 4.1.5: Gensim Word2vec SkipGram 100dim model

```

word2vec_gs_sg1_100.wv.most_similar(positive
=[ 'বাংলাদেশ', 'ইসলামাবাদ'], negative=[ 'ঢাকা'], topn=3)

[('পাকিস্তান', 0.6200746297836304),
 ('সুইডেন', 0.5354422926902771),
 ('দেশটি', 0.5302954912185669)]

word2vec_gs_sg1_100.wv.most_similar(positive
=[ 'রাজা', 'মেয়ে'], negative=[ 'ছেলে'], topn=3)

[('রানি', 0.6271569728851318),
 ('রানির', 0.5631905198097229),
 ('গায়িকা', 0.543364405632019)]

```

Figure 4.1.6: Gensim Word2vec SkipGram 100dim model

```

word2vec_gs_sg1_300.wv.most_similar(["কুয়াশা"] ,
[('ঘনকুয়াশা', 0.7270201444625854),
 ('ভ্যাপসা', 0.6956566572189331),
 ('গুঁড়িগুঁড়ি', 0.6904441714286804),
 ('কুয়াশার', 0.6828926801681519),
 ('কুয়াশায়', 0.6697243452072144),
 ('বজ্রপাতও', 0.6693300008773804),
 ('কনকনে', 0.6553128957748413),
 ('ঘন', 0.6523224115371704),
 ('বাতাসও', 0.6491879224777222),
 ('কুয়াশা', 0.6466500759124756)]

word2vec_gs_sg1_300.wv.most_similar(["মেট্রোরেল"])
[('এলিভেটর', 0.7292377948760986),
 ('মেট্রোরেলের', 0.692162036895752),
 ('এমআরটি', 0.6847621202468872),
 ('এক্সপ্রেসওয়ে', 0.6769453287124634),
 ('টানেল', 0.645955502986908),
 ('বিআরটি', 0.6294627785682678),
 ('টানেলসহ', 0.607150673866272),
 ('এক্সপ্রেসওয়ের', 0.6059511303901672),
 ('এক্সপ্রেসওয়ে', 0.6027036905288696),
 ('ফোরলেন', 0.59942227602005)]

```

Figure 4.1.7: Gensim Word2vec SkipGram 300dim model

```

word2vec_gs_sg1_300.wv.most_similar(positive
=[ 'বাংলাদেশ', 'ইসলামাবাদ'], negative=['ঢাকা'], topn=3)
[('পাকিস্তান', 0.5622924566268921),
 ('পাকিস্তানের', 0.4593409299850464),
 ('ভারত', 0.4499926269054413)]

word2vec_gs_sg1_300.wv.most_similar(positive
=[ 'রাজা', 'মেয়ে'], negative=['ছেলে'], topn=3)
[('ভাজিরালংকর্ন', 0.4837181568145752),
 ('রাজার', 0.47727203369140625),
 ('ভাজিরালংকর্ণ', 0.4726111888885498)]

```

Figure 4.1.8: Gensim Word2vec SkipGram 300dim model

```

fasttext_gs_sg1_100.wv.most_similar(["কুয়াশা"] ,
[('ঘনকুয়াশা', 0.955071747303009),
('কুয়াশাও', 0.9503843784332275),
('কুয়াশার', 0.9033924341201782),
('কুয়াশায়', 0.8972080945968628),
('ঘনকুয়াশার', 0.8786872625350952),
('কুয়াশাচ্ছন্ন', 0.8405905961990356),
('ঘন', 0.799871027469635),
('গুঁড়িগুঁড়ি', 0.7985829710960388),
('মেঘও', 0.7942732572555542),
('ভ্যাপসা', 0.789848804473877)]

fasttext_gs_sg1_100.wv.most_similar(["মেট্রোরেল"])
[('মেট্রোরেলও', 0.9614615440368652),
('মেট্রোরেলে', 0.9439646005630493),
('মেট্রোরেলের', 0.9262422323226929),
('মেট্রোরেলসহ', 0.9020270109176636),
('এলিভেটেড', 0.8451743721961975),
('এক্সপ্রেসওয়ে', 0.7714332938194275),
('ফ্লাইওভার', 0.7673085927963257),
('বিআরটি', 0.7668994665145874),
('এলিভেটর', 0.7664886116981506),
('এমআরটি', 0.764568567276001)]

```

Figure 4.1.9: Gensim Fasttext SkipGram 100dim model

```

fasttext_gs_sg1_100.wv.most_similar(positive
=['বাংলাদেশ', 'ইসলামাবাদ'], negative=['ঢাকা'], topn=3)
[('পাকিস্তান', 0.7125189304351807),
('পাকিস্তানও', 0.6797451972961426),
('ভারত', 0.6759352684020996)]

fasttext_gs_sg1_100.wv.most_similar(positive
=['রাজা', 'মেয়ে'], negative=['ছেলে'], topn=3)
[('সালতামামি', 0.7117267847061157),
('রাজার', 0.7096242308616638),
('রানি', 0.7072427868843079)]

```

Figure 4.1.10: Gensim Fasttext SkipGram 100dim model

```

fasttext_gs_sg1_300.wv.most_similar(["কুয়াশা"] ,
[('ঘনকুয়াশা', 0.9462637901306152),
('কুয়াশাও', 0.933032214641571),
('কুয়াশায়', 0.8841840028762817),
('কুয়াশার', 0.8789616823196411),
('ঘনকুয়াশার', 0.8421176671981812),
('কুয়াশাচ্ছন্ন', 0.7924590110778809),
('ঘন', 0.6507999300956726),
('ঝড়ঝুঁটি', 0.6424121856689453),
('গুঁড়িগুঁড়ি', 0.6393018364906311),
('কনকনে', 0.6302749514579773)]

fasttext_gs_sg1_300.wv.most_similar(["মেট্রোরেল"])
[('মেট্রোরেলও', 0.9612268209457397),
('মেট্রোরেলে', 0.9483367204666138),
('মেট্রোরেলের', 0.923438549041748),
('মেট্রোরেলসহ', 0.9031366109848022),
('মেট্রোর', 0.7218939065933228),
('এলিভেটরে', 0.6957094669342041),
('এক্সপ্রেসওয়ে', 0.6483370661735535),
('এমআরটি', 0.6246091723442078),
('টানেল', 0.621452808380127),
('এক্সপ্রেসওয়ের', 0.6150929927825928)]

```

Figure 4.1.11: Gensim Fasttext SkipGram 300dim model

```

fasttext_gs_sg1_300.wv.most_similar(positive
=['বাংলাদেশ', 'ইসলামাবাদ'], negative=['ঢাকা'], topn=3)
[('দিল্লিও', 0.5941457748413086),
('দিল্লী', 0.5499972105026245),
('দিল্লিহ', 0.5470802187919617)]

fasttext_gs_sg1_300.wv.most_similar(positive
=['রাজা', 'মেয়ে'], negative=['ছেলে'], topn=3)
[('রাজার', 0.5719311833381653),
('রাজাহার', 0.5538002252578735),
('রাজারা', 0.5467007160186768)]

```

Figure 4.1.12: Gensim Fasttext SkipGram 300dim model

```

fasttext_gs_sg0_100.wv.most_similar(["কুয়াশা"] ,

[('ঘনকুয়াশা', 0.9711394906044006),
 ('কুয়াশাও', 0.9310247898101807),
 ('কুয়াশায়', 0.8343454599380493),
 ('কুয়াশার', 0.8266347646713257),
 ('ঘনকুয়াশার', 0.7966285943984985),
 ('ধোঁয়াশা', 0.7893217206001282),
 ('ধোঁয়াশা', 0.767272412776947),
 ('মেঘরোদ', 0.7500921487808228),
 ('তুয়াশা', 0.7469527721405029),
 ('কুয়াশাচ্ছন্ন', 0.7451668977737427)]

fasttext_gs_sg0_100.wv.most_similar(["মেট্রোরেল"]

[('মেট্রোরেলও', 0.9788670539855957),
 ('মেট্রোরেলে', 0.9297658205032349),
 ('মেট্রোর', 0.8885155916213989),
 ('মেট্রোরেলের', 0.8737552762031555),
 ('মেট্রো', 0.8578785061836243),
 ('চট্টমেট্রো', 0.8264012336730957),
 ('মেট্রোপোল', 0.825806736946106),
 ('মেট্রো', 0.8175790309906006),
 ('মেট্রোরেলসহ', 0.8122009038925171),
 ('মেট্রোলিটন', 0.8017096519470215)]

```

Figure 4.1.13: Gensim Fasttext Cbow 100dim model

```

fasttext_gs_sg0_100.wv.most_similar(positive
=[ 'বাংলাদেশ', 'ইসলামাবাদ'], negative=[ 'ঢাকা'], topn=3)

[('পাকিস্তান', 0.7135821580886841),
 ('পাকিস্তানভিত্তিক', 0.7052025198936462),
 ('পূর্বপাকিস্তান', 0.697191059589386)]

fasttext_gs_sg0_100.wv.most_similar(positive
=[ 'রাজা', 'মেয়ে'], negative=[ 'ছেলে'], topn=3)

[('রাজামৌলি', 0.7583073377609253),
 ('রাজকন্যা', 0.7431219816207886),
 ('রাজাই', 0.7378509640693665)]

```

Figure 4.1.14: Gensim Fasttext Cbow 100dim model

```

fasttext_gs_sg0_300.wv.most_similar(["কুয়াশা"] ,
[('ঘনকুয়াশা', 0.9639012813568115),
('কুয়াশাও', 0.914515495300293),
('কুয়াশায়', 0.8360567092895508),
('কুয়াশার', 0.83012855052948),
('ঘনকুয়াশার', 0.7942291498184204),
('কুয়াশাচ্ছন্ন', 0.748650074005127),
('ধোয়াশা', 0.747927725315094),
('তুয়াশা', 0.7375349998474121),
('ধোয়াশা', 0.7273876070976257),
('কুয়ারন', 0.6622710227966309)]

fasttext_gs_sg0_300.wv.most_similar(["মেট্রোরেল"])
[('মেট্রোরেলও', 0.9706897735595703),
('মেট্রোরেল', 0.9232302904129028),
('মেট্রোর', 0.8828109502792358),
('মেট্রোরেলের', 0.8762004971504211),
('মেট্রো', 0.8502136468887329),
('মেট্রোপোল', 0.8210102319717407),
('মেট্রোরেলসহ', 0.8130447864532471),
('মেট্রো', 0.802211582660675),
('চট্টমেট্রো', 0.7979769706726074),
('মেট্রোলিটন', 0.7841386795043945)]

```

Figure 4.1.15: Gensim Fasttext Cbow 300dim model

```

fasttext_gs_sg0_300.wv.most_similar(positive
=['বাংলাদেশ', 'ইসলামাবাদ'], negative=['ঢাকা'], topn=3)
[("ঢাকা'র", 0.6171178817749023),
('দিল্লিও', 0.5736168622970581),
('ঢাকায়ও', 0.5554546117782593)]

fasttext_gs_sg0_300.wv.most_similar(positive
=['রাজা', 'মেয়ে'], negative=['ছেলে'], topn=3)
[('রাজাবাবু', 0.7027453184127808),
('রানীজান', 0.6932054758071899),
('রাভু', 0.6830580234527588)]

```

Figure 4.1.16: Gensim Fasttext Cbow 300dim model



```

fasttext_sg1_100.get_nearest_neighbors("কুয়াশা")

[(0.9573663473129272, 'কুয়াশাও'),
 (0.9519394636154175, 'ঘনকুয়াশা'),
 (0.9419339895248413, 'কুয়াশার'),
 (0.9377858638763428, 'কুয়াশায়'),
 (0.8823006749153137, 'কুয়াশাচ্ছন'),
 (0.8077007532119751, 'ঘন'),
 (0.797359824180603, 'মেঘাচ্ছন'),
 (0.7923212051391602, 'শীত'),
 (0.7809111475944519, 'রোদ'),
 (0.7800885438919067, 'বাতাসও')]

fasttext_sg1_100.get_nearest_neighbors("মেট্রোরেল")

[(0.9716010093688965, 'মেট্রোরেলে'),
 (0.9530009627342224, 'মেট্রোরেলের'),
 (0.9247152209281921, 'মেট্রোরেলসহ'),
 (0.8415833115577698, 'মেট্রোর'),
 (0.8325338363647461, 'টানেল'),
 (0.8243909478187561, 'এলিভেটর'),
 (0.7510474324226379, 'টানেলটি'),
 (0.7493442296981812, 'এক্সপ্রেসওয়ে'),
 (0.7348825335502625, 'প্রকল্পও'),
 (0.7330422401428223, 'রেললাইনটি')]

```

Figure 4.1.17: Fasttext SkipGram 300dim model

```

fasttext_sg1_100.get_analogies("রাজা", "ছেলে", "মেয়ে")

[(0.7303565144538879, 'রাজামৌলি'),
 (0.7001459002494812, 'রাজামৌলির'),
 (0.6951875686645508, 'সুস্তা'),
 (0.6933155059814453, 'ওবেরয়'),
 (0.6886851191520691, 'বিজুরী'),
 (0.6832139492034912, 'মরমি'),
 (0.6831360459327698, 'রাহানে'),
 (0.6774823069572449, 'ইয়োহানি'),
 (0.6752798557281494, 'মেয়েলি'),
 (0.673197329044342, 'সিকার')]

fasttext_sg1_100.get_analogies("ঢাকা", "ইসলামাবাদ", "বাংলাদেশ")

[(0.6285935044288635, 'চট্টগ্রাম'),
 (0.6168743371963501, "'ঢাকা"),
 (0.6159656047821045, 'চট্ট'),
 (0.6057689785957336, 'চট্টগ্রামকে'),
 (0.5886152386665344, 'বিআরবি'),
 (0.5828224420547485, 'চট্টগ্রামভিত্তিক'),
 (0.5753381252288818, 'চট্টগ্রামমুখী'),
 (0.5749698281288147, 'স্টার্জন'),
 (0.574686586856842, 'উইমেন্সের'),
 (0.5703926086425781, 'বারডেম')]

```

Figure 4.1.18: Fasttext SkipGram 300dim model

```

fasttext_cbow_100.get_nearest_neighbors("কুয়াশা")

[(0.9329683184623718, 'ঘনকুয়াশা'),
 (0.9246459603309631, 'কুয়াশাও'),
 (0.8581473231315613, 'কুয়াশার'),
 (0.8538721203804016, 'কুয়াশায়'),
 (0.8070160746574402, 'কুয়াশাচ্ছন্ন'),
 (0.766051173210144, 'ধোয়াশা'),
 (0.7638098001480103, 'কুয়া'),
 (0.7076551914215088, 'ধোয়াশা'),
 (0.6871768832206726, 'কুয়াকাটা'),
 (0.6381382942199707, 'কুয়ার')]

fasttext_cbow_100.get_nearest_neighbors("মেট্রোরেল")

[(0.9139463305473328, 'মেট্রোরেলে'),
 (0.8880037069320679, 'মেট্রোরেলের'),
 (0.8830960392951965, 'মেট্রোর'),
 (0.8736748099327087, 'মেট্রো'),
 (0.8355345726013184, 'মেট্রোরেলসহ'),
 (0.7670664191246033, 'মেট্রো'),
 (0.7185729146003723, 'মেট্রিকটন'),
 (0.6821022033691406, 'মেট্রিক'),
 (0.6774435639381409, 'মেট্রোপলিটন'),
 (0.6760607957839966, 'পেট্রো')]

```

Figure 4.1.19: Fasttext Cbow 100dim model

```

[13] fasttext_cbow_100.get_analogies("বাংলাদেশ","ঢাকা","ইসলামাবাদ")

[(0.6909371018409729, 'ইসলামাবাদে'),
 (0.6885465979576111, 'ইসলামাবাদের'),
 (0.6576516628265381, 'পাকিস্তান'),
 (0.6211831569671631, 'আফগানিস্তান'),
 (0.6204364895820618, 'তাজিকিস্তান'),
 (0.6116588711738586, 'পাকিস্তানও'),
 (0.6104521155357361, 'রাষ্ট্রটি'),
 (0.6098083257675171, 'পাকিস্তান'),
 (0.6094101667404175, 'পাকিস্তানী'),
 (0.6061508059501648, 'ভারতও')]

[14] fasttext_cbow_100.get_analogies("রাজা","ছেলে","মেয়ে")

[(0.7303565144538879, 'রাজামৌলি'),
 (0.7001459002494812, 'রাজামৌলির'),
 (0.6951875686645508, 'সুপ্তা'),
 (0.6933155059814453, 'গুবেরয়'),
 (0.6886851191520691, 'বিজরী'),
 (0.6832139492034912, 'মরমি'),
 (0.6831360459327698, 'রাহানে'),
 (0.6774823069572449, 'ইয়োহানি'),
 (0.6752798557281494, 'মেয়েলি'),
 (0.673197329044342, 'সিকার')]

```

Figure 4.1.20: Fasttext Cbow 100dim model

```

model_glove.most_similar("কুয়াশা", number=10)

[('ঘন', 0.8687427331547395),
 ('বারছে', 0.8162763349998646),
 ('কুয়াশা', 0.8013554908067106),
 ('রোদ', 0.7791172893446027),
 ('পড়তে', 0.7493537134758462),
 ('বাতাস', 0.7272724270286693),
 ('গুঁড়ি', 0.7163666199966546),
 ('ভারী', 0.6936587902713859),
 ('হালকা', 0.6827147538190723)]

model_glove.most_similar("মেট্রোরেল", number=10)

[('নলকা', 0.7272398636506434),
 ('বহুমুখী', 0.7190993992664826),
 ('টানেল', 0.7186439139120673),
 ('পূর্বপাড়', 0.7131852162301778),
 ('রেললাইন', 0.7105248446037318),
 ('সেতু', 0.6977446039315477),
 ('বিদ্যুৎকেন্দ্র', 0.663675926628448),
 ('কেন্দ্রটি', 0.6617114144377974),
 ('কর্ণফুলী', 0.6489573586959178)]

```

Figure 4.1.21: Glove 300dim model

```

model_glove.most_similar("মহিলা", number=10)

[('সংরক্ষিত', 0.7374891682891751),
 ('চাকরিজীবী', 0.7204833131045878),
 ('যুব', 0.7150699880088335),
 ('মুফাসীর', 0.700035850936929),
 ('বদরুন্নেসা', 0.684775710889716),
 ('আদিবাসী', 0.6735624801740926),
 ('সংহতির', 0.6698220807546856),
 ('স্বচ্ছসেবক', 0.6662303105024188),
 ('বুড়ি', 0.662285475573393)]

model_glove.most_similar("রাজা", number=10)

[('সিকান্দার', 0.7689530613030343),
 ('রাতুল', 0.7586055491091518),
 ('আনিস', 0.7542339620810675),
 ('কাওসার', 0.7407827887055306),
 ('সাব্বির', 0.7401117073656571),
 ('শাকিল', 0.715671832809004),
 ('টিটু', 0.7148423152530882),
 ('রাফি', 0.714594931817508),
 ('শিমুল', 0.7140578338473677)]

```

Figure 4.1.22: Glove 300dim model

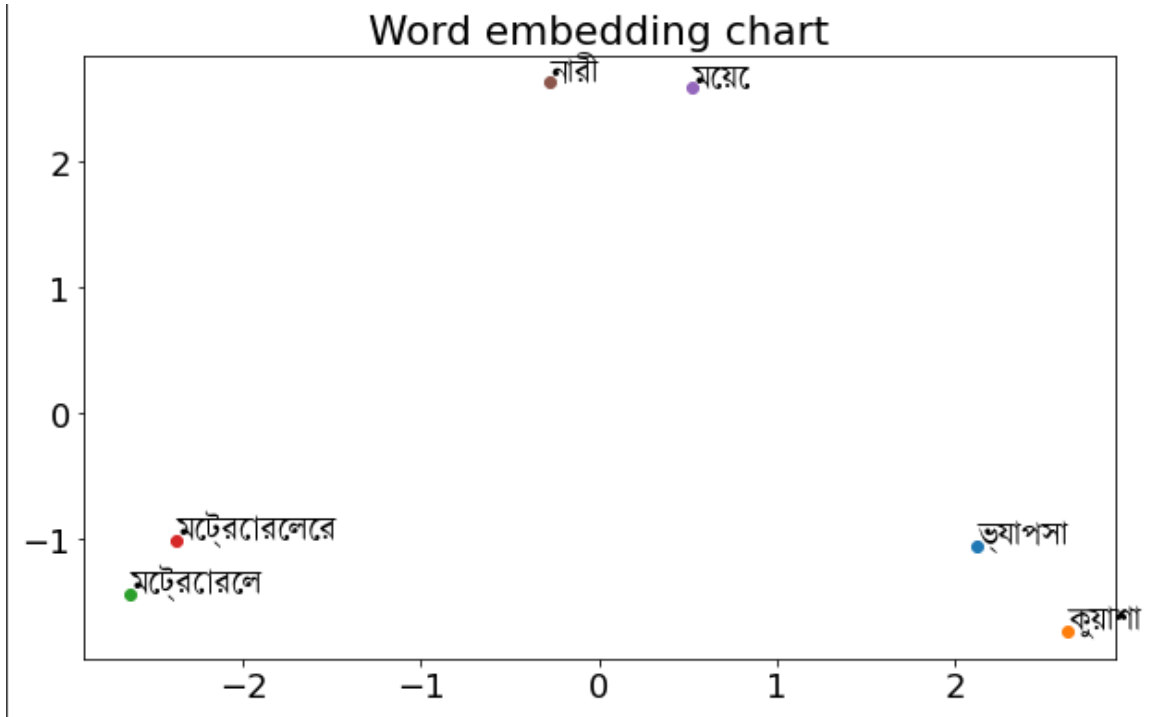


Figure 4.1.23: Word in Embedding Space

## **CHAPTER 5**

### **IMPACT ON SOCIETY, ENVIRONMENT AND SUSTAINABILITY**

#### **5.1 Impact on Society**

Finding out the best word embedding model will provide a great impact on society. This experimental research for bengali language finds out the best word-embedding model for and the training platforms for it. Since embedding is the prerequisite for many important NLP tasks hence finding out the best one will improve the NLP task. If we can ensure the specific embedding model, more research will easily occur in this sector. This research has the same motivation. Improving the NLP tasks for the Bengali language is necessary since we are living in the age of technology which is based on computers and the internet. We believe that our work will increase the NLP task on bengali language and will keep a great impact on bengali internet users.

#### **5.2 Impact on Environment**

Though our study has no direct connection to the environment, a collateral connection exists. Since bengali word has a low resource on the internet but a great amount of internet users. A big portion of the users are newly here and do not have a sound knowledge on foreign languages, namely English. Our work will increase the NLP task on bengali language which is going to improve the bengali language usability on the internet.

It will reduce the training time for many researchers which may cause low heat generation. Many giant companies can reduce their training time and improve their tools like voice command for bengali search or text command for it. Internet users for the Bengali language will enjoy the flexibility of the internet who do not understand the English language. As a result, they can get their desired information with a low amount

of time which indicates a proper use of the internet for bengali language holders. Which will keep a sound impact on environment.

### **5.3 Ethical Aspects**

1. To introduce a vast amount of bengali data on the internet publicly
2. To improve the user's experience on the internet for Bengali language.
3. To increase the user security and privacy
4. To increase the bengali internet user flexibility
5. To improve the flexibility of the users who does not know much about English
6. To introduce the appropriate data collection policy to the others
7. To ensure the users privacy and knowledge of internet
8. To secure the intellectual property of bengali language

### **5.4 Sustainability Plan**

In our study we have trained our models with two lakhs of bengali articles, containing 465211 unique words. We planned to train our models with billions of words in the near future for more appropriate results. Also, we think our study will improve various NLP tasks namely,

- Generating newspaper headline in abstract way (abstractive text summarization)
- Name entity recognition
- Sentiment analysis
- POS tagging

Also, we think our study will help the new researcher who planned to work with the bengali language. Consequently, we make our data set public. It will help the researcher to do research on bengali language without concerns about data set and data preprocessing. We open the door to think about the GLOVE model since it provides us

the worst result in some cases though theoretically it should provide us the best result. Hence GLOVE considers both the local and global probability in training time. We assume GLOVE will provide us a better result when the dataset will be larger than now.

### **Data collection:**

Though Bengali is one of the most spoken languages in all over the world. The textual data is not as rich as English. Also, there is no pre-trained dataset for Bengali according to our knowledge. The only places for getting textual data of Bengali language are, some blog posts and Bengali newspapers.

So, we planned to prepare our own data set. Collecting data was the biggest challenge for this study. To collect data, we made a web crawler. For implementing the web crawler, we have used scrapy since it works faster for crawling large amounts of data and complex websites. We have collected 250000 articles from a renowned Bengali newspaper of Bangladesh. Where maximum data was uncleaned, and some data was dislocating.

Here we did not deal with JavaScript data, hence we don't have to use tools like selenium web driver or chrome web driver.

In this section we will talk about the data collection methodology and our procedure below,

- 1. Websites choose:** It was a great challenge for us get a huge amount of data from the internet. So, choosing the best website was a challenge for me.
- 2. Newspaper archive link extracting:** for extracting the newspaper article and follow the next page we use Rule Extractor and xpath. We convert the relative URL into absolute and vice versa. To follow the next page, we use date and travers into the back and follow the previous page or date. Scrapy use "Rule" which contains Link Extractor for following the next pages.

3. **Request for data:** We get the article link by Rule and then we use a python generator and yield the request into the scrapy engine. We use the scheduler for scheduling the request according to the request.

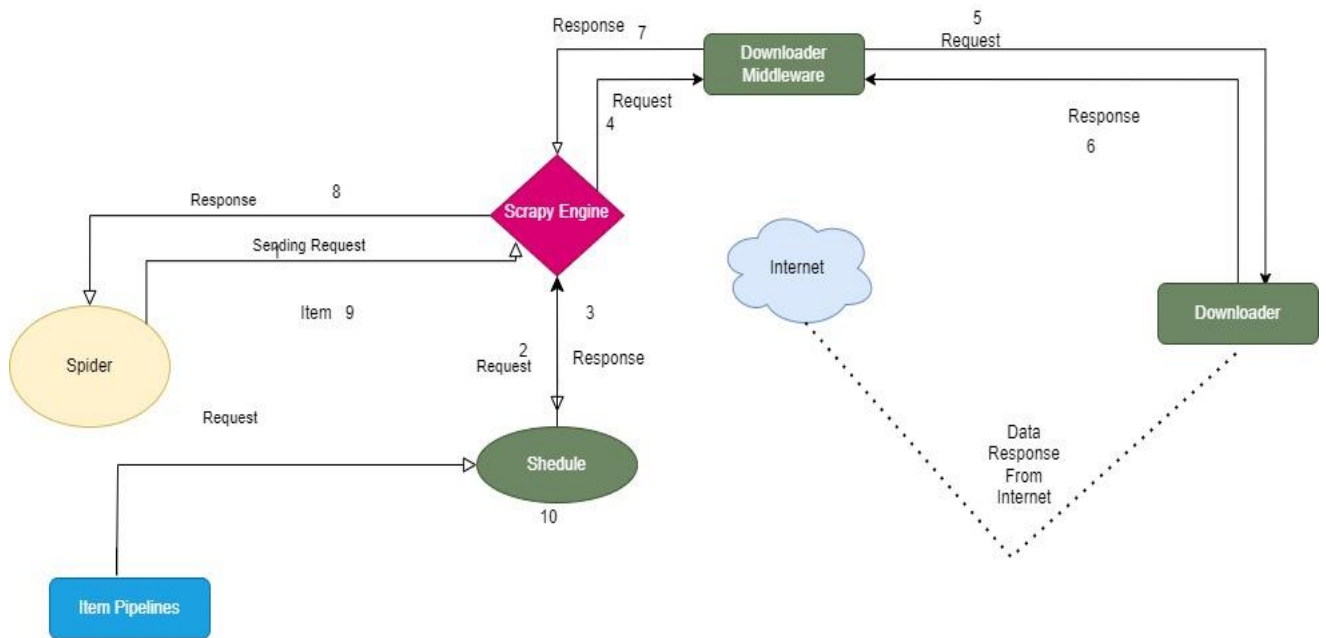


Figure 5.4.1: Dataflow of The Scrapy of Spider

1. HTML response: downloader sends us the HTML data or response into the Spider, now we again select the xpath for getting the textual data from the article, headline and summary.
2. In the time of data storing, we remove the html tag from the raw data and store only textual data.
3. We store our textual data into the local machine in JSON format.



## **CHAPTER 6**

### **CONCLUSION AND FUTURE WORK**

#### **6.1 Summary and conclusion**

This is an experimental study. Our plan was to find out the best word embedding model and the training platforms. In our study, according to the evaluation results of the models we can say that every model has its own magical performance according to the given data set. FastText provided us the best result with skipgram when we trained the data with 100 dimensions for semantic word evaluation task. We can get the best synonyms by using the FastText model. But on the other hand other models like, word2vec glove provide us best syntactic words in the time of model evaluation. Also, wor2vec provides the best performance for word analogy tasks than FastText. Since GLOVE considers both the global and local or contextual probability it provides us a tremendous result for syntactic evaluation.

Here we trained the same models in several platforms like, gensim, python and facebook FastText. Consequently, we have got the FastText with 100 dimensions that showed a better result than any other models. It is demonstrate that this model took a less time for model training in gensim package compared to facebook FastText library.

In this study we found that FastText model with architecture skipgram showing us the best result among rest of the models. We trained the models with different dimension and window size.

#### **6.2 Future work**

In this study we work with 27825539 words which is containing 465211 unique words. Which is a large amount of data for training the model but not maximum. In our study we see the glove showing as the worst result than any other embedding model, though gloves

consider both local and global word co-occurrence probability. Hence, we expect a great result from gloves, but we did not find it. So, our plan is to train the models with more data, perhaps with billions of words. Then we will compare the models with each other again. We also plan to do experiments with several NLP tasks with our embed data. As we can compare the model perfectly from each other. We also want to figure out which embedding model is better for what types of NLP tasks.

## REFERENCES

- [1] Quoc V. Le, Tomas Mikolov. "Distributed Representations of Sentences and Documents".arXiv:1802.0683
- [2] Mikolov, T., and J. Dean. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems (2013). arXiv:1310.4546v1.
- [3] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation." EMNLP. Vol. 14. 2014.
- [4] Piotr Bojanowski, Edouard Grave, Armand Jolanta's Mikolov. Enriching Word Vectors with Sub word Information. Transactions of the Association for Computational Linguistics (2017) 5: 135–146.
- [5] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
- [5]Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 (cs)
- [6] A neural lemmatizer for Bengali word embedding has been proposed by Abhushek et al in 2016 where they have used wor2vec model.
- [7] Adnan Ahmed and Ruhul Amin have collected a large amount of Bengali data for document classification problems from 13 different Bengali newspapers in 2016. Where the vector representation of Bengali word done by using word2vec model.
- [8] Ritu et al try to analyze the performance of various embedding models in 2018 for Bengali language. They perform word2vec in two different platforms, gensim and
- [9] N.Sadman, A. Sadmanee, M. I. Tanveer, M. A. Amin and A. A. Ali, "Intrinsic Evaluation of Bangla Word Embeddings," *2019 International Conference on Bangla Speech and Language Processing (ICBSLP)*, 2019, pp. 1-5, doi: 10.1109/ICBSLP47725.2019.201506.
- [10] R. Rahman, "Robust and Consistent Estimation of Word Embedding for Bangla Language by fine-tuning Word2Vec Model," *2020 23rd International Conference on Computer and Information Technology (ICCIT)*, 2020, pp. 1-6, doi: 10.1109/ICCIT51783.2020.9392738.
- [11] with it. [11] M. A. Al Mumin, A. A. M. Shoeb, M. R. Selim, and M. Z. Iqbal, "Sumono: A representative modern Bengali corpus," *SUST Journal of Science and Technology*, vol. 21, pp. 78–86, 2014
- [11] M. A. Al Mumin, A. A. M. Shoeb, M. R. Selim, and M. Z. Iqbal, "Sumono: A representative modern Bengali corpus," *SUST Journal of Science and Technology*, vol. 21, pp. 78–86, 2014
- [12] A. A. A. Rafat, M. Salehin, F. R. Khan, S. A. Hossain and S. Abujar, "Vector Representation of Bengali Word Using Various Word Embedding Model," *2019 8th International Conference System Modeling and Advancement in Research Trends (SMART)*, 2019, pp. 27-30, doi: 10.1109/SMART46866.2019.9117386.

[13]Tomas Mikolov, Kai Chen, Greg Corrado, Dean Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781

[14] Bengali language: [https://en.wikipedia.org/wiki/Bengali language](https://en.wikipedia.org/wiki/Bengali_language)].

[15] Ali, Md Nawab Yousuf, et al. "Morphological analysis of bangle words for universal networking language." Digital Information Management, 2008. ICDIM 2008. Third International Conference on. IEEE, 2008.

[16] <https://cmry.github.io/notes/Euclidean-v-cosine>

## Embedding MOdel

### ORIGINALITY REPORT

14%

SIMILARITY INDEX

13%

INTERNET SOURCES

8%

PUBLICATIONS

8%

STUDENT PAPERS

### PRIMARY SOURCES

1	<a href="https://dspace.daffodilvarsity.edu.bd:8080">dspace.daffodilvarsity.edu.bd:8080</a> Internet Source	4%
2	Submitted to Daffodil International University Student Paper	2%
3	Aysha Akther, Md. Shymon Islam, Hafsa Sultana, A.K.Z Rasel Rahman, Sujana Saha, Kazi Masudul Alam, Rameswar Debnath. "Compilation, Analysis and Application of a Comprehensive Bangla Corpus KUMono", IEEE Access, 2022 Publication	1%
4	<a href="http://www.saujs.sakarya.edu.tr">www.saujs.sakarya.edu.tr</a> Internet Source	1%
5	<a href="http://www.researchgate.net">www.researchgate.net</a> Internet Source	1%
6	<a href="http://www.cse.iub.edu.bd">www.cse.iub.edu.bd</a> Internet Source	1%
7	<a href="http://www.sciencegate.app">www.sciencegate.app</a> Internet Source	<1%