DOCUMENT CLUSTERING ON COLLECTION OF MULTIDISCIPLINARY ACADEMIC TEXTS: EXPLORING DOCUMENT EMBEDDINGS AND CLUSTERING TECHNIQUES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MURAT KARA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

AUGUST 2022

Approval of the thesis:

DOCUMENT CLUSTERING ON COLLECTION OF MULTIDISCIPLINARY ACADEMIC TEXTS: EXPLORING DOCUMENT EMBEDDINGS AND CLUSTERING TECHNIQUES

submitted by **MURAT KARA** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil KALIPÇILAR
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit OĞUZTÜZÜN
Head of Department, **Computer Engineering** _____

Prof. Dr. Pınar KARAGÖZ
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. İsmail Hakkı TOROSLU
Computer Engineering, METU _____

Prof. Dr. Pınar KARAGÖZ
Computer Engineering, METU _____

Assist. Prof. Dr. Burcu Yılmaz
Institute of Information Technologies, Gebze Technical University _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Murat Kara

Signature :

ABSTRACT

DOCUMENT CLUSTERING ON COLLECTION OF MULTIDISCIPLINARY ACADEMIC TEXTS: EXPLORING DOCUMENT EMBEDDINGS AND CLUSTERING TECHNIQUES

Kara, Murat
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. Pınar KARAGÖZ

August 2022, 155 pages

A large collection of unstructured documents is mostly not comprehensible enough at first stage for people who need to analyse and evaluate these documents. Before working on these pile of documents, the first step is to group by their similarity. Clustering algorithms fit best for these kinds of batch job. Also, nowadays, by using embedding techniques, we could represent the words, sentences, and documents as multi-dimensional vectors. So, we can compare and cluster those documents using these representations.

In this thesis, we have run embedding techniques to cluster multidisciplinary papers from different contexts. A collection of papers of academicians from Cost Action CA18110 project is clustered by different clustering methods, Agglomerative, K-Means and DBSCAN according to text representations created using distributed text representation techniques. The best methods chosen based on the experiments on the mentioned dataset are run on one small and one big text collection which contain scientific papers as well and the results are observed. While doing that, the hyper parameters of different embedding techniques were optimized to get better cluster-

ing results by applying several trials. Also, the parameters of the clustering methods are tried to be optimized to give better clustering result according to silhouette score. With a visualisation and experiment tool, lots of wise trials are performed, the results are evaluated with different clustering scores and visualised with dimensionality reduction techniques.

Keywords: Document Clustering, Bayesian Optimization, Word Embedding, Document Embedding

ÖZ

DİSİPLİNLERARASI AKADEMİK METİN KOLEKSİYONUNUN KÜMELERE AYRILMASI: DÖKÜMAN TEMSİL VEKTÖRLERİ VE KÜMELEME YÖNTEMLERİNİN ARAŞTIRILMASI

Kara, Murat
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Pınar KARAGÖZ

Ağustos 2022 , 155 sayfa

Üzerinde çalışılmamış büyük çaplı döküman kümeleri, onları incelemek ve değerlendirmek isteyen insanlar tarafından ilk aşamada yeterince anlamlandırılamazlar. Bu dökümanların üzerinde çalışmanın ilk aşaması bu dökümanları yakınlıklarına göre kümelere ayırmaktır. Kümeleme algoritmaları, bu tür yığın işler için en uygun yöntemdir. Ayrıca günümüzde, metin temsil algoritmalarıyla kelimeleri, cümleleri hatta dökümanları çok boyutlu vektörler olarak temsil edebiliyoruz. Bu vektörleri birbiriyle kıyaslayıp ona göre gruplama imkanına sahibiz.

Bu tezin kapsamında, metin temsil algoritmaları koşturularak farklı içeriklere sahip, disiplinlerarası makale koleksiyonları üzerinden algoritmaların performansları değerlendirildi. Cost Action CA18110 projesindeki akademisyenlerin makalelerinden oluşan bir veri koleksiyonu; farklı kümeleme yöntemleri ile (Agglomerative, K-Means, DBSCAN), birçok metin temsil algoritmasının ürettiği metin temsil vektörlerine göre kümelere ayrıldı. Yöntemlerden, bahsedilen veri setinin üzerinde en iyi performansla çalışanlar seçilerek, bilimsel makaleler içeren bir tanesi daha küçük, diğeri daha bü-

yük veri setleri üzerinde çalıştırılarak, sonuçlar gözlendi. Bunu yaparken, kümeleme sonuçlarını, silhouette skoruna göre en iyi hale getirmek üzere farklı metin temsil algoritmalarının hiper-parametreleri, birçok farklı denemelerle optimize edildi. Bunun için geliştirilen bir araçla, pek çok deney yapıldı, sonuçlar farklı ölçütlere göre değerlendirildi ve deney sonuçları görselleştirildi.

Anahtar Kelimeler: Metin Kümeleme, Bayes Yöntemi ile Optimizasyon, Kelime Temsili, Metin Temsili

To my family and nation

ACKNOWLEDGMENTS

The author wishes to express his deepest gratitude to his supervisor Prof. Dr. Pınar Karagöz for her guidance, advice, criticism, encouragements, insight and mercy throughout the research.

The project "CA18110 - Underground Built Heritage as catalyser for Community Valorisation (Underground4value)", whose participant academicians' papers are used throughout the research, is appreciated.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xvi

LIST OF FIGURES xix

LIST OF ABBREVIATIONS xxxiii

CHAPTERS

1 INTRODUCTION 1

1.1 Problem Statement 1

1.2 Approach and Contribution 2

1.3 Dataset 3

1.4 The Outline of the Thesis 4

2 LITERATURE REVIEW 5

2.1 Earlier Research 5

2.2 Statistical Approaches 6

2.3 Study of Distributed Representation of Text Using Neural Networks . 8

3 BACKGROUND INFORMATION 11

3.1 Traditional Text Processing Approaches 12
3.1.1 Stop Words 12
3.1.2 Term Frequency-Inverse Document Frequency 12
3.1.3 Stemming, Lemmatization, Tokenization 13
3.2 Artifical Neural Networks 13
3.2.1 Activation Function and Loss Function 15
3.2.2 Well-known ANN Models 17
3.2.2.1 Convolutional Neural Networks 17
3.2.2.2 Recurrent Neural Networks and LSTM 18
3.2.2.3 Sequence-to-Sequence Networks and Attention 20
3.2.2.4 Transformers Networks 22
3.3 Embedding Methods 27
3.3.1 Word Embedding Methods 27
3.3.1.1 Word2Vec 27
3.3.1.2 Fasttext 29
3.3.1.3 GloVe 30
3.3.2 Sentence Embedding Methods 32
3.3.2.1 Skip Thoughts 32
3.3.2.2 BERT 32
3.3.2.3 RoBERTa 35
3.3.2.4 Sentence-BERT 36
3.3.2.5 Language Models are Unsupervised Multitask Learners (GPT2) 37

3.3.3 Document Embedding Methods 37
3.3.3.1 Doc2Vec 37
3.4 Clustering Methods 39
3.4.1 K-Means 39
3.4.2 Agglomerative Clustering 40
3.4.3 DB-Scan 40
3.5 Clustering Metrics 40
3.5.1 Silhouette Score 40
3.5.2 Davies-Bouldin (DB) Index 41
3.5.3 Calinski Harabasz Score 41
3.5.4 Rand Index 41
3.5.5 Homogeneity Score 42
3.6 Hyperopt 42
4 BASIC ARCHITECTURE OF THE EMPLOYED PIPELINE 45
4.1 Dataset Collection 46
4.2 Pre-processing 47
4.3 Embeddings 47
4.3.1 Combination Strategies 47
4.3.1.1 Averaging Word Vectors (avg) 47
4.3.1.2 Term Frequency-Inverse Document Frequency (tf-idf) . 47
4.3.1.3 Sentence Average (sentence-avg) 48
4.3.2 Embedding Methods 48
4.4 Clustering 52

4.4.1 Clustering Methods 53
4.4.1.1 Agglomerative Clustering 53
4.4.1.2 K-Means Clustering 53
4.4.1.3 DBSCAN 54
4.4.2 Clustering Metrics 54
4.5 Hyper-parameter Optimization 55
5 CLUSTERING ANALYSIS TOOL 57
5.1 File Upload and Pre-processing Part 57
5.1.1 File Upload 57
5.1.2 Uploading Documents/Pre-processing Part 58
5.1.3 Uploading Annotations File 59
5.2 Trials Framework Part 59
5.2.1 Usage 60
5.2.2 Data Storage Format 61
5.2.3 Embedding Method Calculations 62
5.3 Result Demonstration and Manipulation Part 64
5.3.1 Clustering Result Table and Line Graph 65
5.3.2 Clustering Result Correlations 65
5.3.3 Clustering Result Scatter Graph 65
5.3.4 Word and Bi-gram Search 66
5.3.5 Word Frequency Graph 66
6 EXPERIMENTS AND RESULTS 69
6.1 Comparisons by Combination Strategy 69

6.1.1 Averaging Combination Strategy (avg, sentence-avg) 70

6.1.2 Term Frequency-Inverse Document Frequency (TF-IDF) Method as Combination Strategy 96

6.1.3 Averaging Combination Strategy Extended 105

6.1.4 Best Results With Other Clustering Metrics 116

6.1.5 Clustering Quality Change With Parameter Optimization . . . 122

6.2 Experimenting With Other Data Sets with Chosen Methods 126

6.2.1 Cappadocia Papers Data Set 126

6.2.2 COVID19 Papers Data set 139

7 CONCLUSIONS AND FUTURE WORK 147

REFERENCES 151

LIST OF TABLES

TABLES

Table 3.1 Glove word co-occurences example [1] 31

Table 6.1 Best Silhouette Score Results Table with Averaging Combination Strategy, Agglomerative Clustering 70

Table 6.2 Best Silhouette Score Results Table with Averaging Combination Strategy, K-Means Clustering 72

Table 6.3 Best Silhouette Score Results Table with Averaging Combination Strategy, DBSCAN Clustering 74

Table 6.4 Best Rand Index Results Table with Averaging Combination Strategy, Agglomerative Clustering 86

Table 6.5 Best Rand Index Results Table with Averaging Combination Strategy, K-Means Clustering 87

Table 6.6 Dimensions of Document Vectors of Embedding Methods with Best Rand Index Results 95

Table 6.7 Best Silhouette Score Results Table with Term Frequency-Inverse Document Frequency Combination Strategy, Agglomerative Clustering . . 97

Table 6.8 Best Silhouette Score Results Table with Term Frequency-Inverse Document Frequency Combination Strategy, K-Means Clustering 98

Table 6.9 Best Rand Index Results Table with Term Frequency - Inverse Document Frequency Combination Strategy, Agglomerative Clustering 102

Table 6.10 Best Rand Index Results Table with Term Frequency - Inverse Document Frequency Combination Strategy, K-Means Clustering 102

Table 6.11 Best Silhouette Score Results Table with Averaging Combination Strategy, Agglomerative Clustering Extended 106

Table 6.12 Best Silhouette Score Results Table with Averaging Combination Strategy, K-Means Clustering Extended 107

Table 6.13 Best Rand Index Results Table with Averaging Combination Strategy, Agglomerative Clustering Extended 112

Table 6.14 Best Rand Index Results Table with Averaging Combination Strategy, K-Means Clustering Extended 112

Table 6.15 Best Davies-Bouldin Index Results of Best Embedding Methods, Agglomerative Clustering 116

Table 6.16 Best Davies-Bouldin Index Results of Best Embedding Methods, K-Means Clustering 117

Table 6.17 Best Homogeneity Score Results of Best Embedding Methods, Agglomerative Clustering 119

Table 6.18 Best Homogeneity Score Results of Best Embedding Methods, K-Means Clustering 120

Table 6.19 Improvements with Optimizations 125

Table 6.20 Best Silhouette Scores Results of Best Embedding Methods, Agglomerative Clustering, Cappadocia Dataset 126

Table 6.21 Best Silhouette Scores Results of Best Embedding Methods, K-Means Clustering, Cappadocia Dataset 127

Table 6.22 Best Davies-Bouldin Index Results of Best Embedding Methods, Agglomerative Clustering, Cappadocia Dataset 128

Table 6.23 Best Davies-Bouldin Index Results of Best Embedding Methods, K-Means Clustering, Cappadocia Dataset 129

Table 6.24 Best Rand Index Results of Best Embedding Methods, Agglomerative Clustering, Cappadocia Dataset 130

Table 6.25 Best Rand Index Results of Best Embedding Methods, K-Means Clustering, Cappadocia Dataset 130

Table 6.26 Best Homogeneity Score Results of Best Embedding Methods, Agglomerative Clustering, Cappadocia Dataset 131

Table 6.27 Best Homogeneity Score Results of Best Embedding Methods, K-Means Clustering, Cappadocia Dataset 131

Table 6.28 Improvements with Optimizations with Cappadocia Dataset 139

Table 6.29 Best Silhouette Score Results of Best Embedding Methods, Agglomerative Clustering, Covid19 Dataset 140

Table 6.30 Best Silhouette Score Results of Best Embedding Methods, K-Means Clustering, Covid19 Dataset 140

Table 6.31 Best Davies-Bouldin Index Results of Best Embedding Methods, Agglomerative Clustering, Covid19 Dataset 141

Table 6.32 Best Davies-Bouldin Index Results of Best Embedding Methods, K-Means Clustering, Covid19 Dataset 142

LIST OF FIGURES

FIGURES

Figure 3.1 $a^{<t-1>}$ refers to the previous hidden state. It is multiplied by W_{aa} weights and then fed to sum operation. The other operands of the sum operation are the actual input ($x_{<t>}$) with its weights W_{ax} and bias term (b_a). The sum is applied an activation function and the hidden state to next node is found (a_t). This hidden state is used to calculate the actual output as well. In order to realize this, the hidden state (a_t) is multiplied with another weights group (W_{ya}), bias term is added (b_y) and another activation function (g_2) is applied [2]. 18

Figure 3.2 The c^t can be thought as long-term, a^t is the short term memory, or a^t is output and c^t is hidden state. LSTM process starts with forgetting irrelevant information. Then, the information is updated with new input in order to decide what new information we're going to store. Next, the relevance gate calculates the long-term memory with results of forget and update gates. Lastly, the output gate decides the output [2]. 20

Figure 3.3 Sequence to sequence method with attention 21

Figure 3.4 Attention Decoding Mechanism 22

Figure 3.5 Transformers Stack 23

Figure 3.6 Transformers Self Attention Visualisation 24

Figure 3.7 Encoder architecture 25

Figure 3.8 Transformer Residual Layer Normalisation 26

Figure 3.9 Overall Architecture of Attention Network 27

Figure 3.10 Word2Vec Proposed Methods [3] 29

Figure 3.11 Skip-thoughts example [4] 32

Figure 3.12 OpenAI's Decoders Model [5] 34

Figure 3.13 BERT Input Structure [6] 35

Figure 3.14 PV-DBOW model [7] 38

Figure 3.15 PV-DM model [7] 39

Figure 4.1 Project Structure 45

Figure 5.1 Triggering One Trial/Pipeline via Automatic or Manual Hyperparameter Selection 59

Figure 5.2 Score Over Cluster Counts Table + Line Graph 65

Figure 5.3 Top Ten Related Papers to the Selected Paper 67

Figure 5.4 Word Frequency Graph 67

Figure 6.1 Best Silhouette Score Results Graph with Averaging Combination Strategy and Agglomerative Clustering Method 71

Figure 6.2 Best Silhouette Score Results Graph with Averaging Combination Strategy and K-Means Clustering Method 73

Figure 6.3 PCA Scatter Plot of 3 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method . 74

Figure 6.4 PCA Scatter Plot of 7 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method . 75

Figure 6.5 TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 40 75

Figure 6.6 PCA Scatter Plot of 3 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method 76

Figure 6.7 PCA Scatter Plot of 7 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method 76

Figure 6.8 TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 30 76

Figure 6.9 TSNE Scatter Plot of 1 cluster With Fasttext Averaging Having Best Silhouette Score Strategy and DBSCAN Clustering Method, perplexity: 30 77

Figure 6.10 TSNE Scatter Plot of 5 clusters With GloVe Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 50 77

Figure 6.11 TSNE Scatter Plot of 5 clusters With GloVe Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 50 78

Figure 6.12 TSNE Scatter Plot of 2 clusters With GloVe Averaging Having Best Silhouette Score Strategy and DBSCAN Clustering Method, perplexity: 50 78

Figure 6.13 TSNE Scatter Plot of 5 clusters With GloVe Pretrained Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 50 79

Figure 6.14 TSNE Scatter Plot of 5 clusters With GloVe Pretrained Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 20 79

Figure 6.15 TSNE Scatter Plot of 2 clusters With GloVe Pretrained Averaging Having Best Silhouette Score Strategy and DBSCAN Clustering Method, perplexity: 50 79

Figure 6.16 TSNE Scatter Plot of 8 clusters With Word2Vec Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 50 80

Figure 6.17 TSNE Scatter Plot of 10 clusters With Word2Vec Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 50 80

Figure 6.18 TSNE Scatter Plot of 1 cluster With Word2Vec Averaging Having Best Silhouette Score Strategy and DBSCAN Clustering Method, perplexity: 50 81

Figure 6.19 TSNE Scatter Plot of 10 clusters With SciBERT Sentence Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 50 81

Figure 6.20 TSNE Scatter Plot of 7 clusters With SciBERT Sentence Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 50 82

Figure 6.21 TSNE Scatter Plot of 2 clusters With SciBERT Sentence Averaging Having Best Silhouette Score Strategy and DBSCAN Clustering Method, perplexity: 50 82

Figure 6.22 TSNE Scatter Plot of 9 clusters With Sentence BERT Sentence Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 50 83

Figure 6.23 TSNE Scatter Plot of 9 cluster With Sentence BERT Sentence Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 50 83

Figure 6.24 TSNE Scatter Plot of 1 cluster With Sentence BERT Sentence Averaging Having Best Silhouette Score Strategy and DBSCAN Clustering Method, perplexity: 50 83

Figure 6.25 TSNE Scatter Plot of 10 clusters With Doc2Vec Document Embedding Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 20 84

Figure 6.26 TSNE Scatter Plot of 9 cluster With Doc2Vec Document Embedding Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 40 84

Figure 6.27 TSNE Scatter Plot of 1 cluster With Doc2Vec Document Embedding Having Best Silhouette Score Strategy and DBSCAN Clustering Method, perplexity: 50 85

Figure 6.28 TSNE Scatter Plot of 7 clusters With Word2Vec Averaging Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30 88

Figure 6.29 TSNE Scatter Plot of 7 clusters With Word2Vec Averaging Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30 89

Figure 6.30 TSNE Scatter Plot of 7 clusters With GloVe Averaging Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30 89

Figure 6.31 TSNE Scatter Plot of 7 clusters With GloVe Averaging Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30 90

Figure 6.32 TSNE Scatter Plot of 7 clusters With GloVe Pre-trained Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30 90

Figure 6.33 TSNE Scatter Plot of 7 clusters With GloVe Pre-trained Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30 90

Figure 6.34 TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30 91

Figure 6.35 TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30 91

Figure 6.36 TSNE Scatter Plot of 7 clusters With SciBERT Sentence Averaging Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30 92

Figure 6.37 TSNE Scatter Plot of 7 clusters With SciBERT Sentence Averaging Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30 92

Figure 6.38 TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30 93

Figure 6.39 TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30 93

Figure 6.40 TSNE Scatter Plot of 7 clusters With Doc2Vec Document Embedding Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30 94

Figure 6.41 TSNE Scatter Plot of 7 clusters With Doc2Vec Document Embedding Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30 94

Figure 6.42 Best Silhouette Score Results Graph with Tf-Idf Combination Strategy and Agglomerative Clustering Method 97

Figure 6.43 Best Silhouette Score Results Graph with Tf-Idf Combination Strategy and K-Means Clustering Method 98

Figure 6.44 TSNE Scatter Plot of 7 clusters With Word2Vec Tf-Idf Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 30 99

Figure 6.45 TSNE Scatter Plot of 7 clusters With Word2Vec Tf-Idf Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 30 99

Figure 6.46 TSNE Scatter Plot of 7 clusters With GloVe Tf-Idf Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 30 100

Figure 6.47 TSNE Scatter Plot of 7 clusters With GloVe Tf-Idf Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 30 100

Figure 6.48 TSNE Scatter Plot of 7 clusters With Fasttext Tf-Idf Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 30 101

Figure 6.49 TSNE Scatter Plot of 7 clusters With Fasttext Tf-Idf Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 30 101

Figure 6.50 TSNE Scatter Plot of 7 clusters With Word2Vec Tf-Idf Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30 103

Figure 6.51 TSNE Scatter Plot of 7 clusters With Word2Vec Tf-Idf Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30 103

Figure 6.52 TSNE Scatter Plot of 7 clusters With GloVe Tf-Idf Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30 104

Figure 6.53 TSNE Scatter Plot of 7 clusters With GloVe Tf-Idf Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30 . 104

Figure 6.54 TSNE Scatter Plot of 7 clusters With Fasttext Tf-Idf Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30 104

Figure 6.55 TSNE Scatter Plot of 7 clusters With Fasttext Tf-Idf Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30 . 105

Figure 6.56 Best Silhouette Score Results Graph with Averaging Combination Strategy and Agglomerative Clustering Method Extended 106

Figure 6.57 Best Silhouette Score Results Graph with Averaging Combination Strategy and K-Means Clustering Method Extended 107

Figure 6.58 TSNE Scatter Plot of 7 clusters With BERT Averaging Extended Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 40 108

Figure 6.59 TSNE Scatter Plot of 7 clusters With BERT Averaging Extended Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 40 108

Figure 6.60 TSNE Scatter Plot of 7 clusters With RoBERTa Averaging Extended Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 40 109

Figure 6.61 TSNE Scatter Plot of 7 clusters With RoBERTa Averaging Extended Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 40 109

Figure 6.62 TSNE Scatter Plot of 7 clusters With GPT2 Averaging Extended Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 40 110

Figure 6.63 TSNE Scatter Plot of 7 clusters With GPT2 Averaging Extended Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 40 110

Figure 6.64 TSNE Scatter Plot of 7 clusters With Skip Thoughts Averaging Extended Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 40 111

Figure 6.65 TSNE Scatter Plot of 7 clusters With Skip Thoughts Averaging Extended Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 40 111

Figure 6.66 TSNE Scatter Plot of 7 clusters With BERT Averaging Extended Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 40 113

Figure 6.67 TSNE Scatter Plot of 7 clusters With BERT Averaging Extended Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 40 113

Figure 6.68 TSNE Scatter Plot of 7 clusters With RoBERTa Averaging Extended Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 40 114

Figure 6.69 TSNE Scatter Plot of 7 clusters With RoBERTa Averaging Extended Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 40 114

Figure 6.70 TSNE Scatter Plot of 7 clusters With GPT2 Averaging Extended Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 40 114

Figure 6.71 TSNE Scatter Plot of 7 clusters With GPT2 Averaging Extended Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 40 115

Figure 6.72 TSNE Scatter Plot of 7 clusters With Skip Thoughts Averaging Extended Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 40 115

Figure 6.73 TSNE Scatter Plot of 7 clusters With Skip Thoughts Averaging Extended Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 40 115

Figure 6.74 Best Davies-Bouldin Index Line Graph of Best Embedding Methods, Agglomerative Clustering 116

Figure 6.75 Best Davies-Bouldin Index Line Graph of Best Embedding Methods, K-Means Clustering 117

Figure 6.76 TSNE Scatter Plot of 6 clusters With Fasttext Averaging Having Best Davies-Bouldin Index Strategy and Agglomerative Clustering Method, perplexity: 30 118

Figure 6.77 TSNE Scatter Plot of 9 clusters With Fasttext Averaging Having Best Davies-Bouldin Index Strategy and K-Means Clustering Method, perplexity: 30 118

Figure 6.78 TSNE Scatter Plot of 6 clusters With Sentence BERT Sentence Averaging Having Best Davies-Bouldin Index Strategy and Agglomerative Clustering Method, perplexity: 30 119

Figure 6.79 TSNE Scatter Plot of 6 clusters With Sentence BERT Sentence Averaging Having Best Davies-Bouldin Index Strategy and K-Means Clustering Method, perplexity: 30 119

Figure 6.80 TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Homogeneity Score Strategy and Agglomerative Clustering Method, perplexity: 30 120

Figure 6.81 TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Homogeneity Score Strategy and K-Means Clustering Method, perplexity: 30 121

Figure 6.82 TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Homogeneity Score Strategy and Agglomerative Clustering Method, perplexity: 30 121

Figure 6.83 TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Homogeneity Score Strategy and K-Means Clustering Method, perplexity: 30 122

Figure 6.84 Best Silhouette Score Line Graph of Best Embedding Methods, Agglomerative Clustering, Cappadocia Dataset 127

Figure 6.85 Best Silhouette Score Line Graph of Best Embedding Methods, K-Means Clustering, Cappadocia Dataset 128

Figure 6.86 Best Davies-Bouldin Index Line Graph of Best Embedding Methods, Agglomerative Clustering, Cappadocia Dataset 129

Figure 6.87 Best Davies-Bouldin Index Line Graph of Best Embedding Methods, K-Means Clustering, Cappadocia Dataset 130

Figure 6.88 TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 10 132

Figure 6.89 TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 10 132

Figure 6.90 TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Davies-Bouldin Index Strategy and Agglomerative Clustering Method, perplexity: 10 132

Figure 6.91 TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Davies-Bouldin Index Strategy and K-Means Clustering Method, perplexity: 10 133

Figure 6.92 TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 10 133

Figure 6.93 TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 10 133

Figure 6.94 TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Homogeneity Score Strategy and Agglomerative Clustering Method, perplexity: 10 134

Figure 6.95 TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Homogeneity Score Strategy and K-Means Clustering Method, perplexity: 10 134

Figure 6.96 TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 10 135

Figure 6.97 TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 10 135

Figure 6.98 TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Davies-Bouldin Index Strategy and Agglomerative Clustering Method, perplexity: 10 136

Figure 6.99 TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Davies-Bouldin Index Strategy and K-Means Clustering Method, perplexity: 10 136

Figure 6.100 TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 10 137

Figure 6.101 TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 10 137

Figure 6.102 TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Homogeneity Score Strategy and Agglomerative Clustering Method, perplexity: 10 138

Figure 6.103 TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Homogeneity Score Strategy and K-Means Clustering Method, perplexity: 10 138

Figure 6.104 Best Silhouette Score Results of Best Embedding Methods, Agglomerative Clustering, Covid19 Dataset 140

Figure 6.105 Best Silhouette Score Results of Best Embedding Methods, K-Means Clustering, Covid19 Dataset 141

Figure 6.106 Best Silhouette Score Results of Best Embedding Methods, Agglomerative Clustering, Covid19 Dataset 141

Figure 6.107 Best Silhouette Score Results of Best Embedding Methods, K-Means Clustering, Covid19 Dataset 142

Figure 6.108 TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 50 143

Figure 6.109 TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 50 143

Figure 6.110 TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Davies-Bouldin Index Strategy and Agglomerative Clustering Method, perplexity: 50 144

Figure 6.111 TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Davies-Bouldin Index Strategy and K-Means Clustering Method, perplexity: 50 144

Figure 6.112 TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 50 145

Figure 6.113 TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 50 145

Figure 6.114 TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Davies-Bouldin Index Score Strategy and Agglomerative Clustering Method, perplexity: 50 146

Figure 6.115 TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Davies-Bouldin Index Score Strategy and K-Means Clustering Method, perplexity: 50 146

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
NLP	Natural Language Processing
TF-IDF	Term Frequency-Inverse Document Frequency
IR	Information Retrieval
BoW	Bag of Words
POS	Part of Speech
MLM	Masked Language Model
NSP	Next Sentence Prediction
HMRFs	Hidden Markov Random Fields
EM	Expectation Maximization
TPE	Tree-of-Parzen-Estimators
reLu	Rectified Linear Activation
CNN	Convolutional Neural Network
RNN	Recursive Neural Network
LSTM	Long-Short Term Memory

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

In the information age, data is growing every year in a very high speed. The growth occurs in different types of media such as news, scientific papers, social media and blogs. Especially papers carry information all around the world, so the scientific research continues cumulatively and rapidly. This can also cause problems such as finding papers related to the research area for the researcher and extracting the unrelated papers without looking into it in detail. Due to the unstructured nature of the text data, machine learning techniques have a difficulty differentiating the documents in a fully automatic way. However, the research is going on and there are several tools to approximate the solution.

Related and unrelated papers can be distinguished according to the similarity of the documents. For that, we need to have a document similarity technique. For a quantifiable similarity measure, documents need to be represented mathematically. In the literature, this is achieved by statistical feature representations such as Bag-of-Words and modern text embedding techniques. Latter one comes from neural network outputs and documents are represented as mathematical vectors with them. So, their similarity can be compared with cosine and euclidean similarity.

There is a variety of text embedding techniques for unstructured text data and they all have hyper-parameters. Also, there are different clustering methods and their parameters. This causes a very high-dimensional search space. In order to achieve good results with as less as possible experiments, an optimization algorithm and an objective function are usable. In this way, the clustering methods and text embedding tech-

niques can be compared together according to their clustering quality performances that can be measured by different clustering quality metrics.

In order to imitate the problem, a collection of papers of academicians from Cost Action CA18110 project is used as the main data set for this thesis. These papers do not have labels and their topics are required to be detected in order for each researcher to benefit from the works of others. However, there are several researchers from different fields of interest and their work needs to be scanned through in order to have a good understanding about the works of others. The information about the relatedness of the papers can give a good understanding for all the participants and outside readers about the conference and give a chance to discover the related papers. So, a framework is required to cluster the papers and shows similar papers in an accurate way. Also, the outputs of the framework needs to be tested with other and similar document collections.

1.2 Approach and Contribution

The problem needs a set of processes that needs to be processed in a sequential way. This sequential process, i.e. pipeline, contains three main parts.

First, the representation of the documents are calculated as n-dimensional vectors that are easy to compare in terms of similarity. Before calculating these vector-type embeddings, we need to get embeddings of smaller entities in a document. So, word embedding, sentence embedding and direct doc2vec embedding results as a vector are obtained. Then, we calculate averages, or weighted averages of these embeddings to reach document embedding. Document embedding values become an n-dimensional vector, where n is variable, as a result.

Secondly, these document embeddings are clustered using different methods (K-Means, Hierarthical, etc.) and the quality of these results are measured by using different clustering metrics. Also, the documents are put into fixed number of clusters by author by benefiting from the clustering results so far. In this way, more clustering metrics, which use the true labels in their calculation, could be used to minimize the measuring errors. In addition, the best result from each technique according to given

clustering metric is chosen in order to compare the quality of results of embedding techniques in terms of clustering quality.

Lastly, to improve the quality of the outcomes obtained from embedding techniques, the hyper-parameters of them are changed and experiment is repeated several times. Also, in order to improve experiment results, these repetitions are done with the aid of a hyper-parameter optimization method called Hyperopt rather than random decisions. For this optimization method, the clustering metrics are output while, embedding hyper-parameters are the input. In addition, this process is repeated for the parameters of the clustering methods in order to decrease the search space and improve the clustering quality.

Thus, we started from embeddings and finished with embeddings in this section. This means, the pipeline feeds its clustering results using metrics back to the embedding techniques. This pipelined architecture has become also the backbone of the tool that we use to do all the experiments. Mostly using this tool and doing other experiments, the results are retrieved and shown in the following sections.

The framework of this thesis has measured the degree of improvement when the parameters of its components (text embedding and clustering) are optimized. This has been done using a complete tool from uploading files to getting the results. In addition, different text embedding and clustering results are compared and analysed in terms of different clustering measures on a multi-disciplinary paper collection after optimization of the parameters. These are the main contributions of this study.

1.3 Dataset

The main data set is a multi-disciplinary paper collection which has 599 papers in the document. The other one is a smaller dataset that has the papers that are about Cappadocia after 2015. These datasets are collected by the author. The last one is the Covid 19 paper collection which has over 30.000 papers in it [1]. The first two contain full paper if exists, the abstract of paper otherwise, and the last one has only abstract part of the papers.

[1] Covid19 Papers Collection

The multi-disciplinary data set is the main data collection for experiments. The other ones are control sets used to control the best results according to the performances on the first collection.

1.4 The Outline of the Thesis

In Chapter 2, literature review about document clustering. In Chapter 3, the background information on the methods we use during thesis. In Chapter 4 and 5, the tool used during thesis for the experiment and demonstration of the results is explained in deep. In Chapter 6, the experiment results are evaluated. Chapter 7 is devoted to conclusions.

CHAPTER 2

LITERATURE REVIEW

Document clustering is an early topic of machine learning. So, there is a whole literature about document similarity and clustering. Research has gone on the way of neural network applications after vector representations of chunks of texts such as words and sentences. This has become the main concern of this thesis as well. Before, some statistical approaches such as Markov models, and some basic IR methods such as BoW, POS methods are utilized for document clustering. Researchers, in general, tried to increase the quality of clustering by changing parameters, applying novel strategies, changing optimization method or objective function, using human help, and more. The ones related to this thesis are explained in this chapter briefly.

2.1 Earlier Research

Earlier research contains attempts of development of efficient algorithms, visualisation of documents in different clusters, browsing large document collections with the help of document clustering, generation of hierarchical clusters of documents, finding the natural clusters in existing document classes and then utilizing these clusters to produce a successful classifier for new documents [8].

As an earlier research [9], the hierarchic clustering method is analysed in terms of different parameters and complete linkage is suggested, which is related to our thesis where the clustering method parameters are optimized and one of those methods is Agglomerative Clustering, an approach for hierarchic clustering. The paper is under the area of IR, searching is measured with defined queries and corresponding documents. The validity of clustering results is measured with three principles:

constructed clusters being away from randomly created clusters, the percentage the clusters recovering the original data set structure in terms of original similarity matrix and the validity of individual clusters. This work is similar to this thesis in terms of trying different parameters of a clustering method and using different clustering quality measures.

2.2 Statistical Approaches

In statistical methods for clustering, human supervision is applied so that clustering quality can be increased with less effort than labeling all the documents. These methods are called semi-supervised clustering approaches. They can be categorized as two general categories: constraint-based and distance-based. Constraint-based methods may enforce some restrictions or make the clustering results to satisfy labeled samples. Distance-based methods utilize the distortion measure by optimizing it with the supervised data. These two approaches are combined in a framework that implements Hidden Markov Random Fields, where the objective function, or measure, is a kind of K-Means algorithm, HMRF-KMEANS, as an EM method [10]. The other attempt for increasing clustering quality with a Markov model proposes a combinatorial Markov Random Field (Comraf), an undirected graph model [11]. This attempt (Comraf) is utilized in another work that is based on an idea of clustering documents according to criteria specified by users in an iterative manner [12]. In this work, the number of clusters and then the feature types are chosen among the possible features such as BoW, POS tags and n-grams according to clustering task at hand. Also, the user adds some new features such as gender of author, genre and authors' sentiments if possible to the existing features. On top of that, a combinatorial Markov Random Field is used to create a good clustering. Also, giving labels to the words is another idea. The users are asked to define classes with a few representative words and some trusted documents are labeled according to this information from user, then the others are clustered by using these labeled documents [13]. As another approach, a top-down and a bottom-up clustering of random variables of document collection, with a local optimization correction routine is utilized [14]. These methods are related to this thesis in terms of having data representations. In these methods, documents are

represented via different random variables and they are clustered according to these variables. These feature variables are created automatically in neural network based models that we used and instead of variables, the hyper-parameters are tried to be optimized. Also, the attempts generally try to optimize the objective function, distortion measure specifically which is similar to one of the aims of this paper: optimizing average silhouette score. Moreover, labeling by human is utilized in order to measure well and then improve clustering quality. In this thesis, human labeling is utilized for only measurement reasons.

Semantic relations inside the documents can be used to measure document similarity and realize document clustering. The former methods were using Bag of Words like features but they do not capture the semantic meaning. With the increase of the data size, some methods such as lexical chains are utilized to get semantic information from text documents. Wei et al. [15] proposes a clustering solution based on these shortcomings and more. The similarity measure is changed as if words that are synonymous have the same importance in the context based on WordNet [1]. Also, the main theme of clusters are predicted according to lexical chain in order to reduce the dimensions of feature set, potentially leading to better clustering results. This is a different way of capturing information about context than using distributed representations of words and phrases that is used in this thesis. Also, the topic extraction of clusters is not in the scope of this thesis but can be worked on for further studies over the thesis.

Term frequency is utilized as a similarity measure in some clustering tasks. Actually tri-gram frequency is used to measure the similarity between two text documents. The co-occurence of words inside tri-grams as starting word and ending word, and uni-gram frequencies of those words are the main elements of measurement index. The tri-gram co-occurences increase the similarity while uni-gram frequencies decrease. Based on these word pair similarity measure, a framework for the overall text similarity is constructed [16]. This similarity measure is extended further since it considers only whether or not terms co-occur in the documents being compared, in a binary fashion. When document with few occurrences of a term is compared with another document with many occurrences may not help much. So, a new measure is created

[1] WordNet

that is also account for the term frequencies [17]. This work also used the similarity measure for clustering the documents. In this thesis, the term frequency and inverse document frequency is utilized per term for weighting the word vectors instead of being a unit for similarity measurement. The reason is that the content of this thesis is about distributed representations of text chunks, so there is not a measurement that does not use these representations.

2.3 Study of Distributed Representation of Text Using Neural Networks

There are clustering research about using the distributed representations of words and other text chunks. There is a work that is very similar to the thesis but on Polish language [18]. The BERT and Word2Vec methods are used in the context of this thesis as well. The improved version of ELMo, GPT2 is used in this thesis. Also, some clustering methods such as and the similarity measures such as cosine similarity is similar in both studies. But the scope of this thesis contains more methods and optimization of parameters of both embedding methods and clustering methods additionally. In another one, news articles are clustered according to Word2Vec, tf-idf and Doc2Vec methods. Then the individual clusters are summarized. The clustering is done in this thesis too but the summarization of clusters is not [19]. This can be thought as a further study. Also, there is a research about a clustering method different from a clustering method which is not used in this thesis: WE Clustering [20]. It is claimed to be the solution of the problem of high dimensionality in an effective manner which is encountered in this thesis, as well.

Increasing clustering quality is the main issue for text clustering problems. For this, several algorithms and structures are proposed. Some of them try to increase the validity by improving K-Means clustering algorithm [21] [22]. One of them [21] adds the Canopy algorithm between creating text representations and K-Means. In this way, pre-clustering is applied and improvements are achieved for big data. However, data is not that big in the context of this thesis. The other method [22] tries to improve K-Means by transforming optimization problem of K-Means to neural network, in fact autoencoders and the hyper-parameters of the network is optimized. This detailed way is not the way thesis followed. Instead of that, classical K-Means parameters

are tried to be optimized. Another approach applies nature inspired optimization algorithms such as Genetic Algorithm, Particle Swarm Optimization Algorithm, Ant Colony Optimization, Krill Herd Algorithm into text clustering problem [23]. Lastly, there is a framework very similar to our approach in this thesis, AutoClust. The basis of this approach is clustering algorithm selection and hyper-parameter tuning. The clustering algorithm and its hyper-parameters are chosen according to a ground truth whose aim is to explore the relationship between internal cluster validity indices and the Adjusted Rand Index and learn a mapping between them [24]. As a searching algorithm, Tree Structured Parzen Estimator is applied. This is similar to this thesis in terms of optimizing the clustering method parameters by Tree Structured Parzen Estimator algorithm. However, silhouette score and the visualisation methods are used to determine and analyse the results in this thesis even if it is criticized and a new method is proposed in the AutoClust paper. Meta learning is also used for optimizing the clustering quality. With some documents labeled, clustering quality can be improved with the meta-learning, i.e. learning by learning approach [25]. Neither objective functions nor clustering indices use this approach but the adjusted rand index used in this thesis is based on a few labeled documents, so a metric with meta learning can be utilized in a further study as an objective function.

Mentioned methods are related to this thesis in terms of different perspectives as explained at above as well:

- Earlier research contains data visualisation, clustering methods and clustering validity for these methods. The clustering validity is tried to improve. In this thesis, data visualisation is utilized to see the denseness and distinguishability of clusters, clustering methods are also used and improving the clustering quality is one of the main concerns.
- The methods utilized in statistical approaches are used in this thesis as a helper in text pre-processing and TF-IDF is used as combination strategy instead of a similarity measure. Structured methods like lexical chains are also mentioned because their aim is to capture context of the thesis, in which we used the neural networks for capturing context through distributed representations.
- There are different methods over distributed representations for understanding

the unlabeled or semi-labeled text documents related to this thesis. They are based on the distributed text representation methods like Word2Vec, BERT, etc. and also, the parameters of clustering methods, and clustering validity measures are tried to be chosen effectively and optimized in some works as explained. In this thesis, these text representations are also utilized and the choice and optimization of parameters of different embedding and clustering methods is one of the main objectives.

CHAPTER 3

BACKGROUND INFORMATION

Text processing is an old area in the history of machine learning and on the rise nowadays. So, some text processing techniques have been developed. These are usually divided into two categories: traditional approaches and embedding approaches. Traditional approaches need more feature engineering and usage of statistical techniques on the extracted data. Until the advance of word embedding techniques, these techniques were utilized to model and analyse text data. Word embedding techniques is an artificial neural network architecture that is wisely designed to treat unlabeled words as if they are labeled with its neighbor word(s), and apply neural network to get the weights, so that these weights can represent the word. This distributed representation of words changed the direction of the text processing research. So, other methods and improvements are applied for improving the quality of these distributed representations. Our thesis has mostly utilized from the embedding techniques, while the traditional approaches are sometimes used.

Clustering is a technique to group similar entities together. Main clustering techniques are used extensively and since the text can be represented with n-dimensional vectors thanks to the advances in embedding techniques, the text entities can also be clustered. Clustering quality can also be measured using clustering metrics, each having a different focus on the clustering quality.

Hyper-parameters are the parameters that cannot be changed during training the neural network. Some optimization tools like "Hyperopt" can optimize them according to the error function defined. This makes randomization lesser to choose those parameters by applying statistical analysis as if hyper-parameter selection is the input and it is decreasing the error function defined.

These methods are used in thesis to reach the conclusion. They will be explained in more detail in this chapter.

3.1 Traditional Text Processing Approaches

Traditional NLP approaches contain the approaches that are used before modern embedding techniques. Their applications in embeddings are complementary or assistive. The ones used in the thesis are explained.

3.1.1 Stop Words

In languages, there are some words that contain no information about the context of the text, so they are eliminated before traditional analyses. According to one implication of Zip's Law [26], the presence of words are inversely proportional to its frequency rank. The words that are mostly at the top of the frequency ranking are the connectives, prepositions, punctuation, numbers, etc. in almost all of the documents. So, these words are generally irrelevant for further analyses. There are different lists for stop words but also there are some words that are almost in all stop-words lists, such as the, and, so.

3.1.2 Term Frequency-Inverse Document Frequency

Stop-words method is not the only method for eliminating the less related words. Weighting the words according to their relevance to the documents is another option. For this, TF-IDF method is broadly utilized as a helper. Term generally refers to the words in documents. These words are noted for their frequencies in each document and for the count of documents the word occured among all documents. Their frequencies in each document is called term frequency and the latter is called document frequency. It is expected the term frequency of a word to be higher and the document frequency to be lower. So, the document frequency is inverted and it is multiplied with term frequency, logically. While calculating, the inverse document frequency is generally normalized by taking the log of it, and not to get negative results, the value

is multiplied with a constant number that never makes the idf negative, total document count in collection (N). Also, not to get 0 value inside logarithm value, 1 is added to the value inside logarithm, and not to get 0 value from division inside logarithm, 1 is added to both nominator and denominator. The equation is formulated as in Equation 3.1.

$$tf - idf = \frac{t_{d,f}}{log(1 + \frac{1+N}{1+df_t})} \tag{3.1}$$

Thus, the formula in Equation 3.1 can be utilized to give less weight words, which do not give much information about the context and used in every text document, without defining a global stop-words list.

3.1.3 Stemming, Lemmatization, Tokenization

Stemming, lemmatization and tokenization are other pre-processing steps before analysis. Stemming and lemmatization transform the words into the root state of the word. Stemming follows an algorithm to realize it but the root word may not always be the language word and somehow meaningful. On the other hand, lemmatization gives exactly the language word as output. For example, 'believes' word is stemmed to 'belive' and lemmatized to 'believe'. We should keep in mind that after lemmatization and stemming, we lose the tense, punctuation like information, that cannot be processed effectively with traditional NLP techniques. Tokenization is the process of splitting data into chunks, and text into words in this context. Generally, tokenization is applied after stemming and/or lemmatization steps.

3.2 Artifical Neural Networks

Artificial Neural Networks is based on human brain to solve the machine learning problems, such as classification and regression. The method basically takes inputs and returns discrete or probabilistic outputs, which are calculated via the aggregation of the multiplication of input values with some weights (Equation 3.2).

$$y_j = \sum x_i * w_i \tag{3.2}$$

In nature, this method needs predefined input and output values, so that the weights can be calculated. If we had one input and its output value, this (Equation 3.2) would be an easy equation but there are several inputs, their outputs, and these weights must apply to all of these equations with small error. So, y_{pred} can be defined as the approximate output value we get and the y_{true} is the correct label of the input. Basically, the main objective of the method is to diminish the error, such as $|y_{pred} - y_{true}|$.

In ANN, weights are calculated with an idea of back-propagation and gradient descent. The ANN is fed by inputs as one batch at a time, but taking one input at a time is easier to think of. Assuming one input is given to network and weights are given artificial values, then the predicted output is calculated. This calculation is compared with the true output and the error is calculated, then the derivative of the error with respect to the weights would give us the increasing direction of the error. At the opposite direction of this derivative value, the weights are updated, negative of the derivative is subtracted from previous weight. Say, our error function, i.e. loss function, is $\frac{1}{2}(y_{pred} - y_{true})^2$. In this situation, the gradient descent method would update the weights as in the formula in Equation 3.3:

$$w_i^{n+1} = w_i^n - \mu(y_{true} - y_{pred})x_i \tag{3.3}$$

μ is the learning rate. Its value is between 0 and 1. If the desired behaviour is going global minimum as early as possible, close to 1 values must be chosen. Otherwise, close to 0 values will be appropriate. Going to minimum fast,or choosing learning rate bigger, causes problem of passing global minimum and the small values of μ will cause a stuck into local minima. So, choosing a suitable learning rate is a problem.

This is an iterative process. For all inputs, this will be repeated. For large datasets, inputs are given as batches and weights are updated with one pass.

In deep learning, there are two main extensions. First, if we call one weight group as layer. There are several layers that are on top of each other in a way that the output of one layer is the input of the upper layer. The second extension is the activation functions. The predicted output seems linear transformation of inputs and weights, but this would not grasp the non-linear relations between input and output, such as $y = x^2$. So, there are functions that do non-linear transformation to the output of

previous network layer.

It should be also noted that, back-propagation and gradient descent are applied to layers by using chain rule. Without much further detail, it works like this. The input of one hidden layer is the output of previous layer, as mentioned. So, we can chain the calculation of each derivation to inputs (Equation 3.4).

$$\frac{\partial f}{\partial x_i} = \frac{\partial f}{\partial \theta 1}\frac{\partial \theta 1}{\partial \theta 2}\frac{\partial \theta 2}{\partial \theta x_i} \tag{3.4}$$

Until that layer, we calculate derivatives with respect to the input of just that layer and multiply them. Then, we take the derivative with respect to the weights of that layer (Equation 3.5).

$$\frac{\partial f}{\partial w_j} = \frac{\partial f}{\partial \theta 1}\frac{\partial \theta 1}{\partial \theta 2}\frac{\partial \theta 2}{\partial \theta w_j}x_i \tag{3.5}$$

This makes the calculation more efficient and easily tractable. Also, we can see that the layers are getting information only from the previous layer and passes to next layer. This means that lower layers have basic information about the data and the upper layers have more abstract information about the data.

Note that, activation function layers are also put into derivative calculation but since they have no weights, there will be no updates in those layers.

3.2.1 Activation Function and Loss Function

As mentioned earlier, activation function breaks the linear behavior of the deep learning architecture. It is chosen to be simple functions that their derivatives of them can also be calculated easily. These are mostly used activation functions [27]. Also, some newer activation functions utilized in Word2Vec algorithm [28] is explained as well.

- *Rectified Linear Activation*: This is one of the mostly used activation functions and it offers the better performance and generalization in deep learning compared to the sigmoid and tanh activation functions.
  max(0.0, x)

- *Sigmoid Activation*: Sigmoid function is used for predicting outputs based on probability by appearing in the output layers of deep learning architectures.

However, it can cause some problems such as vanishing gradients problem and slow convergence.
1.0 / (1.0 + e^{-x})

- *Hyperbolic Tangent Activation*: The function became the preferred function compared to the sigmoid function in that it gives better training performance but it does not solve the vanishing gradients problem either for multi-layer neural networks.
$(e^{x} - e^{-x})/(e^{x} + e^{-x})$

- *Softmax Activation*: If there are multi-class models, this is the function that is mostly used. This gives probabilities to each class summing to 1 and the class with highest probability is chosen according to it. Between sigmoid and softmax, the former is used in binary classification and the latter is used for multi-class classfication tasks.
$(e^{x_i} / \sum e^{(}x_j))$

- *Hierarchical Softmax Activation*: It is an efficient approximation to softmax activation. It has a binary tree whose nodes are relative similarities of its child nodes and the leaves are the outputs. In this way, the actual representation can be calculated with a random walk.

- *Negative Sampling*: It is a method that uses some random noise distribution as negative samples and they need to be distinguished from positive samples, which are actual outputs. Aim is that the predicted value moves away from negative samples, approaches to correct outputs.

Loss function is the same as error function. It calculates the difference between predicted value and actual value. These are the mostly used loss functions:

- *Regression Losses*:
  * *Mean Square Error* : $\sum_{i=1}^{n} \frac{(y_{true} - y_{pred})^2}{n}$
  * *Mean Absolute Error* : $\sum_{i=1}^{n} \frac{|y_{true} - y_{pred}|}{n}$

- *Classification Losses* :

* *SVM Loss* : $\sum_{j \neq 1} max(0, s_j - s_{y_i} + 1)$

* *Cross Entropy Loss* : $y_{true_i} log(y_{pred_i}) + (1 - y_{true_i}) log(1 - y_{pred_i})$

3.2.2 Well-known ANN Models

There are models built on top of each other to grasp features of the data extensively. The models mentioned contain two models (RNN and CNN) that affect whole machine learning research and two methods that revolutionize the NLP research.

3.2.2.1 Convolutional Neural Networks

As a specific version of deep learning, CNNs are fine-tuned combinations of convolutional and pooling layers [2].

Convolutional layer filters the output by passing filter horizontally and vertically. While filter is moving, "stride" is the parameter, which decides how much cell the filter passes for each move. Channel is the parameter for 3rd dimension of the input for any layer. The filter must apply to all channels of input, then, the count of filter used at that layer decides the channel number of the output. As an example, if input dimensions are IxIxC, and there are K filters, then the filters are applied all C channels of input and the output will be in IxIxK dimensions. With stride, we can shrink or extend the same dimensions and with channel parameter we can shrink or extend the third dimension. Careful readers will realize that the the edges of input will not interact with all cells of filter. To overcome this problem, zero padding is used. Given amount of zeros are put into the edges of input. The three hyper-parameters (filter size (F), stride (S) and zero padding (P_{start} and P_{end})) must output to an integer when they are applied to the formula in Equation 3.6, and that output (O) will be the 2D dimensions of the output, where I is the dimensions of input.

$$O = \frac{I - F + P_{start} + P_{end}}{S} + 1 \tag{3.6}$$

Note that, P_{start} is the zero padding in the left-upper side and P_{end} is the zero padding in the right-bottom side, but they become equal most of the time.

Pooling layer is a down-sampling operation to decrease the complexity of the model in order to get rid of bias. Averaging and maximum pooling are the mostly used types that average or take maximum of the cells they are applied. This layer is generally used just after convolutional layer.

3.2.2.2 Recurrent Neural Networks and LSTM

Recurrent Neural Networks allow us to overcome the difficulty of the possibility of processing input of any length. There are two output values for one calculation: one is actual output and the other one is the hidden state that will be passed as the second input to the other node. In equation 3.7 and 3.8 and Figure 3.1 expresses the concept, where $a_{<t>}$ is the hidden state and $y_{<t>}$ is the output at that t.

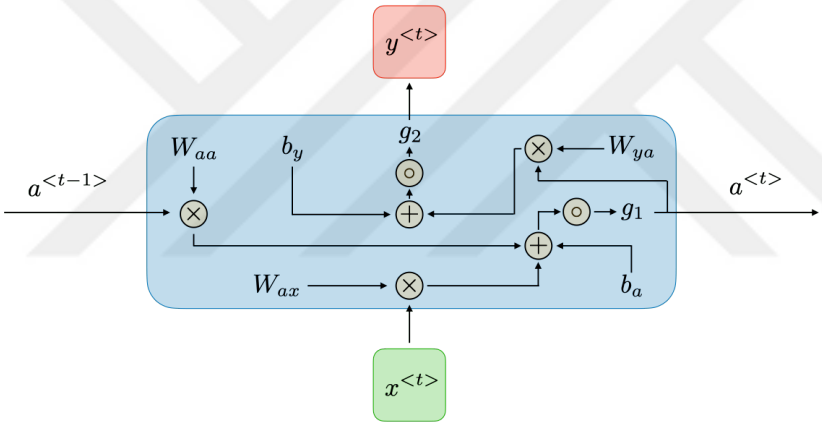

Figure 3.1: $a^{<t-1>}$ refers to the previous hidden state. It is multiplied by W_{aa} weights and then fed to sum operation. The other operands of the sum operation are the actual input ($x_{<t>}$) with its weights W_{ax} and bias term (b_a). The sum is applied an activation function and the hidden state to next node is found (a_t). This hidden state is used to calculate the actual output as well. In order to realize this, the hidden state (a_t) is multiplied with another weights group (W_{ya}), bias term is added (b_y) and another activation function (g_2) is applied [2].

$$a_{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{3.7}$$

$$y_{<t>} = g_2(W_{ya}a^{<t>} + b_y) \tag{3.8}$$

Loss function for RNNs is defined as the some of the summation of losses of all outputs (Equation 3.9).

$$L(y_{true}, y_{pred}) = \sum L(y_{true}^{<t>}, y_{pred}^{<t>}) \tag{3.9}$$

Backpropagation is done in each point in time, for each loss, and this is called backpropagation through time. For very deep networks, the gradients can vanish or explode because the gradient calculations become multiplications with very large number of operands. RNNs, in nature, can cause vanishing gradients problem since the network is extended through T stages [2]. To solve this problem, the gradient clipping is used. That means clipping calculated gradient values after some threshold. Also, the gates are created to overcome this problem. Each gate corresponds to one operation and using these gates, LSTM, a kind of RNN architecture, was created [29]. The gates are:

- *Update Gate* Γ_u*:* How much past should matter now?
- *Relevance gate* Γ_r*:* Drop previous information?
- *Forget Gate* Γ_f*:* Erase a cell or not?
- *Output Gate* Γ_o*:* How much to reveal of a cell?

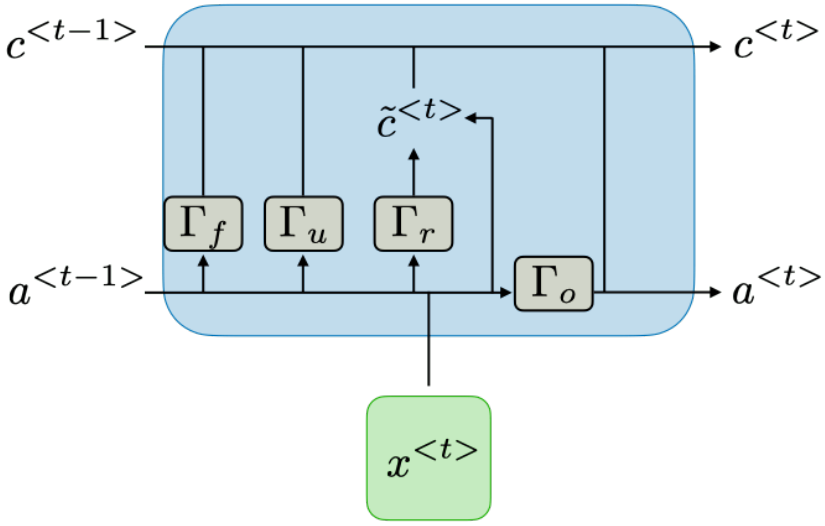

Figure 3.2: The c^t can be thought as long-term, a^t is the short term memory, or a^t is output and c^t is hidden state. LSTM process starts with forgetting irrelevant information. Then, the information is updated with new input in order to decide what new information we're going to store. Next, the relevance gate calculates the long-term memory with results of forget and update gates. Lastly, the output gate decides the output [2].

3.2.2.3 Sequence-to-Sequence Networks and Attention

Before understanding attention, we need to look at sequence-to-sequence networks. They are based on 2 pioneering papers [30] [31]. They are used in machine translation, text summarization and image captioning tasks. Seq-to-seq model takes a sequence of items (e.g. words), and outputs another sequence of items. As examples, for machine translation, input and output are sequence of words from different languages; in text paraphrasing, they are still both a sequence of words but in the same language.

In the details of the architecture, two main components exist: encoders and decoders. Encoders encode the input into context vector, then the decoder creates the output one by one. Encoders and decoders are mostly RNN kind of neural networks; while the context vector is the hidden layer output values of encoder. The sequence of inputs is taken by encoder RNN networks, where arbitrary number of inputs can be processed.

These RNN networks do not give output. The last hidden state is the context word, which is the initial hidden state of the decoder. Decoders do not take extra inputs other than the hidden state, but each RNN inside the decoder creates one output. So, the sequence of inputs in encoder corresponds to outputs in decoder (Figure 3.3).

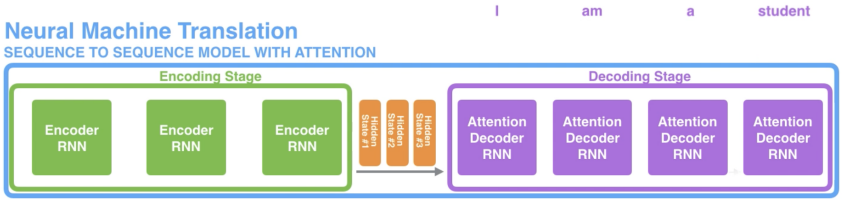

Figure 3.3: Sequence to sequence method with attention [1]

There is a limitation in this model. The context vector cannot capture the context of long sequences. This takes us into adding attention to this architecture. Attention allows the model to focus on the relevant parts of the input sequence as needed. Firstly, the context definition changes. Encoder passes not only the last hidden state but all the hidden states created between RNNs. Each hidden state is related to certain item in input sequence. The hidden states given are scored for each decoder RNN so that, more relevant ones are amplified and the less relevant ones are drowned according to their score.

Scoring mechanism starts working by taking embedding of the <END> token, and an initial decoder hidden state. For tasks that process word sequences, this token is an embedding generated by word embedding algorithms, which are explained in the next section. Afterwards, RNN takes the token and an arbitrary initial state (h_i) and creates hidden state (h_4), while output is discarded. The encoder hidden states (h_1, h_2 and h_3) and the created hidden state (h_4) are employed inside attention network, then the context vector (C_4) is generated. h_4 and C_4 are concatenated and fed into a feed-forward neural network. The output of this network indicates the first item of the output sequence. The hidden state created (h_4) also passes to the next item prediction and the scoring mechanism works in the same way. This iterative process forms the whole decoding mechanism (Figure 3.4).

[1] https://jalammar.github.io/images/seq2seq_7.mp4

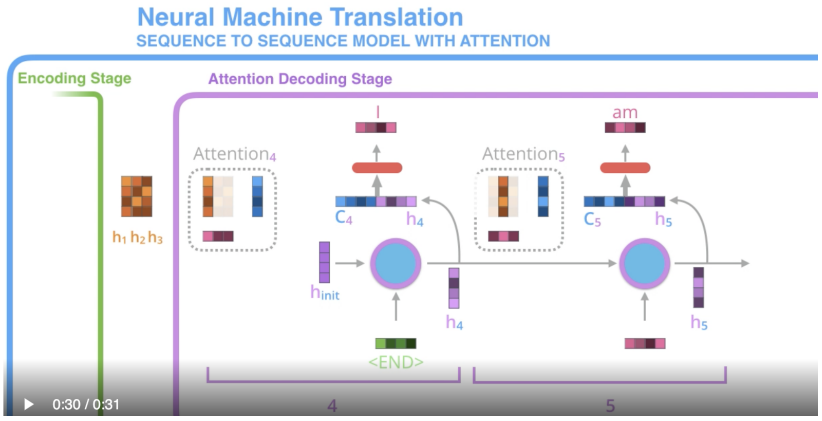

Figure 3.4: Attention Decoding Mechanism [2]

3.2.2.4 Transformers Networks

This architecture is proposed in "Attention: All You Need" paper [32]. It completely changes the encoder-decoder, i.e. seq-to-seq, model in many ways. First one is using multiple encoders and decoders, where only the input of the first encoder is the input, or embedding. The other encoders take the output of the previous encoders as input. The hidden layer created by last encoder is the input to all encoders. The overall architecture is demonstrated in Figure 3.5.

[2] https://jalammar.github.io/images/attention_tensor_dance.mp4

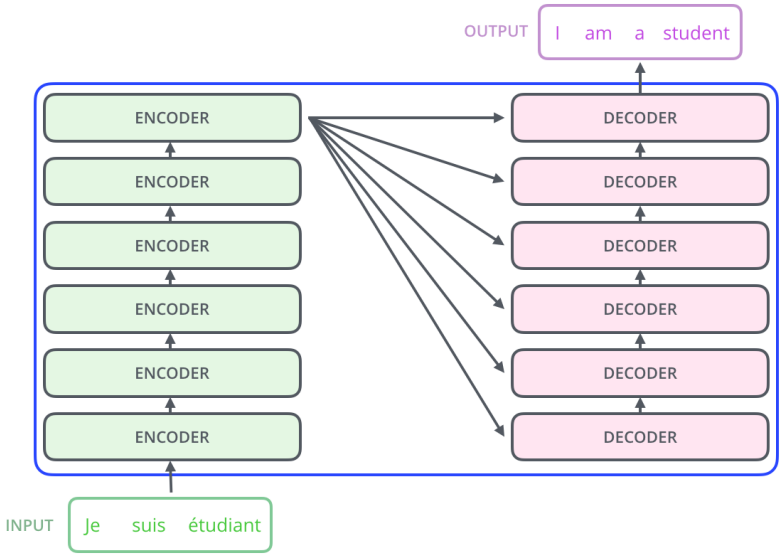

Figure 3.5: Transformers Stack [3]

Another change of this model is self-attention network inside encoders. Self attention network grasps relevance of features among themselves. For example, in the sentence "The animal didn't cross the street because it was too tired", the word "it" is more related with "animal" than any word else in here. It resembles to seq-to-seq RNNs such that the upcoming words have information about the previous words via the hidden state variables. In the Figure 3.6, the relations of each word to "it" after training was shown.

[3] https://jalammar.github.io/images/t/The_transformer_encoder_decoder_stack.png

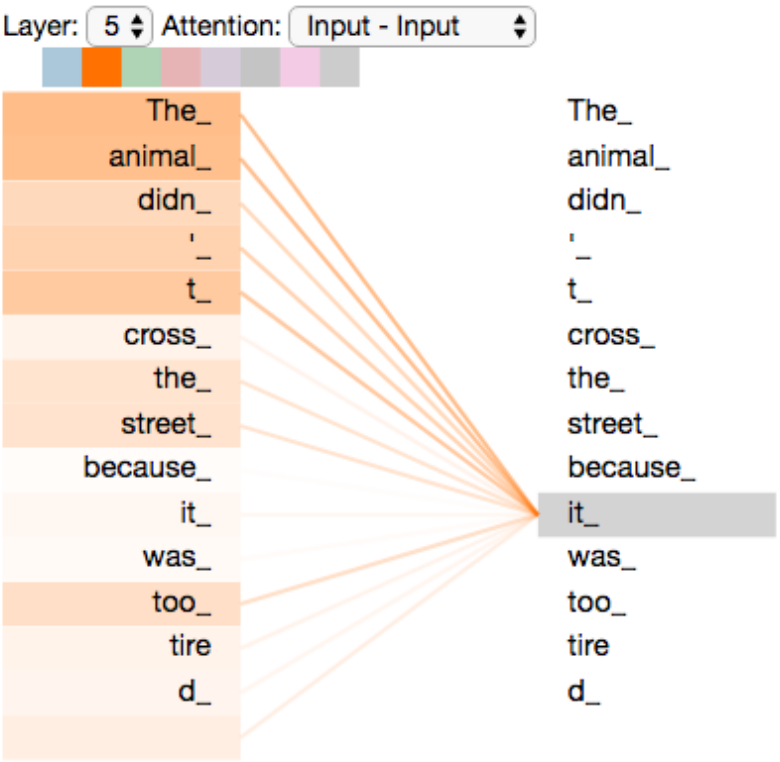

Figure 3.6: Transformers Self Attention Visualisation [4]

A more precise and detailed explanation of self-attention networks requires the terms "key", "value" and "query" to be explained. Key (k_i) and query (q_j) is multiplied for each input (x_i) and softmax function is applied after divided by some constant ($\sqrt{d_k}$, d_k is the size of input vector). Calculated values are summed up, and multiplied by each value (v_j). The output is passed to feed forward network above it. This process is sequential, but the feed forward neural network part can be parallelized. With softmax function, the effect of the less related inputs are lessened, while the more related ones are amplified more (Figure 3.7).

[4] https://jalammar.github.io/images/t/transformer_self-attention_visualization.png

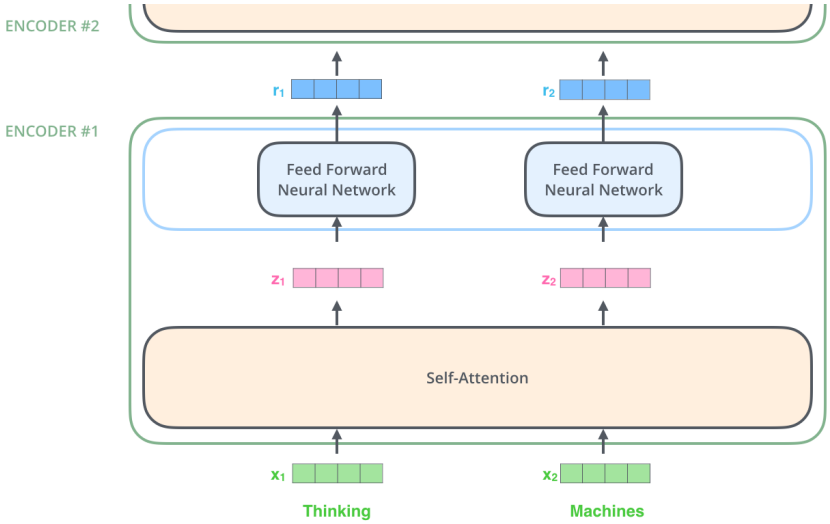

Figure 3.7: Encoder architecture [5]

The "keys", "queries" and "values" have weights W^k, W_q and W_v and their current values are the product of these weights with the current input. If we call the output of the self-attention network as z, the matrix calculations can be formulated as in Equation 3.10.

$$Z = \frac{softmax(Q * K^T)}{\sqrt{d_k}} * V \tag{3.10}$$

where $Q = X * W^Q, K * XxW^K, V * XxW^V$.

"The Attention: All You Need" paper [32] further redefines the self-attention network to improve the performance, but this is out of scope. However, one point remained before moving into the decoder architecture. Add & Normalize step after each sub-layer (self-attention, FFNN) is applied as shown in Figure 3.8.

[5] https://jalammar.github.io/images/t/encoder_with_tensors_2.png

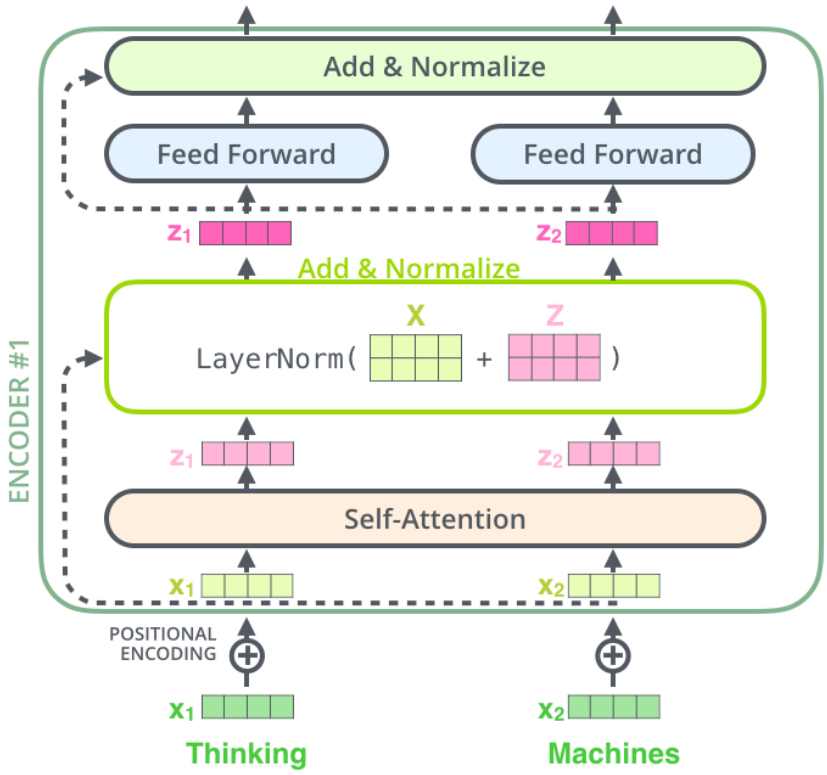

Figure 3.8: Transformer Residual Layer Normalisation [6]

The encoding mechanism outputs a bunch of keys and values. These are fed into the decoder architecture, in which each decoder consists of self-attention network, encoder-decoder attention network and one FNN network. The self-attention layer is only allowed to attend to earlier positions in the output sequence. This is done by masking future positions (setting them to -∞) before the softmax step in the self-attention calculation [32].

The overall architecture is given in Figure 3.9.

[6] https://jalammar.github.io/images/t/transformer_resideual_layer_norm_2.png

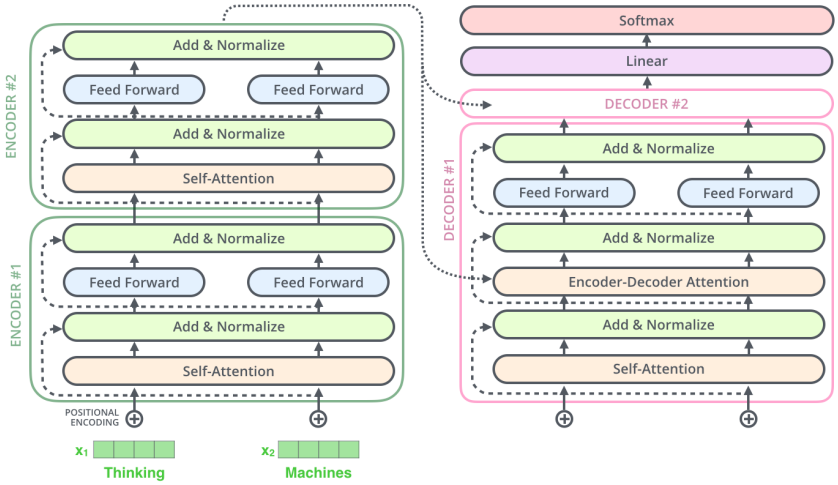

Figure 3.9: Overall Architecture of Attention Network [7]

3.3 Embedding Methods

3.3.1 Word Embedding Methods

3.3.1.1 Word2Vec

Word2Vec is the representation of the words with n-dimensional vectors. It has started with the paper Efficient Estimation of Word Representations in Vector Space [3]. In this paper, Mikolov et all. proposed two new methods (Skip-gram and Continuous Bag-of-Words) according to the following motivations:

- To change how the words are treated in traditional techniques. The words are all atomic units in techniques like n-gram, or bag of words and the similarities among them are not taken into the consideration.
- Another motivation is to deal with the data limitation of the methods. The shortage of domain-related data in speech recognition is given as the example of this limitation in the paper.

[7] https://jalammar.github.io/images/t/transformer_resideual_layer_norm_3.png

- Neural network based language models significantly outperform N-gram models.

For these reasons, they thought that, more advanced model needs to be created to improve the quality of NLP tasks. The attempt to represent words as vectors are tried before this paper [33], which Mikolov worked in as well. In this earlier paper, they made continuous word vectors be learned using simple model and a neural network model is trained on top of these representations. In fact, word2vec approach can be thought as more complete and successful version of the this earlier attempt in a way of continuous word vector representations. This version, known as Word2vec, is used extensively in most of the applications of NLP as pre-processing step.

Continuous Bag-of-Words Method

The first model proposed for Word2Vec representations is the CBOW model. One layer network is trained for the representation of the word. For this network, as a parameter, the number of words taken as input at one training step can be determined by a "window". A window takes the same number of words from both sides of the output word to create input. The input for the network is the sum of one hot vector representations of the words in the window, and the true output is the one hot vector representation of the word in the middle of the word. In other words, the neighbors of a word are classified and the result must be that word. During the processing, the window slides through word list, where weights for each output word does not change. So, we have weights as many as different words, and these weights will be word2vec representation of the output words, when training finished.

Continuous Skip-Gram Model

Same logic of CBOW applies here but in reverse way. The input is one-hot-vector representation of word and output is sum of the one-hot-vector-representations of neighboring words. This time, the weights are the representations of the input word. As a result, similar words are expected to have similar representations. This is achieved according to the paper and even more, the addition and subtraction of word vectors give meaningful results. For example, vector ("biggest") - vector ("big") + vector ("small") gives a vector very close to vector ("smallest") in terms of cosine similarity.

As another example, the relation, "France is to Paris as Germany is to Berlin" can be observed by the same algebraic operation between the word vectors.

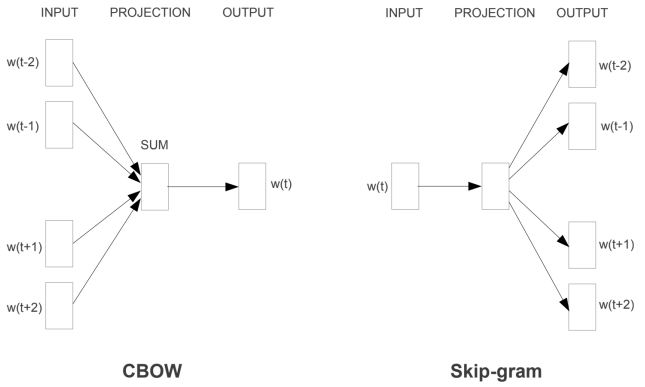

Figure 3.10: Word2Vec Proposed Methods [3]

As mentioned before, this method is used as starting step for text processing applications so that the words are firstly converted into numeric representations, word2vec vectors.

3.3.1.2 Fasttext

FastText is a library for efficient learning of word representations and sentence classification [8]. The word representation part is what we are interested. The Word2Vec representation makes word representations as distinct representations for each word, but it does not consider the internal structure of the word, which is important for morphologically rich languages, such as Turkish and Finnish [34]. So, Fasttext offers using character n-grams as an alternative to skip-gram model proposed in Word2Vec paper.

Before, the subwords, or group of n-gram characters are created, the words are adjusted such that '<' is put as prefix and '>' as suffix to the word. In this way, the starting and ending points of words are represented and words are differentiated from n-grams. For example, the where becomes <where>, and the tri-grams set becomes:

[8] `https://fasttext.cc/`

{<wh, whe, her, ere, re> }, and the her tri-gram is represented different from <her> word. These representations are added up to represent the word (Equation 3.11).

$$s(w, c) = sum_{g \in G_w} z_g^T v_c \tag{3.11}$$

where g is character n-gram, G_w is the set of character n-grams of a word, v_c is the output of one training iteration, z_g is the vector representation of the character n-gram. This summation process provides with a sliding window over a word. This makes n-grams independent of the word, so the rare words can be represented better by smaller character n-grams summation.

3.3.1.3 GloVe

There are works on the statistics of term-term, or term-document relationships in documents in order to generate low-dimensional word representations by using the global relations between terms, or terms and documents using matrix data type. Studies of term-document relationships are called Latent Semantic Analysis (LSA) and the studies of term-term relations are done under the title of the Hyperspace Analogue to Language (HAL). The problem with HAL methods, the most frequent words are everywhere and contribute to similarity measure too much.

Local context window is a method utilized in CBOW method we see in the word2vec. Using that method, one can create word vectors based on local information, which is the neighboring words. While scanning whole document, it can grasp semantic relationships patterns among terms as linear relationships. These methods suffer from not using the statistics of word occurrences in a corpus, which is also available for learning word representations as we mentioned.

So, GloVe [1] is created to generate the meaning from these statistics, and add these statistics to resulting word vectors, so they might represent that meaning. In other words, Glove is a model that combines two methods: global matrix factorization and local context window.

To implement the method, first we need to construct the word co-occurence matrix, or

Probability and Ratio	k=solid	k=gas	k=water	k= fashion
P(k|ice)	1.9 x 10^{-4}	6.6 x 10^{-5}	3.0 x 10^{-3}	1.7 x 10^{-5}
P(k|steam)	2.2 x 10^{-5}	7.8 x 10^{-4}	2.2 x 10^{-3}	1.8 x 10^{-5}
P(k|ice)/P(k|steam)	8.9	8.5 x 10^{-2}	1.36	0.96

Table 3.1: Glove word co-occurences example [1]

table. From that table, ratio of the word co-occurence probabilities are used instead of direct co-occurence probabilities. The reason is that the relationships are more intuitive in a way that, one can compare the level of relatedness of one word to two other words. For example, comparison of relatedness of each word to the words ice cream and steam can be observed from the Table 3.1.

The model is based on the ratio, so the most general form of the model is shown in Equation 3.12.

$$F(w_i, w_j, \hat{w_k}) = \frac{P_{ik}}{P_{jk}} \tag{3.12}$$

where the $w_i, w_j, \hat{w_k}$ are word vectors and the P_{ik}, P_{jk} are the word co-occurence probabilities and $P_{ij} = P(i|j)$.

In the model, a linear transformation is expected, so w_i, w_j relationship can be represented by subtraction as $w_i - w_j$. Also, the result of the model needs to be a scalar instead of a vector, so the model becomes as in Equation 3.13.

$$F((w_i - w_j)^T \hat{w_k}) = \frac{P_{ik}}{P_{jk}} \tag{3.13}$$

or under the assumption that, the context word is not different from the other words in the word co-ouccurence matrix, so the equation becomes as in Equation 3.14 under some algebraic assumptions:

$$F((w_i - w_j)^T \hat{w_k}) = \frac{F(w_i^T \hat{w_k})}{F(w_j^T \hat{w_k})} \tag{3.14}$$

If we know that P_{ij} is equal to $\frac{X_{ij}}{X_i}$, and take the F function to be exp, the last model becomes like in Equation 3.15.

$$w_i^T \hat{w_k} = log(P_{ik}) = log(X_{ik}) - log(X_i) \tag{3.15}$$

According to the table given in Table 3.1, the $w_{soid}^T \hat{w_ice}$ is forced to be equal to the $1.9 * 10^{-4}$ and this makes the Word2Vec models to grasp global word co-occurences.

3.3.2 Sentence Embedding Methods

3.3.2.1 Skip Thoughts

Skip-thoughts is based on the idea that predicting the sentences instead of words as in Skip-Gram model [4]. It is trained with an encoder-decoder (seq-to-seq) model in order to reconstruct the surrounding sentences of an encoded passage. As an aim, the similar sentences are represented with similar skip-thought vectors.

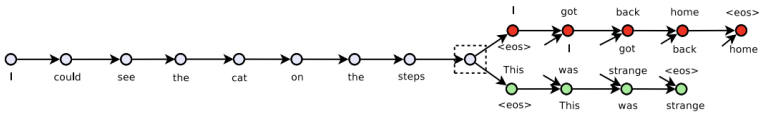

Figure 3.11: Skip-thoughts example [4]

As shown in the Figure 3.11, given a tuple (s_{i-1}, s_i, s_{i+1}) of contiguous sentences, with s_i the i-th sentence of a book, the sentence s_i is encoded and tries to reconstruct the previous sentence s_{i-1} and next sentence s_{i+1}.

The internal structure of the encoder and decoder is explained in the paper in detail.

3.3.2.2 BERT

BERT is abbreviation for Bidirectional Encoder Representations from Transformers [6]. It is a bidirectional model that is inspired mostly from ELMo and OpenAI GPT models. So, first these models need to be explained shortly.

Embeddings from Language Models

The word2vec, fasttext and glove are extensively used word-embedding models. But they construct word embeddings through whole pass of the words, and the same words have the same embedding anywhere in the text. However, for some tasks, the meaning of the word can change, so the word embedding also needs to be changed in different contexts. For example, the word "go" can take several meanings according to the

position in the context. So, the contextualized word embeddings have been presented.

In the paper defining ELMo, the contextualized word vectors are the quantities represented uniquely inside a sentence by using a a bidirectional LSTM that is trained with a coupled language model objective on a large text corpus [5]. So, the method is called ELMo, Embeddings from Language Models.

ELMo is based on the bidirectional language model, and it is generally explained as a probability estimation of a sequence from both past (forward language model) and future (backward language model) values of a token.

$$\sum_{k=1}^{N}(logp(t_k \mid t_1, ..., t_{k-1}; \theta_x, \overrightarrow{\theta}_{LSTM}, \theta_s)+ \\ logp(t_k \mid t_{k+1}, t_{k+2}, ..., t_N; \theta_x, \overrightarrow{\theta}_{LSTM}, \theta_s)) \tag{3.16}$$

As can be seen from the Equation 3.16 the aim of the bidirectional language model is to increase the log likelihood of the forward and backward passes all through sequence of N tokens. The input and the softmax parameters are shared in both forward and backward, while the LSTM parameters are different in them. ELMo is a task specific combination of the intermediate layer representations in the biLM [5].

Improving Language Understanding by Generative Pre-Training (GPT) [35]
The method, the name of the paper that proposed the method is given in the title. It is published by OpenAI community and it uses transformer decoders for language modeling. The language modeling must be familiar from previous section. It is about predicting the sequence of tokens given the token sequence except the token itself. This paper introduces us to use the transformer decoders for this task. In the Transformers section, we saw that the job of the decoder is to predict next word as it proceeds, so it is eligible for this task like the LSTM in ELMo. The method uses one pass in contrary to biLM model of ELMo. Number of decoders are given 12 in the experiments of this paper.

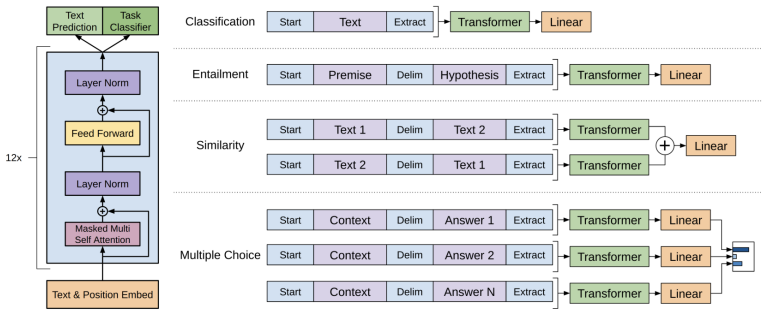

Figure 3.12: OpenAI's Decoders Model [5]

Extracted features (i.e., pre-trained model) are fine-tuned -as in the Figure 3.12- to realize different tasks.

BERT reacts to OpenAI's GPT model as: "... in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers of the Transformer (Vaswani et al., 2017). Such restrictions are sub-optimal for sentence-level tasks, and could be very harmful when applying fine-tuning based approaches to token-level tasks such as question answering, where it is crucial to incorporate context from both directions." Even if it embraces bidirectional language model, it does not concatenate two uni-directional model into one. BERT utilizes the transformers encoders as the architecture rather than the decoders as in ELMo. It also uses two novice methods to realize pre-training: Masked Language Model (MLM) and Next Sentence Prediction (NSP).

BERT utilizes WordPiece embeddings to represent the inputs. [CLS] is the first token of the sequence and [SEP] is the token to separate sequences, or specifically sentences. Sentences are separated by this token and a learned embedding is added to every token that indicates whether this token is in Sentence A or Sentence B. This information will be used in NSP task. So, the input embedding will be like in the Figure 3.13.

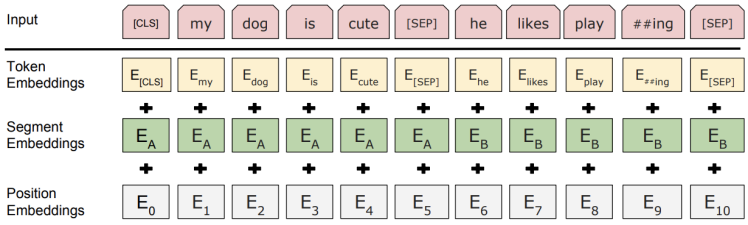

Figure 3.13: BERT Input Structure [6]

The MLM indicates that 15/100 of the words are masked inside the sequence. The reason is coming from the restrictions of the language models that the token itself will take attention in transformers models when the sequence is predicted through past and future passes of that token. With MLM the output is masked words to be predicted instead of whole sentence, or any denoised version of that sequence. To improve the performance for the some following tasks like question answering, next sentence prediction is also applied during pre-training. It is done according to the output.

The final hidden state corresponding to [CLS] token is used in the next prediction, so it is utilized in the aggregate sequence representation for classification tasks.

BERT has two different models in order to show the results. The $BERT_{BASE}$ is for comparing the model to ELMo in tasks and the other model is $BERT_{LARGE}$. $BERT_{BASE}$ has 12 layers (L=12), 768 hidden units (H=768), and 12 self-attention heads (A=12), 110M total parameters; the $BERT_{LARGE}$ is (L=24, H=1024, A=16, Total Parameters=340M).

3.3.2.3 RoBERTa

RoBERTa is an abbreviation for "Robustly Optimized BERT Pretraining Approach" [36]. The BERT architecture is changed in simple ways and pre-training operation is duplicated. The authors claim that "We find that BERT was significantly undertrained, and can match or exceed the performance of every model published after it [36]." Over this they extended BERT as in the following ways before training:

- Training the model longer, with bigger batches, over more data;
- Removing the next sentence prediction objective;
- Training on longer sequences; and
- Dynamically changing the masking pattern applied to the training data.

3.3.2.4 Sentence-BERT

Sentence-BERT is an extension to BERT and RoBERTa methods in order to improve the semantic similarity between sentences so that they can be compared using simple similarity measures like cosine similarity and euclidean distances [37].

As an extension:

- It applies pooling in order to derive a fixed sized sentence embedding. Three ways are possible: Using only [CLS] token embedding, averaging token embeddings (MEAN) and max strategy (MAX). The default is MEAN.
- For fine-tuning, the siamese and triplet networks are used in order to capture semantically meaningful and easily comparable embeddings. The network architecture and loss functions are changed according to the task.

The network architecture will be explained with Triplet Objective function. The others are explained in the paper as well. Two semantically related papers are taken and one of them is called negative and another one is called positive sentence. Also, a semantically unrelated sentence is chosen as negative sentence. The weights learned from anchor example is given exactly to the networks of positive and negative samples. The results are calculated and given to loss function. The objective of the loss function is that anchor result is similar to positive and different from negative. The similarity measure is euclidean in the formula and the ϵ ,i.e. margin, hyper-parameter is the measure which decides how far away the dissimilarities should be. For example, say, margin = 0.2 and difference between anchor and positive example equals 0.5, then difference between anchor and positive example should at least be equal to 0.7.

With these architectures, Sentence-BERT improves the BERT and RoBERTa in the way of semantic similarity. Without it, the claim is that BERT gives even worse results than the GloVe [37].

3.3.2.5 Language Models are Unsupervised Multitask Learners (GPT2)

GPT is explained inside BERT section as a subtitle. This is an extension to GPT from the same community, OpenAI [38].

Firstly, the dataset quality is improved. It is stated that crawled data has very unqualified data. So, they utilized from human comprehension for filtering the web pages. After finding a heuristic helper via chosen Reddit outbound links, the WebText was created. Wikipedia data is extracted so that the testing and comparing with other methods will be more effective. Secondly, they changed the way inputs are represented. Instead of just word representations, byte representations are also benefited. By using Byte Pair Encoding (BPE) representation, practical middle ground between character and word level language modeling [38] is preserved. Lastly, some extensions to GPT architecture are added. As an example, layer normalization is moved down to the input of each decoding block, so one layer normalization was added after the self-attention block.

3.3.3 Document Embedding Methods

3.3.3.1 Doc2Vec

Doc2Vec creates a dense vector representation of a piece of text such as paragraphs, sentences and documents. Rather than the Bag-of-Words model, it takes the ordering of the words into consideration. Actually, the Word2Vec models inspired this work. Two methods are offered that are very similar to CBOW and Skip-Gram models that are explained in the Word2Vec section. The first model is Distributed Bag of Words version of Paragraph Vector (PV-DBOW) which is similar to CBOW. As we can see from the Figure 3.14, the only difference to the CBOW is the paragraph matrix, which contains one vector per paragraph. The paragraph vector per paragraph does not

change in the process of window sliding. Note that, the vectors for each word and for each paragrah is unique. So, there are number of parameters of vector size X words count + paragraph vector size X paragraphs count for the model.

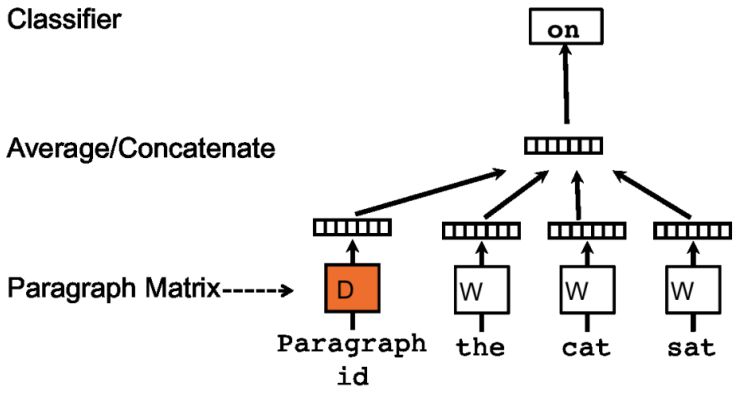

Figure 3.14: PV-DBOW model [7]

The loss function also gives intuition about the method. In the Equation 3.17, the log probability of predicting the word of the window is maximized for all words in the paragraph. The first and last k words are not included to be the output of the model since the left of first k words and the right of last k words does not have enough (k) inputs.

$$\frac{1}{T}\sum_{t=k}^{T-k} log(p(w_t|w_{t-k}, w_{t-k+1}, ..., w_{t+k-1}, w_{t+k})) \tag{3.17}$$

The other model offered is called Distributed Memory version of Paragraph Vector (PV-DM). It is similar to the Skip-Gram model but the paragraph vector is used as input instead of the middle of word of the window (Figure 3.15).

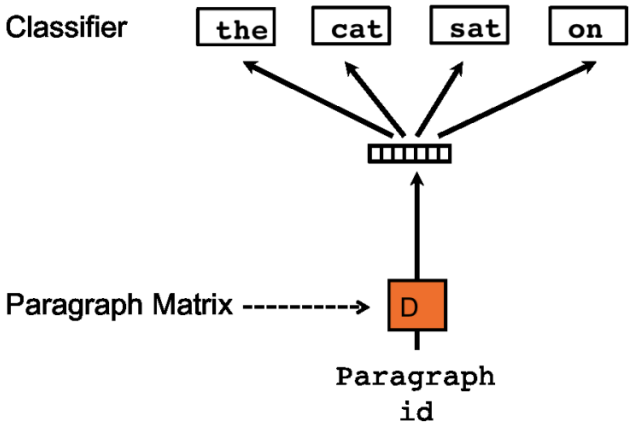

Figure 3.15: PV-DM model [7]

3.4 Clustering Methods

In this section, the utilized clustering methods are explained [39].

3.4.1 K-Means

K-Means is a well-known clustering method that constructs clusters of equal variance. Number of clusters are defined before applying the algorithm and the mean of the points of elements, i.e. centroids, define clusters. In the algorithm:

1. The centroids are initialized using a method, such as choosing K<N points from dataset sized N.

2. Each point is put into the most similar cluster according to some similarity of point to the centroid of the cluster.

3. The centroids of the clusters are set to the mean of the cluster points.

4. Step 1 and step 2 is repeated until the method converges

3.4.2 Agglomerative Clustering

Agglomerative Clustering is a kind of hierarchical clustering in a bottom-up way. The algorithm is:

1. It starts with that the most similar two data points are put into one cluster.
2. The most similar clusters or cluster-data points are linked by treating the cluster as single data point by averaging the data points inside the cluster (Average Linkage), by taking the most similar points between clusters(Single Linkage), by using Euclidean distance (Ward Linkage), etc.
3. Step 2 is repeated and a tree was constructed at the end, called dendrogram.

3.4.3 DB-Scan

The DBSCAN algorithm views clusters as areas of high density separated by areas of low density. The two parameters defined for the algorithm are min-sample and eps. The min-sample is the core samples of the cluster and the eps decides the neighbors of the core samples. Number of clusters is decided by the algorithm and dataset with the same order is going to give the same clustering results.

3.5 Clustering Metrics

To measure the performance of the clustering, the following methods are used in this thesis. The silhouette score, Davies-Bouldin index and Calinski Harabsz Score does not require ground truth cluster information, while the other requires it [39].

3.5.1 Silhouette Score

Silhouette score measures the performance by the formula in Equation 3.18:

$$\frac{b-a}{max(a,b)} \tag{3.18}$$

where b is the distances of the data point to all other data points in the same cluster and the b is the the distances of the data point to all other data points in the next most similar cluster. To get a high silhouette score, the similarity of the points in the same cluster must be high and the similarity of the points in different clusters must be low. The range of scores is between -1 and 1.

3.5.2 Davies-Bouldin (DB) Index

DB index is a measure of average intra-clusters and inversely proportional with the distances between clusters. Lower value implies better clustering, where 0 is the ideal value. This is clustering metric with no labels (Equation 3.19).

$$\frac{1}{k}\sum_{i=1}^{k} max_{i\neq j}\frac{s_i + s_j}{d_{ij}} \tag{3.19}$$

where s_i is the measure of distances of all points inside cluster to the centroid of the cluster and d_{ij} is the distance between two cluster centroids.

3.5.3 Calinski Harabasz Score

The index is the ratio of the sum of between-clusters dispersion and of within-cluster dispersion for all clusters (where dispersion is defined as the sum of distances squared) [39]. This is also clustering metric that requires no labels.

3.5.4 Rand Index

Rand Index requires cluster labels to work. Rand index measures the similarity between ground truth and the predicted cluster values by not taking the permutations into consideration. Actually, the adjusted rand index version is used in the context of this thesis. It ensures that the values smaller to zero are random, where rand index cannot be sure of that. The value range is between 0 and 1, 1 is the best value. The unadjusted rand index is as in the Equation (3.20)

$$\frac{a+b}{C_2^{n_{samples}}} \tag{3.20}$$

where a is the number of point pairs that are in same cluster in ground truth and in same cluster in predicted clusters at the same time. b is the number of point pairs that are in different clusters in ground truth and in different clusters in predicted. The denominator is the all possible pair count. The adjusted rand index is an improvement over the unadjusted one (Equation 3.21).

$$\frac{RI - E[RI]}{max(RI) - E[RI]} \tag{3.21}$$

3.5.5 Homogeneity Score

Homogeneity Score requires cluster labels to work as rand index. Homogeneity is defined as each cluster contains only members of a single class. There is a completeness as well and v measure as the weighted average of them. But the homogeneity score is used in the context of this thesis.

3.6 Hyperopt

Hyperopt is a hyper-parameter optimization tool used to optimize hyper-parameters of machine learning models by using Bayesian optimization method that is one of the most efficient methods of function minimization.

Advantages of Hyperopt are summarized as follows:

- leverages smoothness without analytic gradient,
- handles real-valued, discrete, and conditional variables,
- copes with hundreds of variables, even with budget of just a few hundred function evaluations [40]

In order to use Hyperopt, a configuration space and one evaluation function needs to be defined. For configuration space, data distribution requires to be specified in addition to hard bounds to the data. By changing values in this configuration space, the evaluation function outcomes are minimized. There are two algorithms in the module

that can be followed during this process: random search and TPE. Each experiment is called and memorized in a Trials object. Without going into further detail, in order to optimize the clustering scores, the hyper-parameters of embedding methods and parameters of clustering methods are optimized in the context of the thesis.

CHAPTER 4

BASIC ARCHITECTURE OF THE EMPLOYED PIPELINE

In this thesis, the main aim is to cluster the documents according to modern approaches, compare the results and comment on the approaches that are used for document clustering. For this, an experimental setup is created. It has the structure given in Figure 4.1 and explained in this section in detail.

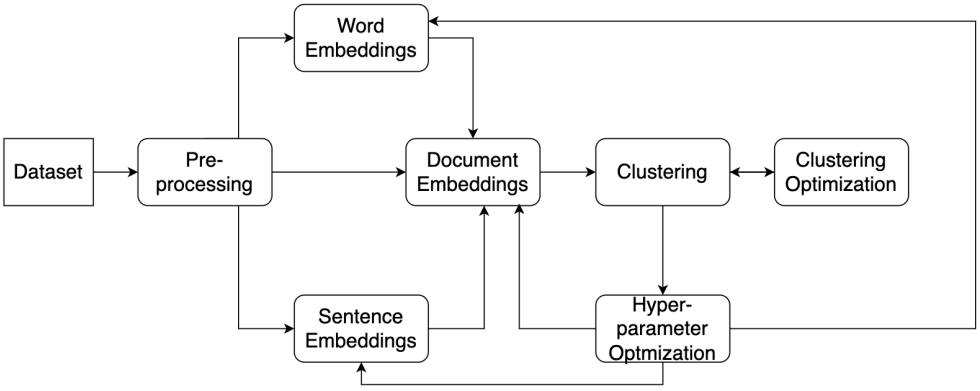

Figure 4.1: Project Structure

In summary:

1. Data collection is pre-processed.
2. Different kinds of embeddings are extracted from documents, and these embeddings are combined to have document embeddings.
3. These documents are clustered using K-Means, Agglomerative and DBSCAN

clustering methods.

4. As a clustering metric, the silhouette index is calculated for 5, 10, 15 and 20 clusters of documents, and the scores are averaged.

5. Hyper-parameters of the embedding methods are optimized in order to maximize that score.

6. For each trial, all the clustering methods with given cluster count are subjected to 20 trials with Hyperopt for further optimization with clustering method parameters.

7. This process is repeated from step 2 for each embedding method and embedding method option, which is the composition strategy to have the document embeddings.

4.1 Dataset Collection

There are two data collections and they contain scientific papers. They are collected from PDF files manually and saved into text files. During collection, a lot of papers were not public, so their only abstract part is collected and marked as putting '-lack' into filenames.

The first data collection is the papers of the scientists participating in the COST European Cooperation in Science and Technology action. This is the large collection containing 599 papers, where 193 papers only include the abstract of the papers and 406 full text of papers which do not include formulas, most figure captions, and references. The small data collection is the collection of papers that mention about the Göreme and Cappadocia. In that collection, 9 of the papers have only the abstract of the papers, 47 of them contains full papers. In these, the figure captions, formulas and references are also excluded. The last one is the Covid19 dataset which contains only abstract of over 30.000 papers related to Covid 19 [1].

[1] Covid19 Papers Collection

4.2 Pre-processing

The pre-processing is required to make data less noisy and create necessary files. As a first step, cleaning is applied to data collections so that some non-meaningful parts existed in them. As an example to these, non-ASCII characters, non-meaningful characters while copying the PDF, and superscript numbers that give reference to papers. Also, the hyphen usage in the papers made words noisy. The non-ASCII characters are extracted and the hyphen problem is solved by merging the parts in the both sides of hyphen.

4.3 Embeddings

Embeddings are the vector representations for chunks of a text. The chunks can be words, sentences and even the documents. These methods are used and combined if needed in order to construct vector representations for documents of the same dimension, so we can compare and cluster them. The embeddings are actually the neural network models, so they have hyper-parameters. In this section, used hyper-parameters of these methods, how these methods are used and the options for combining the vectors of them is explained. Used combination strategy, i.e. embedding method options, are explained before it to understand which combination strategy better fits with the embedding method.

4.3.1 Combination Strategies

4.3.1.1 Averaging Word Vectors (avg)

The word vectors are simply averaged to create the document embedding

4.3.1.2 Term Frequency-Inverse Document Frequency (tf-idf)

tf-idf can be utilized as weights to words since its general purpose is to emphasize the more important and rarely used words in documents, and understate the mostly used,

less relevant words, such as stop words. In this way, a weighted average approach is followed in addition to only averaging.

4.3.1.3 Sentence Average (sentence-avg)

Sentence embeddings is averaged according to this strategy.

4.3.2 Embedding Methods

Word2Vec : Word2Vec is utilized to create word-embeddings with the following hyper-parameters:

- **Vector Size (vector_size)**: Resulting vector size, which is equal for all word vectors.
- **Window Size (window)**: In word2vec algorithm, there is a sliding window, in which there are input or output words according to the model for each input. The size of it is decided with this parameter.
- **Minimum word frequency (min_count)**: This parameter ignores all words with total frequency lower than this.
- **Loss Function (hs)** : This is a binary parameter. If the value is 1, then the hierarthical softmax is used as the loss function. If the value is 0, then the negative sampling is used as the loss function.
- **Learning Rate (alpha)** : Learning rate is the hyper-parameter for almost all the neural network models, which is a constant inside the gradient descent algorithm. It decides the rate of learning and it can change over time according to the method. The starting value for the learning rate is set before starting the training.

Words are tokenized according to the nltk.tokenize library and fed to the network. As a combination strategy, the tf-idf and the averaging are used.

Fasttext: Fasttext is another method for word-embedding which also includes the character n-grams for training. The hyper-parameters are given as in the following:

- **Model**: CBOW and Skip-gram are two models offered for word2vec approach. One of them is chosen as the hyper-parameter.
- **Learning Rate (lr)**: The learning rate parameter is the same as the one in word2vec model.
- **Dimension (dim)**: It is the same as vector size in word2vec model: resulting vector size, which is equal for all word vectors.
- **Window Size (ws)**: In Fasttext algorithm sliding window size exists as in word2vec, but the character n-grams and words both can be the input in contrary to word2vec model.
- **Minimum word occurences (min count)**: Words occur in the document several times. To train the model with words that occured in the documents more than some number, this parameter is used.
- **Minimum character n-gram (minn)**: It defines the minimum n for the n-grams.
- **Maximum character n-gram (maxn)**: It defines the maximum n for the n-grams. This must be greater than he minimum character n-gram.
- **Negatives sampled (neg)**: This is the number of negatives sampled (in Negative sampling algorithm).
- **Word n-grams (wordNgram)**: Word n-grams are the pairs given in the opposite side of the sliding window. In CBOW, this is the size of the output, and vice versa in Skip-gram.
- **Loss Function (loss)**: Fasttext has more options as loss function than the word2vec: "Negative Sampling", "Hierarchical softmax", "softmax", "One-Vs-All".

All the documents are put to a text file in the pre-processing step, then this is fed to the network. As a combination strategy, the tf-idf and the averaging are used.

GloVe: GloVe is a another method that outputs word vectors. It uses global word co-occurences in addition to local representations obtained via word2vec model and combines them in one framework. The hyper-parameters are:

- **Minimum frequency of words to include (vocab_min_count)**: It is the same as "min count" parameter in word2vec.
- **Vector Size (vector_size)**: It is the same as "vector_size" parameter in word2vec.
- **Maximum Iteration Count (max_iter)**: This is the number of iterations of the algorithm.
- **Window size (window_size)**: It is the same as "window" parameter in word2vec.
- **Maximum co-occurrences (x_max)**: This is the maximum number of word co-occurrences used as a weight in the loss function. Weights will be between 0 and 1. For the co-occurences greater than x_max will have the 1 as weight.

The method is used according to the guide in the GitHub [2] link of the method. The parameter file is changed according to the format, and the full data text file created in pre-processing step is used as the corpus as in the fasttext. It has been run through the synchronous shell process created. In addition to the training like in fasttext and word2vec, the pre-trained embeddings are also used and they are averaged to create document vectors.

Skip-Thoughts: Skip thoughts method trains word vectors over LSTM to create sentence vectors. Its resulting vectors are concatenation of uni-skip vector (dimension:2400) and bi-skip vector (dimension:2400) for each sentence. Total vector size, i.e. dimension, is 4800 for each sentence vector. There is no hyper-parameter to use for this method in our setting.

BERT: BERT is a method that evaluates the subword tokens in their context and the unit of that context is the sentences. So, the sentence embeddings are taken into consideration instead of the token representations one by one. Sentence embeddings are taken as the average of embeddings of tokens inside it in our case, where token embeddings are calculated via the average of the weights of last 4 layers.

[2] https://github.com/stanfordnlp/GloVe

As a method, BERT's chosen model representations, the "uncased_L-2_H-128_A-2" in our case are taken as the checkpoint and the local representations fed to network as sentences are calculated. According to this, the token embeddings, then the sentence embeddings are taken as explained above.

The GitHub page and the original implementation is used, but there is only hyper-parameter that we could manage to change in the hyper-parameter file defined for that BERT implementation. However, BERT is used with more hyper-parameters with the SciBert.

- **Dropout probabilities for attention prob. (attention_probs_dropout_prob)**: It is the dropout ratio for the attention probabilities.

Only the sentence averaging is utilized, where average of sentence embeddings is taken as document vector.

Sentence-BERT: Sentence-BERT is a method that utilizes the BERT embeddings and add additional layers and make the all sentence vectors to be related each other instead of just pairs of them as in BERT. The hyper-parameters used are:

- **Pre-trained Model (pretrained_model)**: These are the pretrained models as initial checkpoint as in the BERT model. In BERT model, there is one model, but there are several in here.
- **Batch Size (batch_size)**: Batch input size for feeding network.
- **Normalizing embeddings (normalize_embeddings)**: To normalize output as if their sum is equal to one or not

Sentence averaging is utilized as combination strategy but in two different ways. In the first one, sentences of documents are fed to network, document by document. In the other one, all of the sentences are fed and the results are obtained.

Transformer Methods (RoBERTa, GPT2, BERT with SciBERT): Three methods of contextual word/token embedding are all using transformers and implemented via the same library, so they are all using the same hyper-parameters and same combination strategies applied to all. For all the models, there is one pre-trained model

used as an initial checkpoint: roberta-base for roberta, gpt2 for gpt2 and scibert for bert. SciBERT is chosen because the local data collection is also scientific papers, the others are the base pre-trained models of those methods.

The hyperparameters are:

- **Number of train epochs (num_train_epochs)**: The number of epochs the model will be trained for.
- **Learning Rate (learning_rate)**: Learning rate of the network
- **Warm up Steps (warmup_steps)**: Number of training steps where learning rate will warm up.
- **Maximum Sequence Length (max_seq_length)**: Maximum sequence length the model will support.

Same as the Sentence-BERT, two ways of sentence embeddings are used as combination strategy.

Doc2Vec: This method works in a way similar to word2vec model and only difference is having one additional embedding that has one additional layer for each document that is the document embedding itself. The hyper-parameters are:

- **Vector Size (vector_size)**: Resulting vector size, which is not necessary to be equal to the size of word vectors.
- **Window Size (window)**: In doc2vec algorithm, there is a sliding window as in word2vec, in which there are input or output words according to the model for each input. The size of it is decided with this parameter.

4.4 Clustering

Clustering is the output of whole system. There are three clustering methods, five clustering metrics to evaluate clustering quality. Parameters of clustering methods, clustering method and metric usage are explained in this section.

4.4.1 Clustering Methods

4.4.1.1 Agglomerative Clustering

Agglomerative Clustering is a bottom-up approach, and uses pair-wise distances between documents. This method is used with the following parameters:

- **linkage**: After measuring distance between the document vectors, this parameter decides which distance strategy shall be used. The options are 'ward', which minimizes the variance of the clusters being merged according to some center point; 'average' takes average of distances between all data points from both clusters, 'complete/maximum' linkage measures distances between all data points from both clusters and takes maximum, 'single' does the same calculation and takes minimum distance.
- **affinity**: While measuring distance between the document vectors, this parameter decides which distance method shall be used. 'euclidean', 'l1', 'l2', 'manhattan', 'cosine' are the methods for affinity between two document vectors to use.

It should be noted that Ward linkage strategy only uses euclidean distance.

4.4.1.2 K-Means Clustering

K-Means is a method that uses centroids and chooses data points most similar to them. Then, the new centroids are determined and the centroids converge at the end. This method has the following parameters:

- **init**: For initial choice of cluster centers, there are two ways to follow. They are initialized randomly or 'kmeans++' is picked up and the centers are initialized in a wiser way.
- **algorithm**: There are two algorithms that can be followed during processing, one of which, 'elkan', is alternative to EM-style algorithm 'full'.

- **n_init**: Number of centroids, or equally cluster count.
- **max_iterint**: It is used to restrict maximum number of K-Means iterations without waiting convergence.

4.4.1.3 DBSCAN

This is a clustering method that does not get the clustering count as input and this is run for all the methods, their combination strategy pairs. Its parameters are:

- **eps**: Distance between two points where points in the same neighborhood.
- **min_samples**: Number of samples in the neighborhood that decide if a point is a core point or not.

4.4.2 Clustering Metrics

Five clustering metrics exist in the scope of this thesis. Three of them does not need the cluster labels as input: silhouette score, Calinski Harabasz Index and Davies-Bouldin Index. The other two need the true labels of clusters to work: Rand Index and Homogeneity Score.

To make Rand Index and Homogeneity Score metrics work, the 200 of the 599 documents in larger collection and 26 of 56 of documents in smaller collection is labeled by human observation. Before that, the trials are run and benefited during labeling. These are not enough to calculate clustering metric since the metrics need all the documents to be labeled. The following algorithm is used to approximate the metric evaluation:

Clustering is calculated first. Each cluster is labeled with the the major label of labeled documents in that cluster. If there is not any labeled document in that cluster, the cluster is labeled with a new cluster label. After each cluster is labeled, the Rand Index and Homogeneity Score is applied.

These metrics are calculated for clustering points and only silhouette score of 5,10,15,20 clusters are used for hyper-parameter optimization.

4.5 Hyper-parameter Optimization

Clustering, the output of whole system, is optimized continually with new trials by updating the hyper-parameters of embedding methods. This is obtained via a Bayesian Optimization method.

Tree of Parzen Estimators is the name of the Bayesian Optimization method used. The inputs are the hyper-parameters of one embedding method and the output is the average silhouette score of 5,10,15,20 clusters when Agglomerative clustering is used.

This setup is used to search for best clustering for each method, combination strategy pair on its own. This optimization loop lets us automatize the selection of hyper-parameters. In addition, hyper-parameters can be chosen manually so that, the researcher can select hyper-parameters and those hyper-parameters are added to experiment as if it is selected by the algorithm.

In addition to this mechanism, parameters of all the clustering methods for a given cluster count are optimized according to silhouette score with 20 trials for each trial for further optimization.

CHAPTER 5

CLUSTERING ANALYSIS TOOL

In order to implement the document clustering framework explained in the previous chapter, an automatized tool is built with Python programming language as Dash application. It has layout and callbacks at some order. Layout is the view of the application and the callbacks are the listeners for user interactions. The helper classes for the callbacks are coded according to some structure [1].

This application has mainly three parts, each having different implications in the application.

5.1 File Upload and Pre-processing Part

5.1.1 File Upload

In this part of the application, the collection of documents as zip file is uploaded. Each document must be in a text file, the file extensions of which must be '.txt'. These files can be under different folders inside the zip file, they are parsed and each file is uploaded as a single document. The uploaded documents are saved into the 'initial-files/documents' folder. The second box is for the annotations. A json file is required to be uploaded for the true labels of clusters.

[1] UML diagram of Project Classes

5.1.2 Uploading Documents/Pre-processing Part

The documents uploaded are put into 'initial-files/documents' folder and they are read from there for processes. The papers are represented as key-value pairs, where keys are the paper names that must be supplied in the first line of the document, and values are the document text. During uploading the documents as a zip file, a relatively long process goes on and the required files for the Trials Framework Part are created in the following order and put inside the 'initial-files' folder:

1. Sentences in the documents are extracted and put into 'sentences.txt' file. As a format of the file, the sentences are written in the order inside the document and there is 2 newline between sentences of different documents.

2. Tf-Idf calculations are done for both the Fasttext and the GloVe part. GloVe vocabulary and Fasttext vocabulary are different from each other, so two different 'tf_idf' files are created for both of them. These files have one dictionary with two items:

   - tf_idf_dict: tf-idf values of words as a dict.
   - tf_idf_paper_total_dict: Sum of tf-idf values of words in a document in order to normalize the word vectors, weighted with tf-idf values.

   As a note, in the word2vec method, tf-idf values of gloVe are used.

3. token_info file is created. This file contains information about graph demonstrations that will be explained in the Result Demonstration and Manipulation Part. This holds dictionary values with two keys:

   - token_papers : This is a mapping between token and papers having these tokens.
   - token_counts : This holds information about how many times each token exists in whole collection.

4. In this step, the 'fullData.txt' file, which is concatenation of all documents, is created. This will be used in Fasttext training.

5.1.3 Uploading Annotations File

A json formatted file is required to give true cluster labels to the clusters. These true labels are required for the Rand Index and Homogeneity Score. This json file is not compulsory at first. It can be uploaded later and the 'Re-Calculate Clusters' button needs to be clicked just after uploading. The structure of the file is a json object and transformed into dictionary in Python. Keys of the documents are the cluster label name and the values are lists of document names in that cluster. This file can have any number of clusters, which will be adapted to different number of clusters with an approximation method as described in Section 4.4.2.

5.2 Trials Framework Part

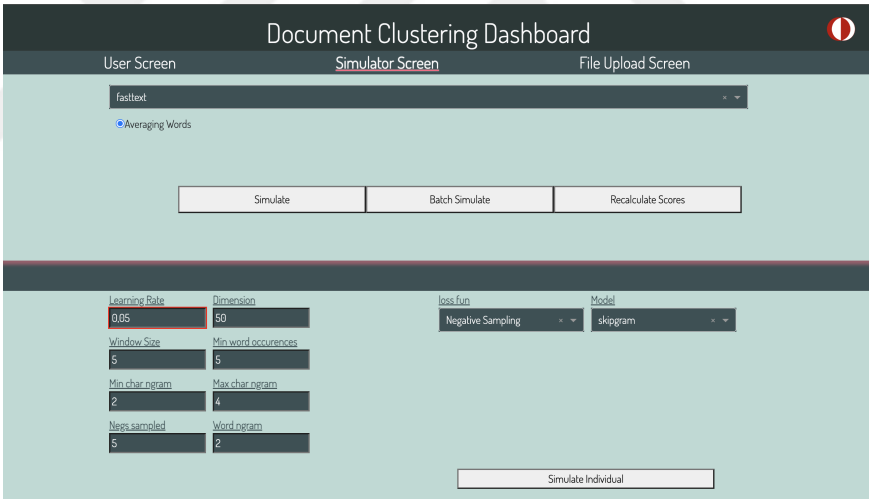

Figure 5.1: Triggering One Trial/Pipeline via Automatic or Manual Hyper-parameter Selection

This part is the backbone of the testing framework. Trial refers to one full experiment. One full experiment is explained in previous chapter and shown at Figure 5.1. According to that, the following steps are followed to realize one full experiment with the testing framework.

5.2.1 Usage

Embedding method (e.g. Fasttext), embedding method option (e.g. tf-idf weighting or averaging) are chosen. Then, the hyper-parameters of the method needs to be determined. For that, there are two options: automatically and manually. If 'Simulate' button is chosen, the hyper-parameters are chosen automatically, which means according to the TPE for this testing framework. The choice is done as black box. The result is calculated by the following function of Hyper-opt module.

```
fmin(fn=self.update output obj fun,
        space=self.search space,
    algo=tpe.suggest,
    max evals=len(trials) + 1,
    trials=trials
)
```

"fn" parameter needs to be the objective function. It is designed to be the whole function that calculates the document embeddings, cluster them, calculate silhouette score for 5,10, 15, 20 clusters, average them and return the negative of the averaged score. The negative is chosen since the score needs to be minimized, silhouette score needs to be bigger to be better. The chosen hyper-parameters are given as parameter to this objective function so that the embedding method can use them.

Actually function output is optimized using the inputs with a given range in search space parameter of the function. So, the search space is the ranges of hyper-parameters. Three options are used for defining the ranges. It can be an integer range, float range or discrete range that contains options.

As mentioned before, the optimization algorithm is TPE that is defined with algo parameter.

max_evals are the maximum number of calculations need to be done. If the function converges, the calculation is finished. For our case, this value is set to the one more of the trials count, which means one more trial is added to the existing trials.

The Trials object can be thought as the memory of the trials and keeps the previous

input (hyper-parameters), and the output (silhouette score) values, and given to the "fmin" function. "fmin" chooses the hyper-parameters using the mentioned Bayesian approach, and according to calculation results of objective function, new trial values are saved to trials object for further use. It is said that, there is another option which is that hyper-parameters can be chosen manually. This is realized by manipulating the "Trials" object. A new trial instance is crated by utilizing the generate_trial method of Hyperopt, whose arguments are a new id and manual parameters determined by GUI. This new trial instance is fed to the existing trials object through the insert_trial_docs method. This action is triggered by the 'Simulate Individual' method. Trials work sequentially, so this one unit of trial needs to be awaited to initialize the next.

5.2.2 Data Storage Format

Trials and their parameters are stored in MongoDB database as collections are the embedding method names. Each collection contains two kinds of data.

First one contains id, the Trials object and the corresponding embedding method option. The Trials object is saved as an object and serves for the continuum of trials. Number of trials object is equal to the number of embedding methods. For example, there are two Trials objects for Fasttext collection: one for averaging and one for tf-idf. So, a given embedding method-embedding method option pair, there is one trial history.

The second one contains the document embeddings, parameter values, the corresponding embedding method option and avg_score, scores and time. The document embeddings are kept with their document name. These are kept to calculate different clustering methods for different parameters. The "avg_score" is the average of silhouette scores for 5, 10, 15 and 20 clusters. "score" contains information about different clustering metrics for different clustering methods. These are calculated values for default parameters of clustering methods. In section 5.3, we can see that these different results can be recalculated via different clustering parameters on the fly by using document embeddings from database. Hyper-parameters for each embedding method is kept as key-value pairs in this part as well. Time parameter keeps the time of whole operation: calculation of vectors, clustering operations and db read and write

operations.

Before each trial operation, the trials object keeping history of chosen embedding method-embedding method option pair is retrieved. This is used in operations. After operation, the trial object is updated and the new result is inserted according to the format. The best document embeddings for embedding method-embedding method option pair are retrieved by graphs and tables from GUI to visualize them. These and other database operations such as connection are implemented in TrialsDAO class.

5.2.3 Embedding Method Calculations

Embedding method calculations are implemented using Template design pattern. So, there is a class called Pipeline that does common operations among different embedding methods. One trial operation is implemented fully. The hyper-parameter optimization with "fmin" function is implemented as explained before. For that, the training is implemented according to each embedding method. These are called from classes created for each embedding method-embedding method option pair (e.g. FasttextTfIdfPipeline). These classes inherit the main Pipeline class, so the parent class only calls the necessary training function from child class. While implementing the training, the original implementations of embedding methods are preferred as much as possible. Also, the database operations are implemented in this parent Pipeline class.

The training for each embedding method is done in the following ways:

Fasttext Fasttext training is done via train_unsupervised [2] which takes the "fullData.txt" file as parameter, which is concatenation of all documents, created in the pre-processing part. The other parameter of this method is chosen hyper-parameters for Fasttext. Created word embeddings are compounded for all each file by tf-idf (via FasttextTfIdfPipeline) and averaging method (via FasttextPipeline) options and the resulting document embeddings are returned.

[2] https://Fasttext.cc/

gloVe gloVe training is implemented using the Github page [3] of the method. It is implemented as shell process opened from Python code. The chosen hyper-parameters are updated through the ".sh" file being run. Created word embeddings are compounded for all each file by tf-idf (via GloveTfIdfPipeline) and averaging method (via GlovePipeline) options and the resulting document embeddings are returned as in Fasttext. The only difference is gloVe and Fasttext vocabulary is different. Also, gloVe has the pre-trained embeddings, which do not require further training but the pre-trained word vectors compounded through averaging (via GlovePretrained-Pipeline). For that the pre-trained Wiki models published by GloVe creators.

word2vec word2vec is trained through the "gensim" library. The documents are tokenized with nltk library and a 2D list is created where tokens for each document corresponds to a 1D list. This 2D list and hyper-parameters are fed into word2vec method of "gensim" library. The created word vectors are compounded through averaging (Word2VecPipeline) and tf-idf (Word2VecTfIdfPipeline) method options. tf-idf uses the pre-calculated gloVe tf-idf values for tokens.

bert The Github implementation for the BERT is utilized to implement BERT method. The following shell code is run as a process from Python code.

```
python extract features.py
    input file=/tmp/input.txt
    output file=/tmp/output.jsonl
    vocab file=BERTBASEDIR/vocab.txt
    bert config file=BERTBASEDIR/bert config.json
    init checkpoint=BERTBASEDIR/bert model.ckpt
    layers=1,2
    max seq length=128
    batch size=8
```

In that formula, the "input.txt" is the same as the "fullData.txt" used in the Fasttext. The output.jsonl is the resulting weight values,i.e. possible embedding values, for each layer. The other files are given by chosen model, which is the "uncased_L-2_H-

[3] https://github.com/stanfordnlp/GloVe

128_A-2" in our case. The last two layers are averaged to get the token embeddings. Each token embedding is averaged to create sentence-embedding and each sentence embedding is averaged to created document embedding (via BertPipeline). Hyper-parameters inside the "bert_config.json" are changed if they are replaceable for training.

Other Contextualized Word Embedding Methods/Simple Transformer Models These methods use the simpletransformers [4] application in order to feed hyper-parameters easily. Like the BERT, the token embeddings are averaged in order to create sentence embeddings and the sentence embeddings are averaged to create document embeddings. The documents are trained individually (via AfterBertPipeline) and the whole document is trained in one pass (via AfterBertPipeline2) as different method options. As a note, the embedding methods used are RoBERTa with pre-trained roberta-base model, GPT2 with pre-trained GPT2 model and BERT with pre-trained SciBERT model.

Skip-Thoughts Skip-thoughts is designed as if it takes no hyper-parameters. So, it is trained only once and stored into a file instead of database. The sentence embeddings created as the result of this method are averaged to create document embeddings (via SkipThoughtsPipeline).

Sentence BERT For SentenceBERT, its open-source Python method is used. The sentences and hyperparameters are fed to this method. The resulting sentence embeddings averaged to create document embeddings (via SentenceBertPipeline).

5.3 Result Demonstration and Manipulation Part

The clustering results are demonstrated in the GUI of this framework in different ways. These are:

[4] https://simpletransformers.ai

5.3.1 Clustering Result Table and Line Graph

Clustering results according to the chosen clustering metric is shown in this part for 5, 10, 15, 20 clusters to give the idea about clustering performance. Also, there is a line graph from 2 to 20 clusters (Figure 5.2).

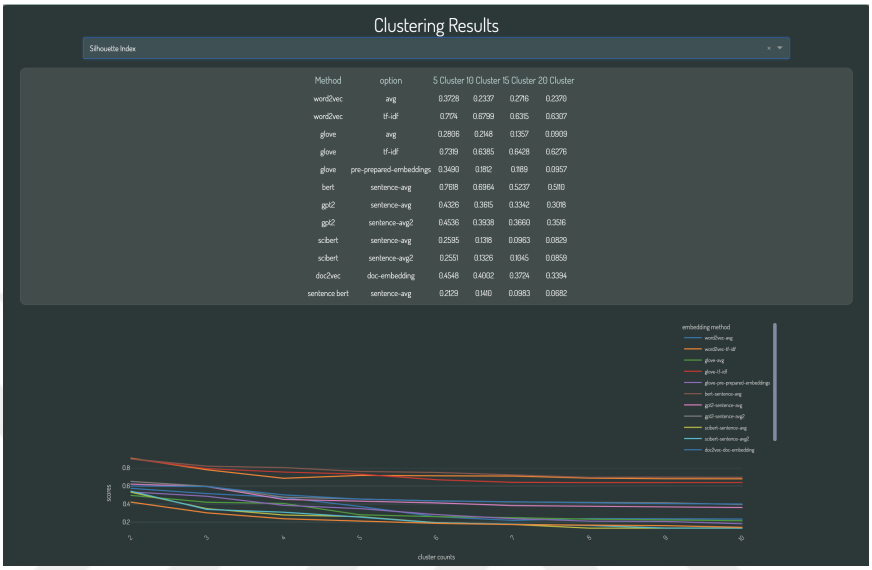

Figure 5.2: Score Over Cluster Counts Table + Line Graph

5.3.2 Clustering Result Correlations

In order to understand how clustering metrics are related, this graph is used. In this way, it can be understood that, the metrics needing human help and not needing human help can be compared as well.

5.3.3 Clustering Result Scatter Graph

By using different visualisation tools (t-SNE and PCA), the document embeddings are projected into 2D and shown as a scatter plot in this part.

5.3.4 Word and Bi-gram Search

The words -from GloVe vocabulary created as a side effect of gloVe training- can be searched via this part. The bi-gram search is also available. If the word/bi-gram is chosen, the document names are listed in the model that contains that word/bi-gram. If the document in that model is chosen, the short parts that contain chosen word/bi-gram in chosen document is listed in another modal.

5.3.5 Word Frequency Graph

The words -from GloVe vocabulary created as a side effect of gloVe training- are shown in this graph in the descending order of their frequencies as at most 100 pages. If the word is chosen, the papers that contain that word is listed in the model. If any document in that modal is chosen, then the 10 most close document according to cosine similarity is listed in another modal.

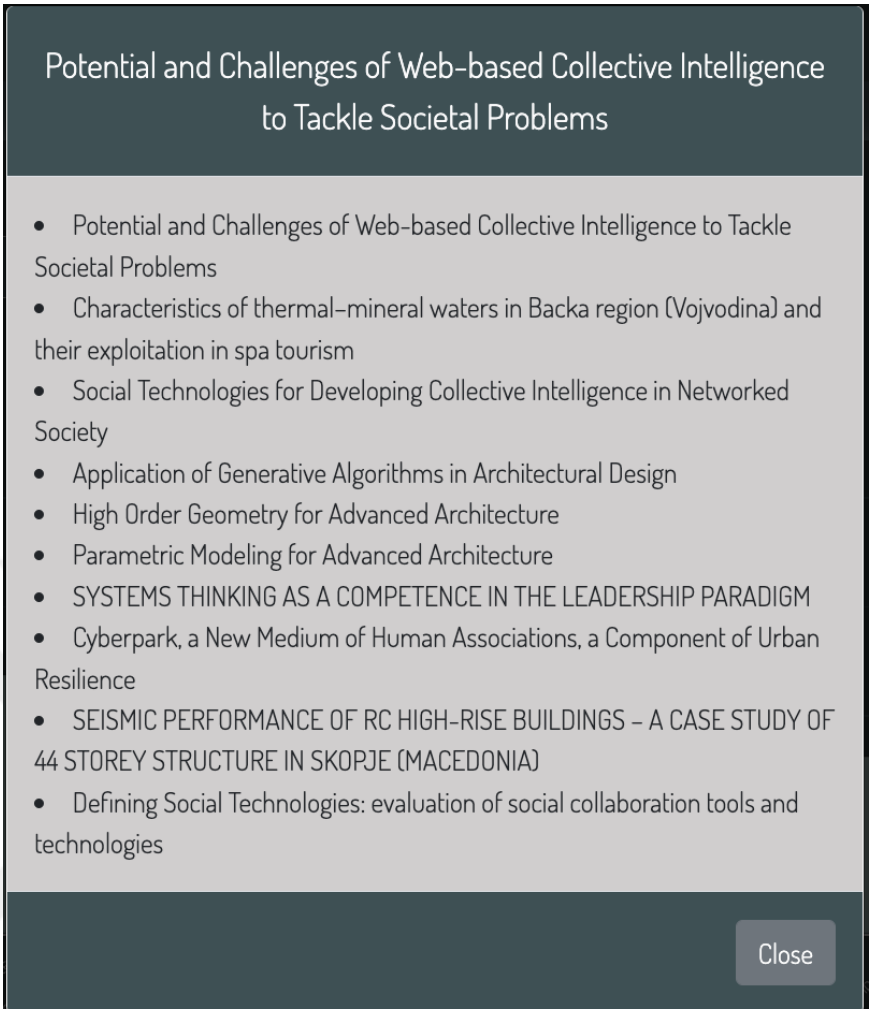

Figure 5.3: Top Ten Related Papers to the Selected Paper

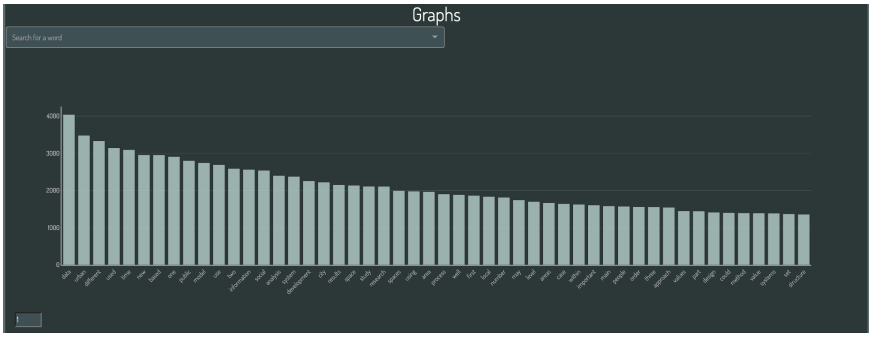

Figure 5.4: Word Frequency Graph

CHAPTER 6

EXPERIMENTS AND RESULTS

By using the Clustering Analysis Tool, documents are grouped into different clusters as optimally as possible via different trials. The results of these trials and their implications are discussed in this section. The discussion follows the order that the comparison of different groups of strategies, global comparison of the results, the reliability of metrics and the evaluation of optimization strategy.

Results can be grouped by embedding methods, and the combining strategies, which are used to combine embeddings of related text chunks since for a given embedding method and combining strategy, one trial optimization process works. So, the results are grouped according to the embedding combining strategy and embedding method. These constitute the first section. Secondly, the clustering methods are also optimized according to their parameters. The results of them are evaluated in the second section. In the third section, clustering metrics are evaluated by using their correlation in terms of best results chosen. Lastly, the hyper-parameter optimization strategy is evaluated in order to see how successful it works.

6.1 Comparisons by Combination Strategy

In this section, the comparisons are done with different clustering methods: Agglomerative Clustering, K-Means, DBSCAN, respectively. For each clustering method, different clustering metrics are used for comparison: Silhouette Score, and Rand Index, respectively. This order is preserved and the best scoring results are compared throughout the section. In addition, t-SNE algorithm is utilized in order to show multi-dimensional data points on 2D scatter plot.

According to the rand index values of clustering results of utilized embedding methods and combination strategies, the results are grouped into three categories: Averaging (sentence or word averages) results with better rand indices, averaging results with rather lower rand indices and the ones using tf-idf weighted word averages. This grouping is done according to the last evaluation but the sections include the evaluations done in the scope of the thesis. While the silhouette score and rand index are utilized through evaluations over the course of three sections mentioned, one section is for the Davies-Bouldin Index, Calinski-Harabasz Index and Homogeneity Score. In that section, the results of best embedding method will be used. In the last section, the results of all trials instead of only the best result for the best embedding method and combination strategy are utilized in order to understand the role of optimization strategy.

6.1.1 Averaging Combination Strategy (avg, sentence-avg)

This section includes the methods that have good rand index scores and mostly uses averaging method as mentioned (only Doc2Vec is different that is calculated through). These are Fasttext, GloVe, Word2Vec, Sentence BERT, BERT with scibert data set and Doc2Vec.

Table 6.1: Best Silhouette Score Results Table with Averaging Combination Strategy, Agglomerative Clustering

Method	**Option**	**5 Clusters**	**10 Clusters**	**15 Clusters**	**20 Clusters**
word2vec	avg	0.3728	0.2337	0.2716	0.2370
glove	avg	0.2806	0.2148	0.1357	0.0909
glove	pre-prepared	0.3490	0.1812	0.1189	0.0957
fasttext	**avg**	**0.5392**	**0.5137**	**0.5021**	**0.5210**
scibert	sentence-avg	0.2551	0.1326	0.1045	0.0859
sentence-bert	sentence-avg	0.2129	0.1410	0.0983	0.0682
doc2vec	doc-embedding	0.4548	0.4002	0.3724	0.3394

This method is used with the word-embedding techniques: Word2Vec, Fasttext and

GloVe. These techniques create word vectors after training on all documents and the average of these word vectors is evaluated for each document.

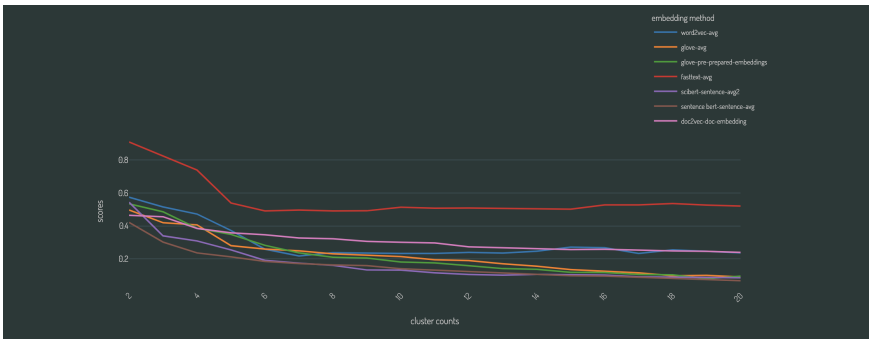

Figure 6.1: Best Silhouette Score Results Graph with Averaging Combination Strategy and Agglomerative Clustering Method

As shown in Table 6.1 and Figure 6.1, the result are not very close to the 1, which is maximum silhouette score. These low results, when it is thought that these are the best silhouette scores of trials for each embedding method, show that the data is not separable enough to see good separations. This can be the result of the fact that all documents are scientific papers and from the same conference even if the collection is a multi-disciplinary collection.

One may assume that the local training with Fasttext GloVe, Doc2Vec, Word2Vec cannot capture the word similarities enough because they are only trained with the papers given. This is not a true statement since the GloVe pre-trained embeddings, which was trained with Wikipedia data, the Sentence-BERT which was trained with different types of data sets and the given data set, SciBert model which was trained with scientific papers and the given data set requires to give good silhouette scores. However, their scores are generally lower than the methods that are trained only with given data set. So, different embedding methods cannot overcome the problem of not very distinguishable data.

Even, the pre-trained models assume less-separability. This supports the claim that the data may not be very distinguishable in essence because all documents are scientific papers and from the same conference because the the more global training makes things worse in terms of silhouette score. We can see this situation when we compare

the Fasttext, Word2Vec with the GloVe, where Fasttext and Word2Vec favors the local positions of words and the GloVe favors local positions and the global co-occurence at the same time. Thus, one can assume good separation of data with methods favoring local features, and slightly bad separation of data with methods favoring global features. But, this does not imply that good separation means a good clustering since data actually can be non-separable as well.

Even these are correct, data can be assumed to be distinguished to some extent, as well since the silhouette score is not very low but medium for some embedding methods.

In addition, before cluster count is 5, there are relatively high scores but it starts to be even smaller than 0.5 for all embedding method as the clustering count increases. This can be the result of outliers which cause silhouette scores to be higher at first.

Table 6.2: Best Silhouette Score Results Table with Averaging Combination Strategy, K-Means Clustering

Method	**Option**	**5 Clusters**	**10 Clusters**	**15 Clusters**	**20 Clusters**
word2vec	avg	0.2952	0.3002	0.2763	0.2797
glove	avg	0.1995	0.1806	0.1696	0.1667
glove	pre-prepared	0.1363	0.1077	0.1056	0.1012
fasttext	**avg**	**0.5324**	**0.5513**	**0.5407**	**0.5350**
scibert	sentence-avg	0.1139	0.0838	0.0826	0.0775
sentence-bert	sentence-avg	0.1047	0.0967	0.0988	0.0955
doc2vec	doc-embedding	0.2464	0.2647	0.2989	0.3152

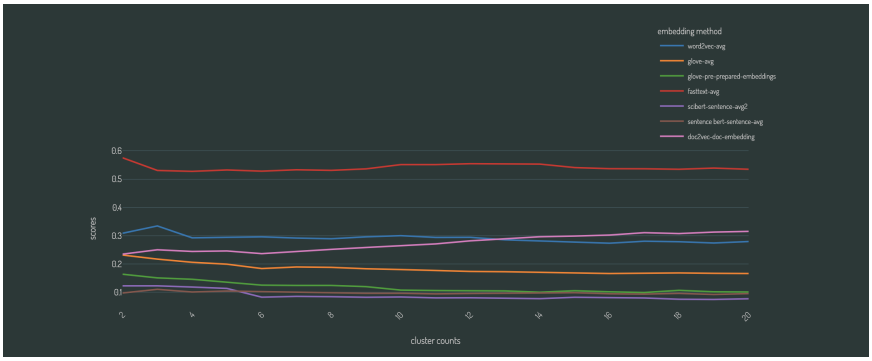

Figure 6.2: Best Silhouette Score Results Graph with Averaging Combination Strategy and K-Means Clustering Method

As shown in Table 6.2 and Figure 6.2, K-Means Clustering gives smoother scores over cluster counts than Agglomerative Clustering. If there was outliers as we expected, these seem to be separated to more crowded clusters for relatively smaller cluster counts. Then, this situation continues for more cluster counts as well since the silhouette scores do not change much over cluster counts. Except the Doc2Vec method, K-Means results seem to have similar average silhouette score over cluster counts to Agglomerative results. This means that the data set was clustered more smoothly with K-Means but probable not very distinguishable data problem would continue with K-Means since the scores are still very low. The other claims with Agglomerative Clustering can be repeated with K-Means as well. The fact that Doc2Vec silhouette scores are different from other methods in which the scores decrease should be analysed in scatter plots.

Table 6.3: Best Silhouette Score Results Table with Averaging Combination Strategy, DBSCAN Clustering

Method	**Option**	**Score**	**Generated Cluster Count**
word2vec	avg	-	1
glove	avg	0.5472	3
glove	pre-prepared	0.4528	2
fasttext	**avg**	**0.9100**	**2**
scibert	sentence-avg	-	1
sentence-bert	sentence-avg	-	1
doc2vec	doc-embedding	-	1

DBSCAN clustering gives the silhouette scores in Table 6.3. Mostly, small number of clusters are predicted, where most of them is 1 cluster. The ones other than 1 can make different and hopefully clustering than K-Means and Agglomerative Clustering.

Now, the scatter plots of the documents which are resulting in the best silhouette scores for each embedding method and combination strategy will be analysed. Data visualisation can give us more information about our claims.

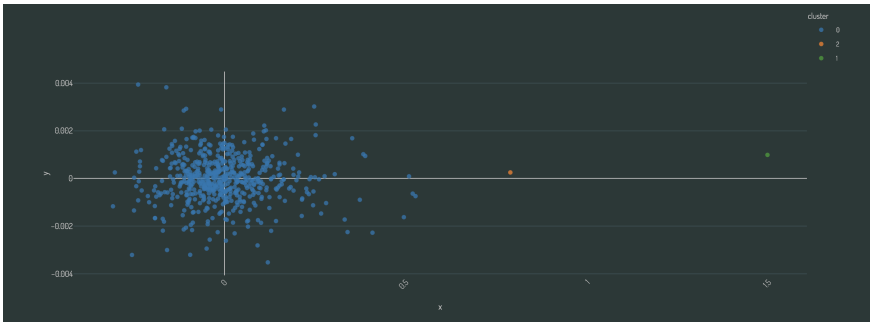

Figure 6.3: PCA Scatter Plot of 3 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method

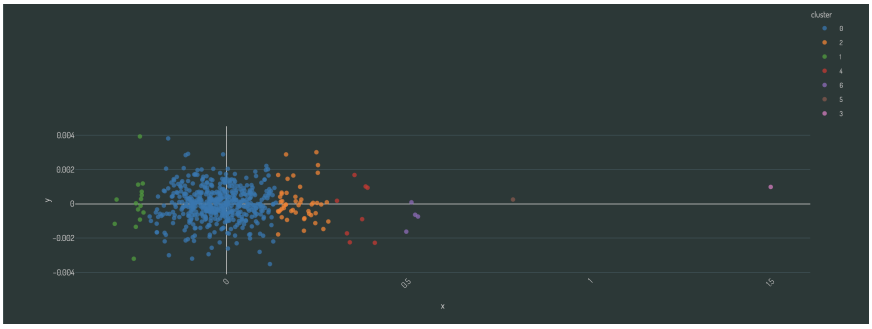

Figure 6.4: PCA Scatter Plot of 7 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method

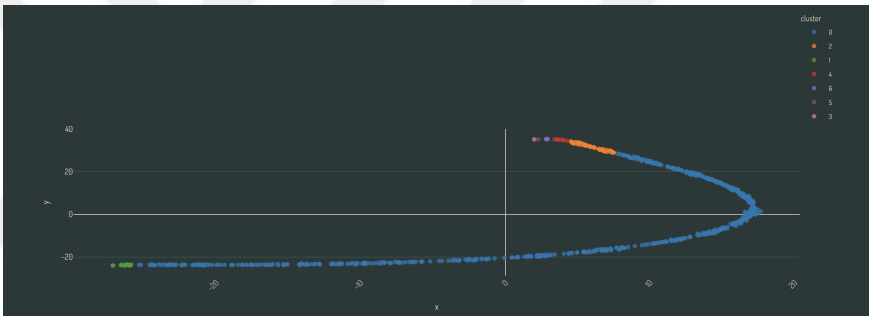

Figure 6.5: TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 40

Figure 6.3 shows that Fasttext has outliers and silhouette scores are sensitive to outliers, so the Fasttext give better silhouette scores for 2 and 3 clusters and that is not stable. According to Figure 6.1, the silhouette scores for Fasttext becomes steady after 7 clusters. The decrease until that point is because of the small clusters that are well separated as we can see from Figure 6.4. Figure 6.5 shows us that there is a linearity in the data and even if the silhouette score is high, the general data with Fasttext seems not separable enough. Thus, Fasttext with best silhouette score does not give good, equally separated clusters but only outliers and small clusters separated.

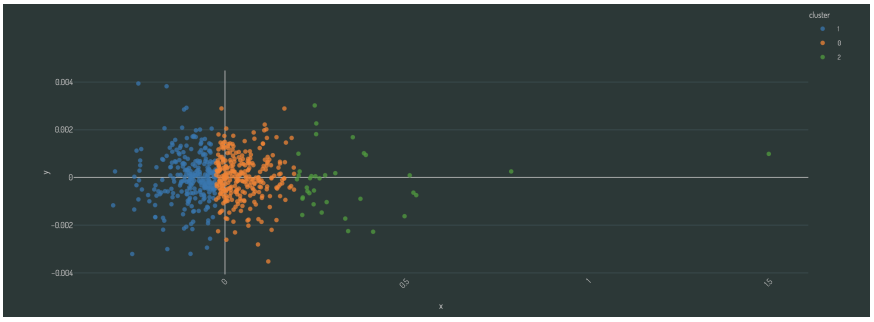

Figure 6.6: PCA Scatter Plot of 3 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method

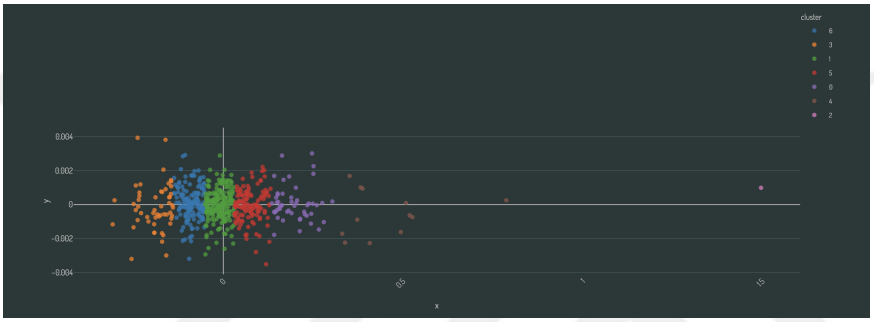

Figure 6.7: PCA Scatter Plot of 7 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method

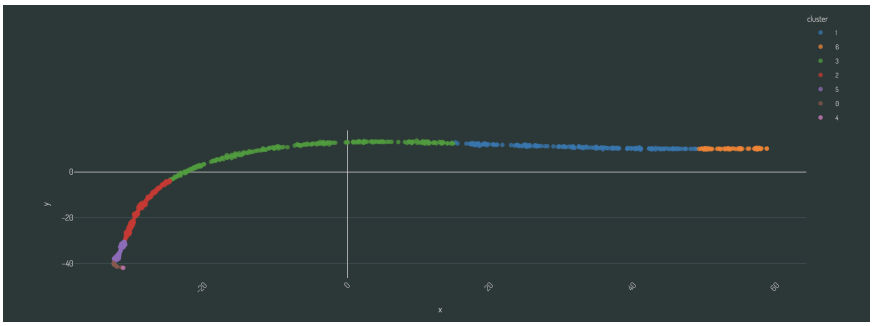

Figure 6.8: TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 30

Figure 6.6 is similar to Figure 6.3 except outliers are put in a more crowded second cluster. Also, the unsplit cluster in Agglomerative Method is split into two but not with distant clusters. Also, Figure 6.7 shows that the central 3 clusters do not

need to be distinguished. So, the clusters are more equally distributed but not very distinguishable clusters are created. As a result, Agglomerative Clustering creates more distinct but small clusters and one big cluster which is not effective in terms of distinguishing papers and the K-Means creates more crowded clusters but they are not separated enough. In addition, the Figure 6.8 proves the inseparability of points created by Fasttext method as in Agglomerative Clustering via TSNE method.

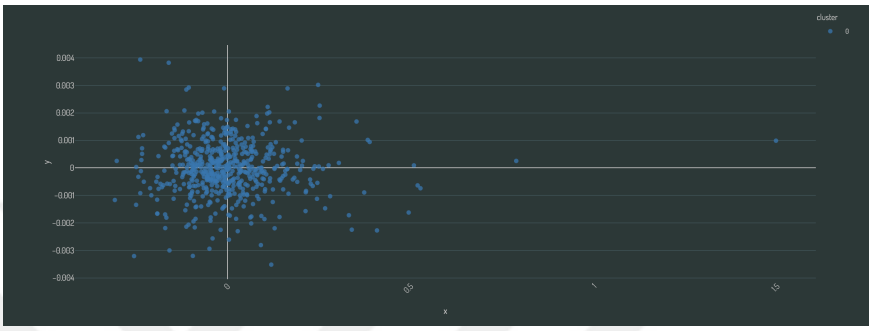

Figure 6.9: TSNE Scatter Plot of 1 cluster With Fasttext Averaging Having Best Silhouette Score Strategy and DBSCAN Clustering Method, perplexity: 30

DBSCAN expects one cluster for the Fasttext, so DBSCAN cannot distinguish data. When we visualize it via PCA, we see that data points are not distinguishable enough in Figure 6.9, as well.

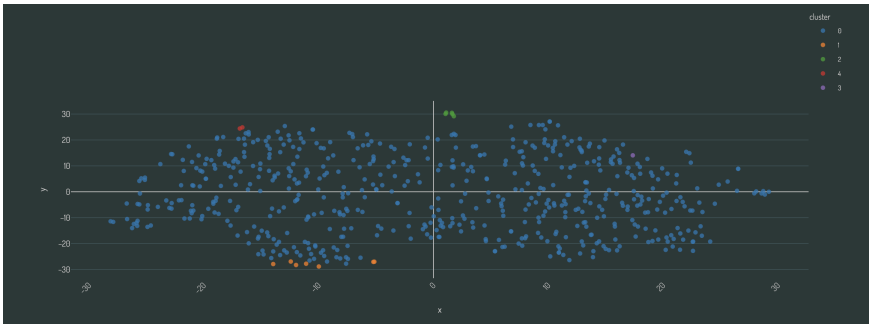

Figure 6.10: TSNE Scatter Plot of 5 clusters With GloVe Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 50

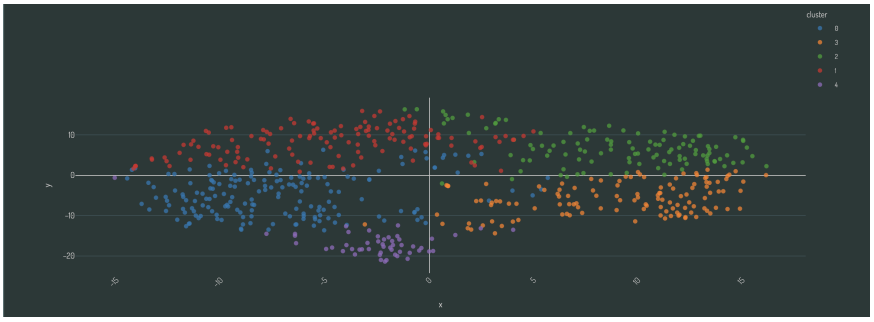

Figure 6.11: TSNE Scatter Plot of 5 clusters With GloVe Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 50

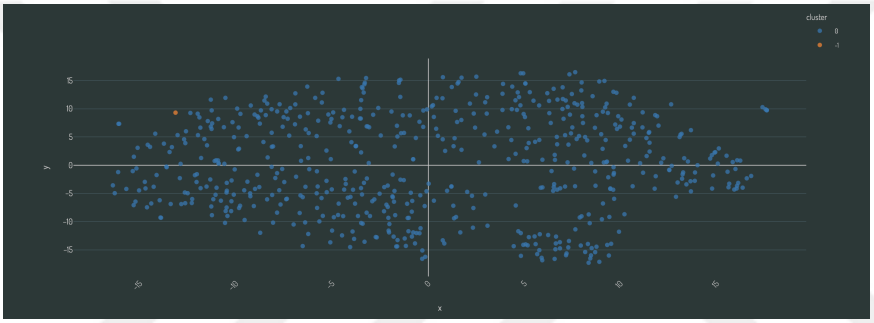

Figure 6.12: TSNE Scatter Plot of 2 clusters With GloVe Averaging Having Best Silhouette Score Strategy and DBSCAN Clustering Method, perplexity: 50

In this GloVe method, which is trained only with local data, there is a noisy data distribution for all clustering methods. 5 clusters are chosen because there is a dramatic change until 5 clusters in Agglomerative Clustering and the scores go steady for K-Means Clustering. Agglomerative Clustering for 5 clusters do not separate the noisy data. On the contrary, only some outlier points are separated over more cluster counts. This can be the reason that the silhouette score diminishes over cluster counts for GloVe. However, K-Means seems to give a better clustering visualisation result. This makes the silhouette score over cluster counts to be stable. Actually, this works for all methods. Still, the data is noisy and the dense clusters are not formed, so the silhouette scores are low. DBSCAN predicts 2 clusters and only one point is distinguished from the main cluster. This implies that there can be outliers.

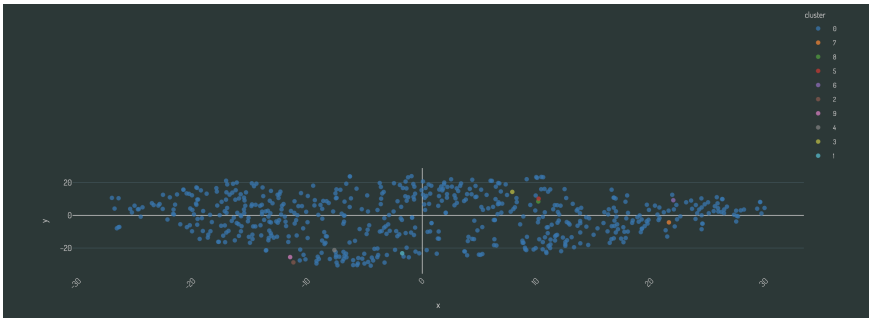

Figure 6.13: TSNE Scatter Plot of 5 clusters With GloVe Pretrained Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 50

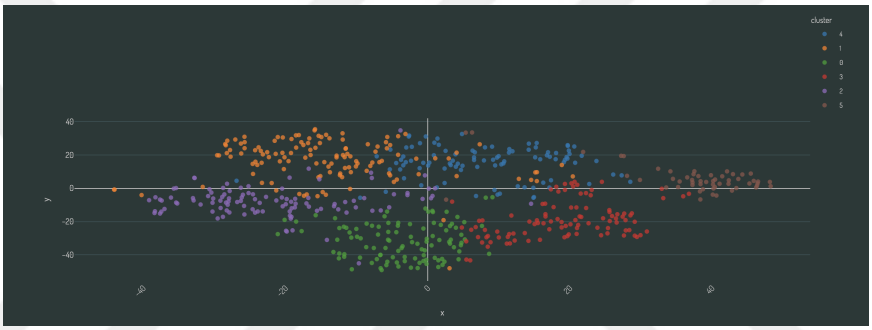

Figure 6.14: TSNE Scatter Plot of 5 clusters With GloVe Pretrained Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 20

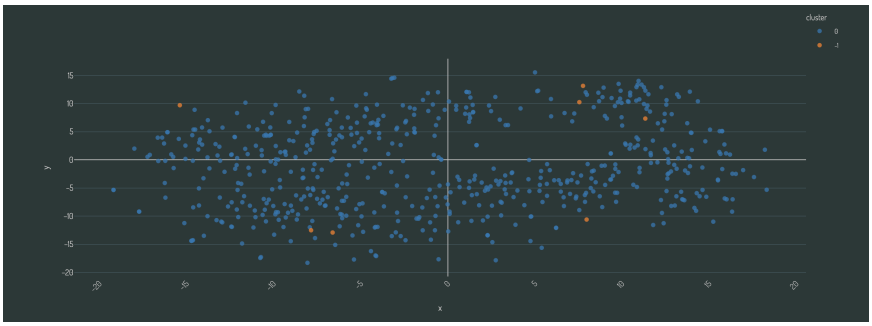

Figure 6.15: TSNE Scatter Plot of 2 clusters With GloVe Pretrained Averaging Having Best Silhouette Score Strategy and DBSCAN Clustering Method, perplexity: 50

In this GloVe model, only the pre-trained embeddings from Wikipedia data set is used. The noisiness is similar to the locally trained GloVe results. Outliers in Agglomerative Clustering is far more than any other embedding method. Even if 10 is chosen as cluster count, the clusters with very few data points are formed except the main cluster. This changes with the K-Means algorithm but the noisiness still applies. The clusters are crowded but they are not well separated enough. It is again similar to the locally trained GloVe results. DBSCAN also implies the noisiness of the data.

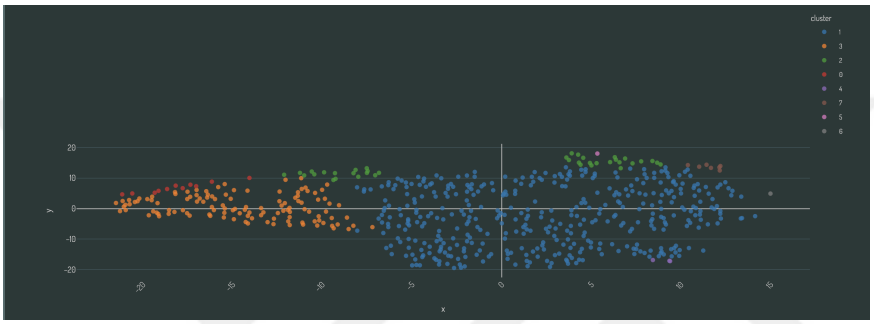

Figure 6.16: TSNE Scatter Plot of 8 clusters With Word2Vec Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 50

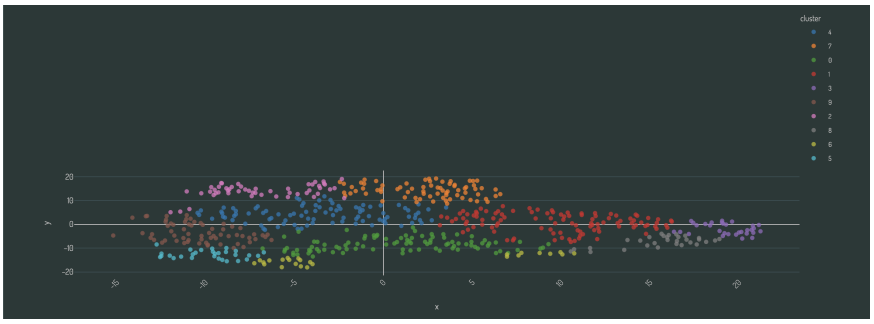

Figure 6.17: TSNE Scatter Plot of 10 clusters With Word2Vec Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 50

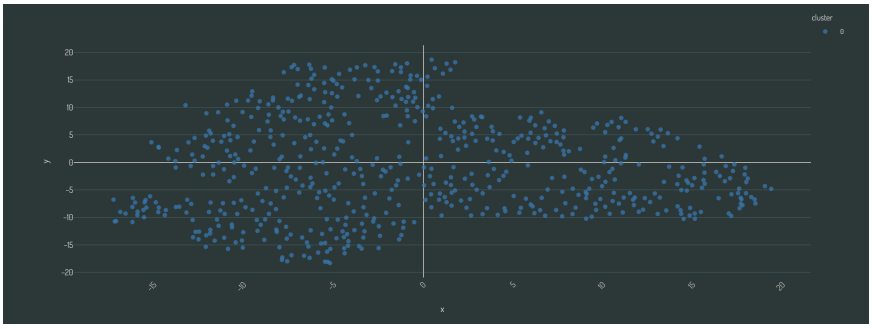

Figure 6.18: TSNE Scatter Plot of 1 cluster With Word2Vec Averaging Having Best Silhouette Score Strategy and DBSCAN Clustering Method, perplexity: 50

For the Agglomerative Clustering method, 8 clusters are chosen since the silhouette scores decrease until 6 clusters, and peak at 8 clusters. The same applies for the K-Means clustering, in which decrease goes until 8 clusters and 10 clusters peak. According to that choice, Agglomerative Clustering creates outliers and crowded clusters but the data points are distributed noisy. For outliers in the plot, Word2Vec is better than Fasttext and GloVe both. But the data points show noisiness like in the GloVe method. The results from K-Means clustering method have less outliers as always. Whereas the noisiness is kept and this decreases the silhouette score, it is better than GloVe results. So, a better clustering visual can be seen from the scatter plot. DBSCAN actually proves the noisiness one more time.

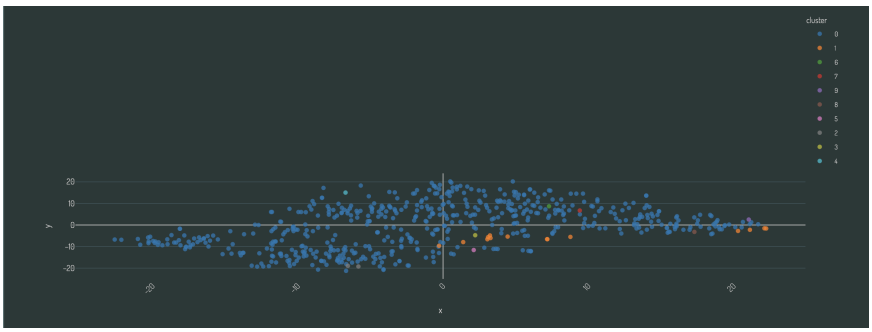

Figure 6.19: TSNE Scatter Plot of 10 clusters With SciBERT Sentence Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 50

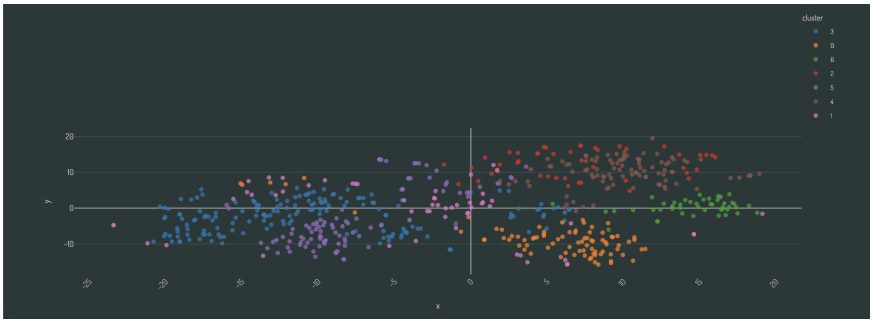

Figure 6.20: TSNE Scatter Plot of 7 clusters With SciBERT Sentence Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 50

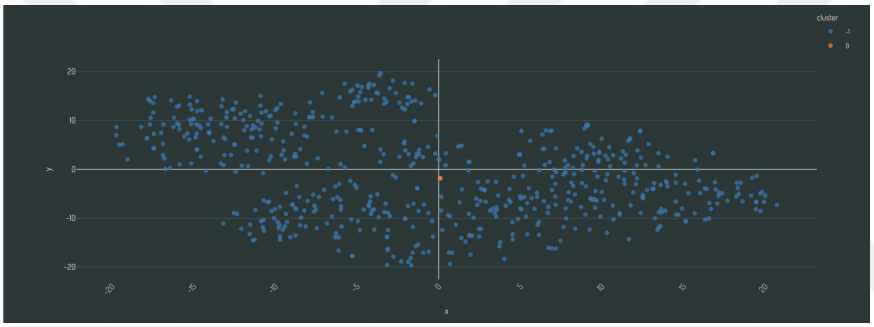

Figure 6.21: TSNE Scatter Plot of 2 clusters With SciBERT Sentence Averaging Having Best Silhouette Score Strategy and DBSCAN Clustering Method, perplexity: 50

Silhouette scores for the SciBERT over the cluster counts are lower according to other methods for Agglomerative Clustering, so, it gives a noisy points at hand. The silhouette scores decrease over time for Agglomerative Clustering, so there are outliers that can be expected since the silhouette scores seem bigger for smaller cluster counts. With K-Means Algorithm, noisiness still continues but rather equally distributed clusters are formed as this is the case for K-Means algorithm. However, the clusters are mostly intertwined. This seems similar to GloVe but the GloVe has a more separated distribution in terms of K-Means algorithm. DBSCAN expects two clusters but as one outlier point and one main cluster, which is too distributed.

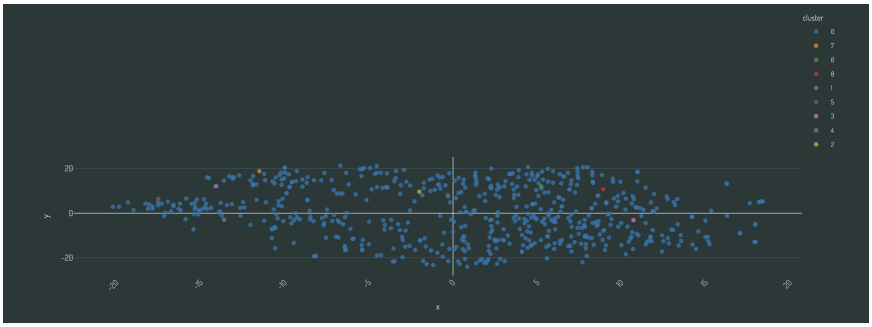

Figure 6.22: TSNE Scatter Plot of 9 clusters With Sentence BERT Sentence Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 50

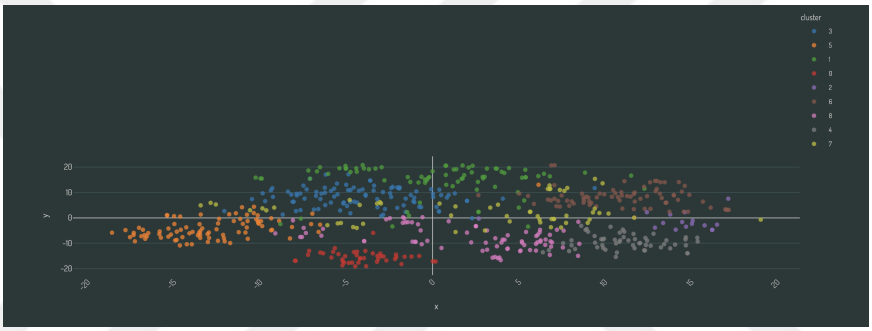

Figure 6.23: TSNE Scatter Plot of 9 cluster With Sentence BERT Sentence Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 50

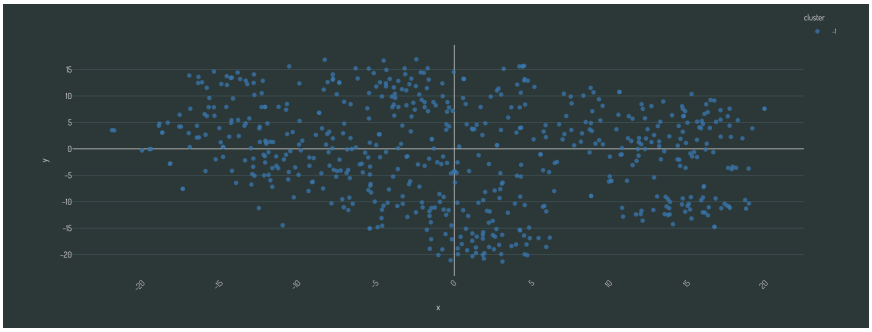

Figure 6.24: TSNE Scatter Plot of 1 cluster With Sentence BERT Sentence Averaging Having Best Silhouette Score Strategy and DBSCAN Clustering Method, perplexity: 50

9 clusters are chosen because it has a slight increase in Agglomerative Clustering after a decrease at the silhouette score-cluster count graph. Scatter plot with Agglomerative Clustering 8 outlier clusters and 1 main cluster. This explains the decrease of silhouette score. With K-Means, these outliers do not affect much the clusters as in Agglomerative Clustering. This is the case for all the analysis in this part. However, Sentence BERT has low silhouette score, so there are intertwisted clusters even more than SciBERT method. DBSCAN results are also similar to the ones in SciBERT. This can be explained as SciBERT is using BERT architecture and the Sentence BERT is only extension to BERT.

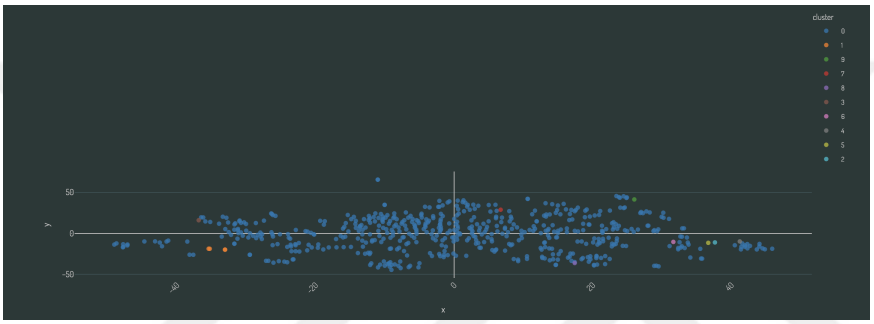

Figure 6.25: TSNE Scatter Plot of 10 clusters With Doc2Vec Document Embedding Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 20

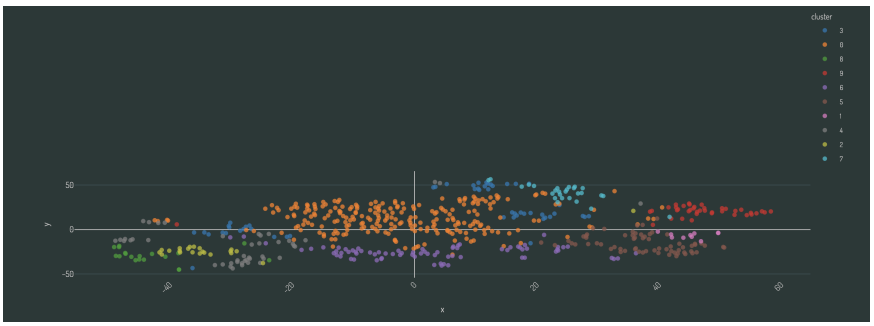

Figure 6.26: TSNE Scatter Plot of 9 cluster With Doc2Vec Document Embedding Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 40

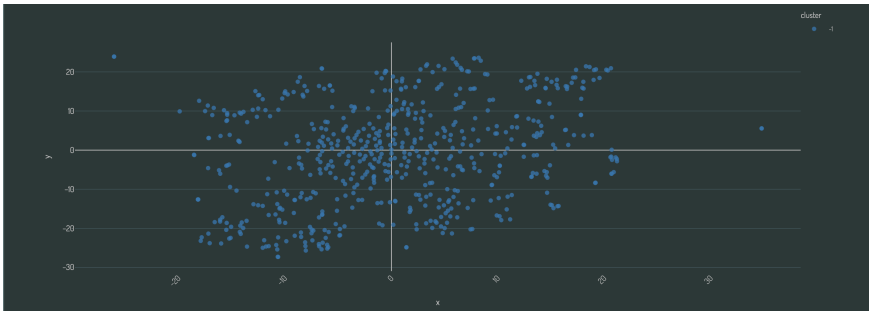

Figure 6.27: TSNE Scatter Plot of 1 cluster With Doc2Vec Document Embedding Having Best Silhouette Score Strategy and DBSCAN Clustering Method, perplexity: 50

Agglomerative Clustering gives similar results as if it has outliers and the silhouette score decreases over cluster count. However, there seems to be some clusters that can be formed if there is not so much outliers. K-Means solves this problem and a slightly well clustered results are demonstrated. Actually, Doc2Vec has the second best silhouette scores for K-Means. There are a few intertwined clusters and very few outliers. DBSCAN results seem to be noisy as always but some groups can be seen.

Comparing them, the SciBERT and Sentence BERT has the lowest silhouette scores. As expected, their Agglomerative Clustering results are noisy and has many outliers. Between them, SciBERT shows slightly better performance. Fasttext has the best silhouette scores but the TSNE clustering visualisation shows that Fasttext sees the data as linear. With PCA, there are good clusters at the edges horizontally but, the main cluster is not differentiable or distinguishable enough. DBSCAN supports the idea that the Fasttext has: data is not separable. Instead of Fasttext, Doc2Vec gives better clustering results, even if it is the second best embedding method in terms of silhouette score with K-Means. The data distribution is between noisy and tight. GloVe methods (averaging and pre-trained embeddings) have similar behaviour. They seem to be noisier than Fasttext but they are the second method that forms good clusters with K-Means. Word2Vec also has the best clusters. They are well-separated, not noisy as GloVe. Agglomerative Clustering did not give us good cluster visualisations in general but the Word2Vec works even better than the others with this methods as well. Actually, Word2Vec and Doc2Vec applies similar idea, so their idea beats other

methods in terms of best silhouette scored results in document embedding.

This concludes the part with Silhouette Scores. From this point on, only the Rand Index can help to identify better clustering results. Even if the results are optimized according to the silhouette score, the score is sensitive to outliers, so there can be better clusters which are not detected with best silhouette score. In order to detect that, the Rand Index is utilized. As mentioned in previous chapters, the labeled document names are uploaded to the testing tool. So, we can evaluate the Rand Index over only the labeled documents. Thus, we can detect better clustering results according to the Rand Index and these results will be treated as the final measures of clustering quality of different embedding methods and their corresponding combination strategy. Also note that, for all the graphs 7 clusters are used and DBSCAN is not applied since the labeled cluster count is only 7.

Table 6.4: Best Rand Index Results Table with Averaging Combination Strategy, Agglomerative Clustering

Method	**Option**	**Score**
word2vec	avg	0.5417
glove	avg	0.5824
glove	pre-prepared	0.5066
fasttext	**avg**	**0.6118**
scibert	sentence-avg	0.5641
sentence-bert	sentence-avg	0.5616
doc2vec	doc-embedding	0.3187

Table 6.5: Best Rand Index Results Table with Averaging Combination Strategy, K-Means Clustering

Method	**Option**	**Score**
word2vec	avg	0.5549
glove	avg	0.5682
glove	pre-prepared	0.4553
fasttext	avg	0.6272
scibert	sentence-avg	0.5997
sentence-bert	**sentence-avg**	**0.6287**
doc2vec	doc-embedding	0.3176

These results show that, the listed methods give mostly similar scores that are between and around 0.50-0.60 as Rand Index except the Doc2Vec. Their results show that the correct clusters of documents are predicted with such an accuracy. Actually, these results are not so close to the best results as the best result is 1 for Rand Index. However, there is a better performance than the silhouette scores. Silhouette score favors the cluster separation and rand index favors accuracy. So, even the data is not separated well, the prediction quality can be better for this data set. This indicates that the data is not very separable on its own.

Fasttext, GloVe, Word2Vec are the word-embedding methods. Among them, Fasttext and GloVe with averaging combination strategy are better than Word2Vec in terms of performance. Actually, these both methods are extensions to Word2Vec in order to detect some patterns about the words that are not detected by Word2Vec. So, this result is not a coincidence. Fasttext detects the semantically more complex and unknown words better and GloVe captures information from the global co-occurences of the words. Fasttext performs better than GloVe. This can be because of the scientific terms that are related to specific fields such as archaeology in the dataset and some words that are not even English. As an example to these words, there are saponification and reconstructionists, and rzeczpospoli ta (from Polish). Between two different GloVe methods, the locally trained one gives relatively better Rand Index, so it can be said that among methods utilized in the context of this thesis, the ones with

specifically trained gives better results. Also, the SciBERT is a BERT model trained on scientific data, so it has been got involved in the better clustering method group rather than the other sentence embedding, or contextual embedding methods such as RoBERTa, BERT with data set other than scientific papers and GPT-2, whose results are shown in next chapters. Sentence BERT is a method that uses contextual embeddings but it has a different strategy. Actually, it is an extension to BERT in order to overcome the low performance of BERT and RoBERTa for creating sentence embeddings. It seems successful for this dataset in our experiments as the the embedding method that has the best result for the K-Means algorithm is Sentence-BERT. Also, in terms of Agglomerative Clustering Results, it comes just after Fasttext and GloVe. Doc2Vec is a method that is using the context vector to grasp the contextual information of the document. It seems the least successful of successful methods in terms of Rand Index. It has given a better distributed visualisation than other methods with silhouette score but this does not work for the rand index.

Table 6.4 and Table 6.5 gives the idea of performances about the embedding methods over the given data set. With the scatter plot, we can have a better understanding about the data set. The embedding methods with having higher and lower rand indices will give different distributions, so the ones with higher rand indices are closer to the perfect distribution of data points. According to this fact, one can have assumptions about the data. Also, we can evaluate the data distributions according to methods and compare the scatter plots of the best rand index values with the ones with best silhouette score values.

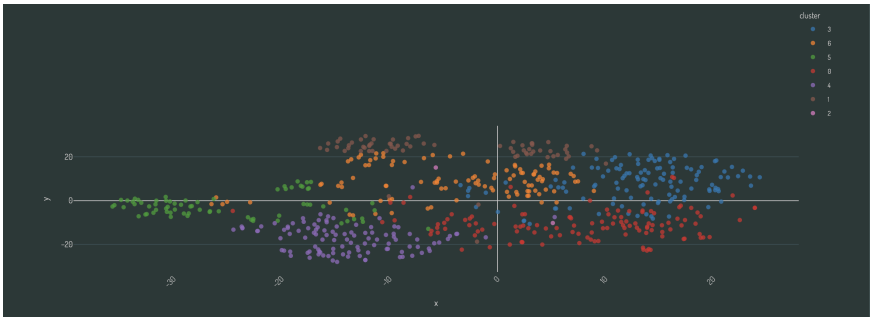

Figure 6.28: TSNE Scatter Plot of 7 clusters With Word2Vec Averaging
Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30

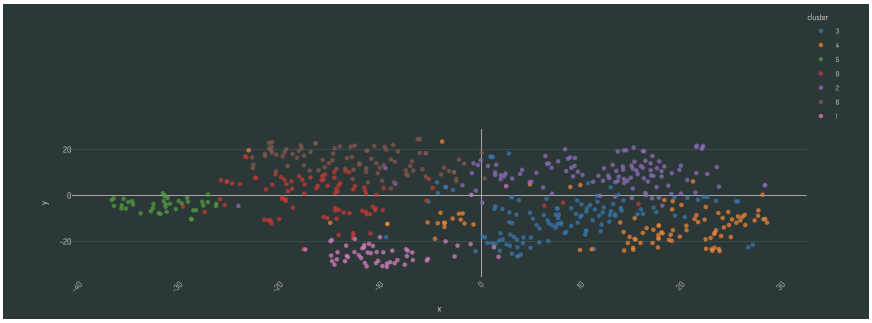

Figure 6.29: TSNE Scatter Plot of 7 clusters With Word2Vec Averaging Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30

For Word2Vec embedding clustering results with the best rand index, K-Means and Agglomerative Clustering results give similar rand indices and similar distributions except that the cluster positions change. There are uncertain points that seem closer to one cluster but belong to another cluster at the in the boundaries of the clusters. Actually data set has papers that are at the intersection of different subtopics, so this is not a coincidence. Also, the clusters are not dense. This case is better seen from Agglomerative Clustering result. K-Means has a better clustered and less intertwined clusters but clusters are still not dense. When we compare it to the ones with best silhouette scores for the Word2Vec embedding method (Figure 6.16 and Figure 6.17), it can be seen that the noise decreased and the density of clusters increased. Thus, Word2Vec gives a rather good cluster result and better visualisation than results with best silhouette scores for the Word2Vec embedding method.

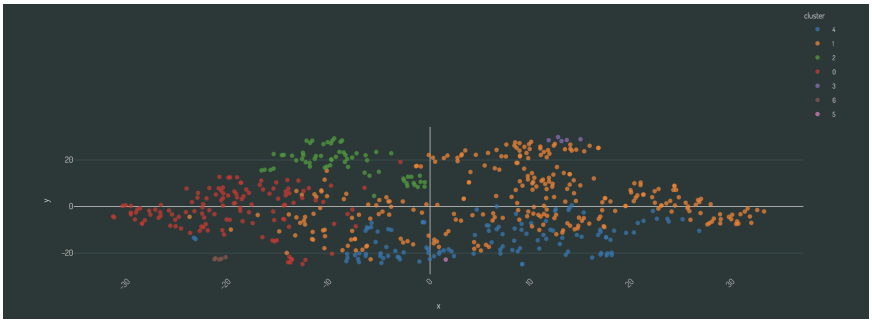

Figure 6.30: TSNE Scatter Plot of 7 clusters With GloVe Averaging Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30

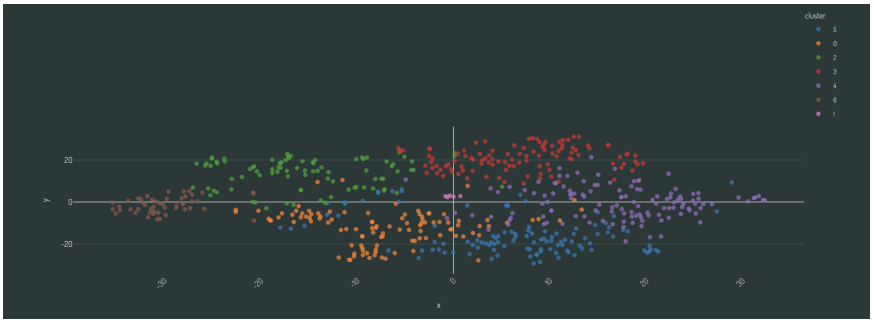

Figure 6.31: TSNE Scatter Plot of 7 clusters With GloVe Averaging
Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30

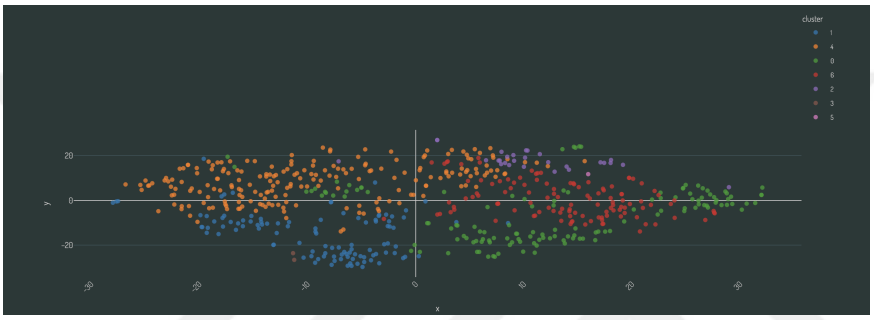

Figure 6.32: TSNE Scatter Plot of 7 clusters With GloVe Pre-trained
Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30

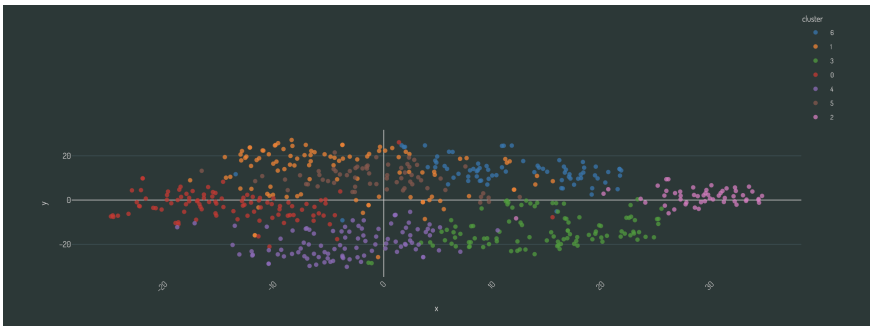

Figure 6.33: TSNE Scatter Plot of 7 clusters With GloVe Pre-trained
Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30

Clustering results of GloVe with two different combination strategy are visualised in Figure 6.30, 6.31, 6.32 and 6.33. Their results are noisier and the clusters are not

formed well even if the rand index value is relatively high. So, the GloVe is worse than Word2Vec in terms of getting rid of the noise and the non-dense clusters. Actually, results with the best silhouette score has a better clustering visualisation than these ones for K-Means algorithm (Figure 6.11, Figure 6.12). In terms of rand indices of two combination strategies, the averaging method has better rand index and better visualisation as well.

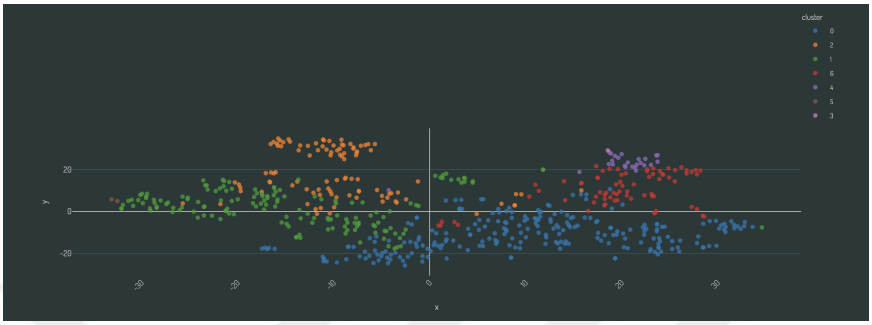

Figure 6.34: TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30

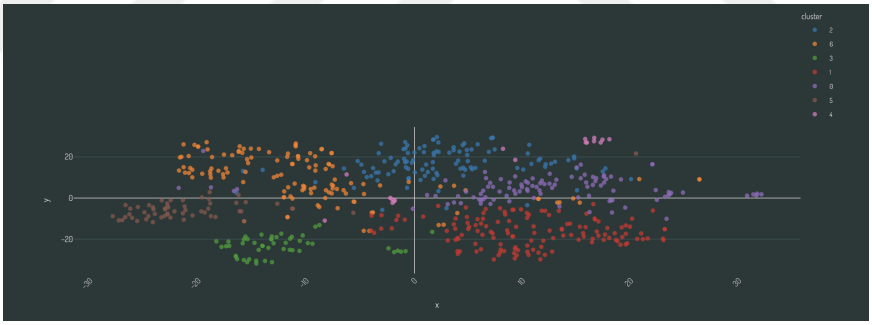

Figure 6.35: TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30

Fasttext gives the best Rand Index score for Agglomerative Clustering, so we can assume good results with that clustering method. However, the visualisation for Agglomerative clustering seems worse than Word2Vec. However, K-Means with a slightly less rand index has a better visualisation about the data. Actually, there are even some rather dense clusters like the pink and brown ones. Also, as in the GloVe and Word2Vec, the data is not distributed to clusters equally. This gives a clue about

the data set that some areas of interest have more papers than the others. The results with the best silhouette scores predicted the data points to have a linear form and be non-distinguishable enough. However, the ones with the best rand index separates the data to clusters better to some extent. Thus, the results of Fasttext gives good clustering score, visualisation, and can be one of the best results for this thesis.

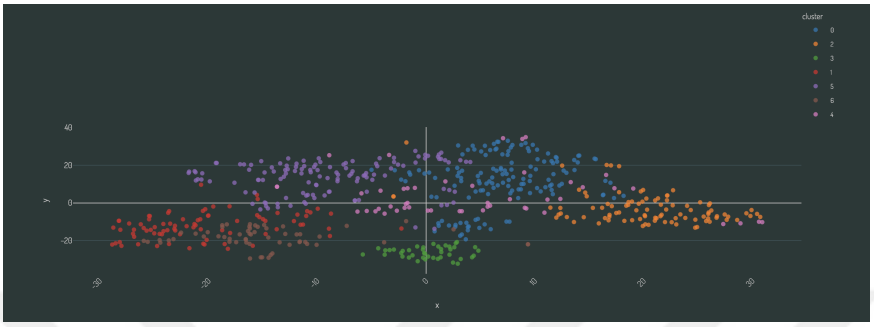

Figure 6.36: TSNE Scatter Plot of 7 clusters With SciBERT Sentence Averaging Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30

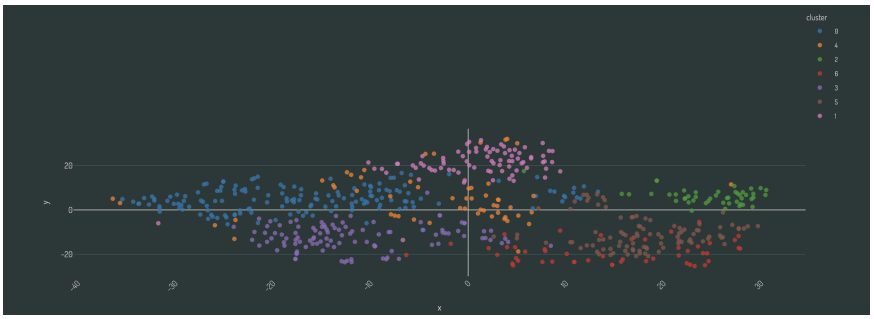

Figure 6.37: TSNE Scatter Plot of 7 clusters With SciBERT Sentence Averaging Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30

SciBERT has a good rand index with Agglomerative Clustering and better result with the K-Means clustering. Although the visualisation results are not very good where the clusters are intertwined so much, there are well separated clusters as well. So, this method does well in terms of score but visualisation is not as good as that.

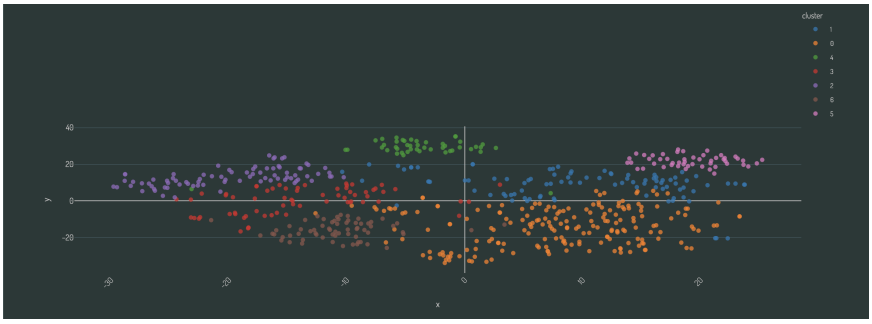

Figure 6.38: TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30

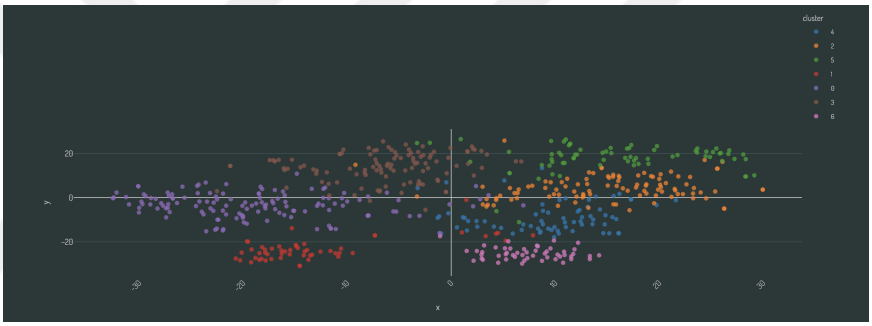

Figure 6.39: TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30

Sentence BERT is the best method in terms of the rand index with K-Means clustering. Actually, it is better than all results with Agglomerative Clustering, so Sentence BERT with using K-Means clustering is the best result among the embedding methods among their results with the best rand index. Sentence BERT with the best silhouette score has relatively good visualisation and the the method with best rand index is even better. There are very less unstable points in the boundaries clusters and clusters are relatively dense and better separated. Its performance in general seems even better than Fasttext in terms of score and visualisation.

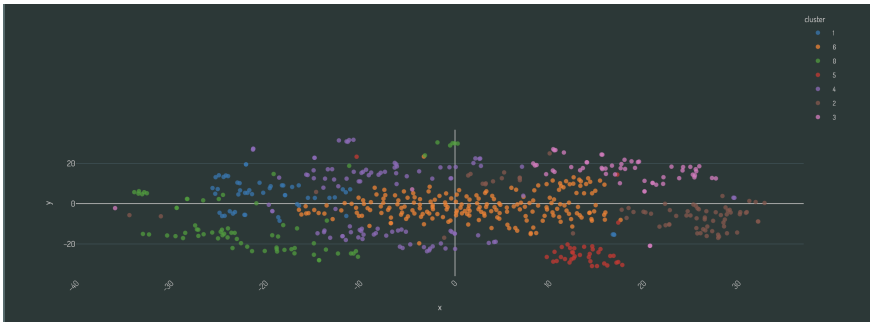

Figure 6.40: TSNE Scatter Plot of 7 clusters With Doc2Vec Document Embedding Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30

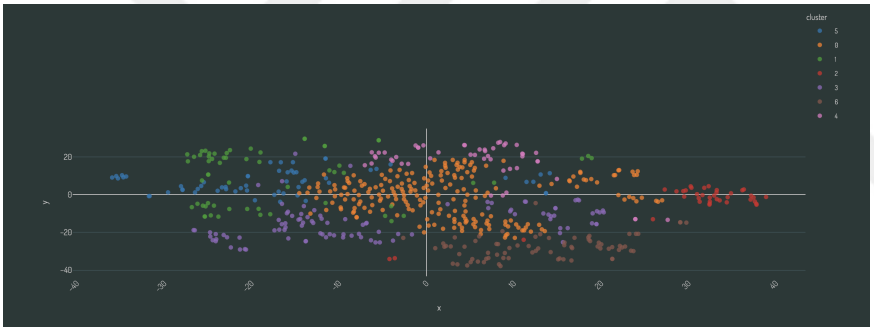

Figure 6.41: TSNE Scatter Plot of 7 clusters With Doc2Vec Document Embedding Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30

Doc2Vec has had a good performance with silhouette score but that does not do well with the rand index. This shows it does not have good accuracy for predicting correct clusters even if cluster separation is slightly good. Clusters seem well separated but the points in the same cluster are distributed too much. Doc2Vec results with best rand index has a similar performance with the ones with best silhouette score. So, one can conclude that the Doc2Vec did good for optimization output ,average silhouette score, but that effort did not increase the accuracy enough.

Table 6.6: Dimensions of Document Vectors of Embedding Methods with Best Rand Index Results

Method	Option	Clustering Method	Dimension
word2vec	avg	Agglomerative	145
word2vec	avg	K-Means	75
glove	avg	Agglomerative	90
glove	avg	K-Means	120
glove	pre-prepared	Agglomerative	200
glove	pre-prepared	K-Means	300
fasttext	avg	Agglomerative	155
fasttext	avg	K-Means	125
scibert	sentence-avg	Agglomerative	768
scibert	sentence-avg	K-Means	768
sentence-bert	sentence-avg	Agglomerative	768
sentence-bert	sentence-avg	K-Means	768
doc2vec	doc-embedding	Agglomerative	75
doc2vec	doc-embedding	K-Means	230

There is inconsistency between rand indices and their corresponding visualisations for some methods. This can be because of the dimensions of embedding vectors of documents for each embedding method. The reason is that the vectors are reduced to 2 clusters for visualisation and this causes some data loss and it can be expected that the higher the vector dimension higher the data loss and worse visualisation results. So, Table 6.6 can be helpful for this issue. Word2Vec with K-Means has better good clustering results than the GloVe and Fasttext even if it has a lower Rand Index. Word2Vec vectors have 75 scalars in one document, Glove has 120 and Fasttext has 125. So, Word2Vec can have had a better visualisation because the its vector dimension is lower according to the others. The same incosistency exists for SciBERT as well. Its rand index value is higher but its visualisation is not as good as its value. But this can be explained as the dimension 768 can have so much data loss while visualising data points.

This concludes this section. The methods using averaging as combination strategy and having good prediction results are analysed in this section. Among the classical word embedding methods, Fasttext has the best values and Sentence BERT has the best among contextual word embedding methods. Between them, Sentence BERT has a better performance for clustering results. Actually, one of the purposes of the Sentence-BERT is to fix the issue that the contextual embedding methods perform worse than GloVe [37].

During the section, the silhouette scores are evaluated and their results are analysed. In order to obtain more information about the clustering, scatter plots of the data points are drawn. It is seen that that using only silhouette scores as target is not sufficient and can be misleading when evaluating embedding methods and their corresponding combination strategies. For that, human labeling and Rand Index have given us better clustering results among trials. According to rand index results, data points of best trial for each method is visualised as well. Lastly, dimensions of document vectors are analysed so that one can understand why there is inconsistency between score values and visualisation results.

6.1.2 Term Frequency-Inverse Document Frequency (TF-IDF) Method as Combination Strategy

Term Frequency and Inverse Document Frequency method reflects the importance of a word inside a document. So, the tf-idf values of words can be utilized as their weights and weighted average can be calculated instead of the only average of words in order to create the document embedding from word embeddings. Word2Vec, Fasttext and GloVe creates word vectors as outputs. So, these methods are suitable to using weighted averages. So, this section contains the experimental results of tf-idf combination strategy.

Table 6.7: Best Silhouette Score Results Table with Term Frequency-Inverse Document Frequency Combination Strategy, Agglomerative Clustering

Method	Option	5 Clusters	10 Clusters	15 Clusters	20 Clusters
word2vec	tf-idf	0.7174	0.6799	0.6315	0.6307
glove	tf-idf	0.7319	0.6385	0.6428	0.6276
fasttext	tf-idf	0.7021	0.6814	0.6629	0.6568

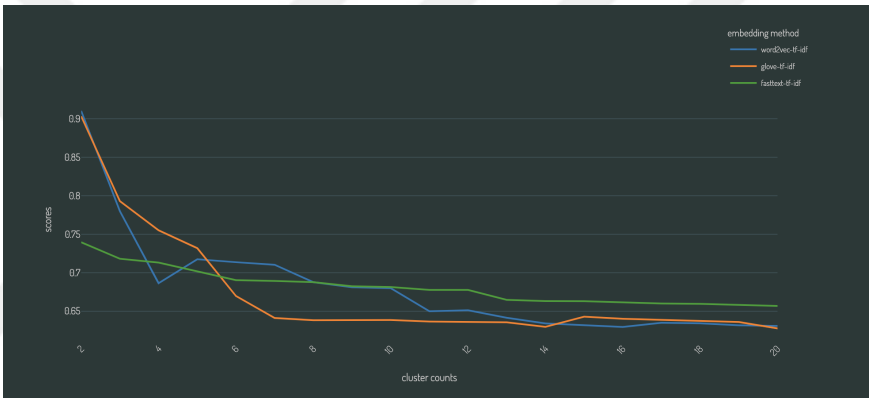

Figure 6.42: Best Silhouette Score Results Graph with Tf-Idf Combination Strategy and Agglomerative Clustering Method

Silhouette scores for the Agglomerative Clustering is shown at Table 6.7 and Figure 6.42. Silhouette scores are higher than the ones with averaging combination strategy, so we can expect a better separated data points whereas it is not still very close to the best silhouette index. This can be because that the dataset is not separable with known methods enough. Also, there is a dramatic decrease until 5-7 clusters in the line graph for each embedding method. After that, scores go steadier. So, we can assume that there are outliers till those points and more crowded clusters start to form after.

Table 6.8: Best Silhouette Score Results Table with Term Frequency-Inverse Document Frequency Combination Strategy, K-Means Clustering

Method	Option	5 Clusters	10 Clusters	15 Clusters	20 Clusters
word2vec	tf-idf	0.5757	0.5617	0.5407	0.5337
glove	tf-idf	0.6140	0.5290	0.5132	0.4851
fasttext	**tf-idf**	**0.7385**	**0.5885**	**0.5635**	**0.5629**

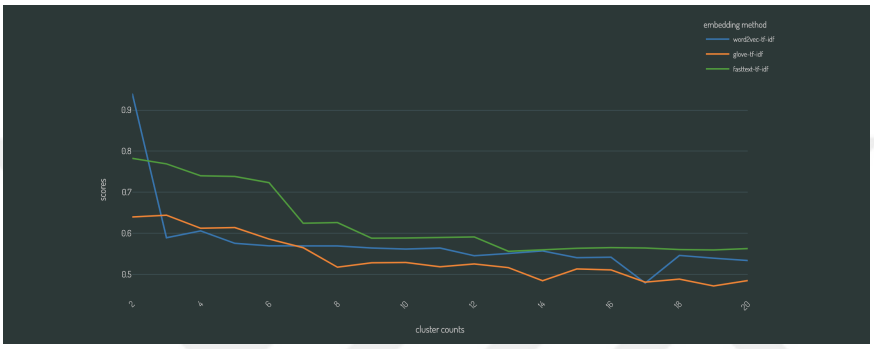

Figure 6.43: Best Silhouette Score Results Graph with Tf-Idf Combination Strategy and K-Means Clustering Method

Silhouette scores for the K-Means Clustering is shown at Table 6.8 and Figure 6.43. The scores are lower than the ones created by Agglomerative Clustering but still is better than the silhouette score results of methods with averaging combination strategy. So, a well separated data is expected for K-Means results as well. the Also, fluctuations for small number of cluster counts exist for silhouette scores by K-Means algorithm and the methods, and after that the score goes steady for different cluster counts. Although, K-Means is expected to give less outliers in terms of its nature, scatter graphs will show us whether there are outliers in K-Means as well.

In this section, 7 is chosen as the cluster count for all the scatter graphs. For Agglomerative clustering, silhouette score is about to decrease after 7 clusters for Word2Vec, Fasttext starts to be steady at this point and GloVe is steady until large number of clusters. So, 7 cluster silhouette score values can give scatter plots worth analysing. For the K-Means algorithm, GloVe and Word2Vec are almost equal and Fasttext is

just close in terms of silhouette score for 7 clusters. Also, the Rand Index needs to have 7 clusters as explained. Thus, scatter plots in this section is drawn to 7 clusters.

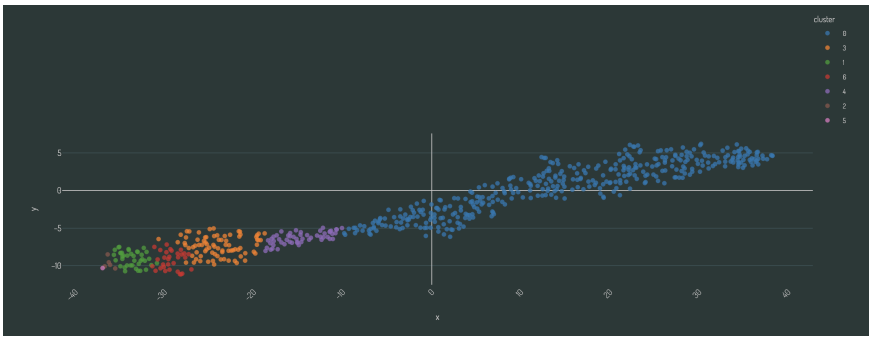

Figure 6.44: TSNE Scatter Plot of 7 clusters With Word2Vec Tf-Idf Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 30

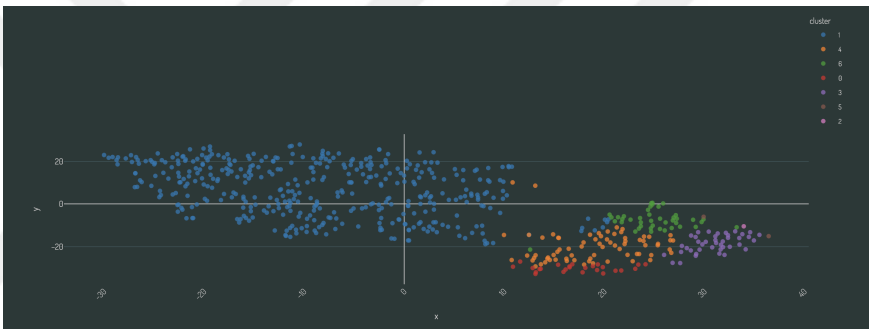

Figure 6.45: TSNE Scatter Plot of 7 clusters With Word2Vec Tf-Idf Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 30

The Agglomerative Clustering result (Figure 6.44) shows that Word2Vec has different plunge points and 7 is a pre-plunge point for Word2Vec. With these kinds of decreases over cluster counts, Word2Vec has the lowest silhouette score for big cluster counts, while it is the best for smaller cluster counts. So, we can expect some less crowded clusters to split as the process goes on. These smaller clusters and the ones that do not decrease silhouette scores can be seen. The brown and pink clusters are the example of former and the purple, red and yellow ones are the examples of the latter. The separation of clusters and its effect to silhouette score can be seen as if the clusters form from left to right. The formed clusters seem to be dense and less noisy but the data points are very close to each other and lined up with a small variation. Thus, the

data set is predicted to have less variation over the diagonal line and the separable by Word2Vec with tf-idf and Agglomerative Clustering.

The K-Means clustering result (Figure 6.45) shows that the there is one cluster which is noisy and non-separable till 7 clusters. Clusters are worse than the ones with Agglomerative Clustering visually and as a silhouette score as well. So, K-Means show worse performance than the Agglomerative Clustering for this method in terms of silhouette score.

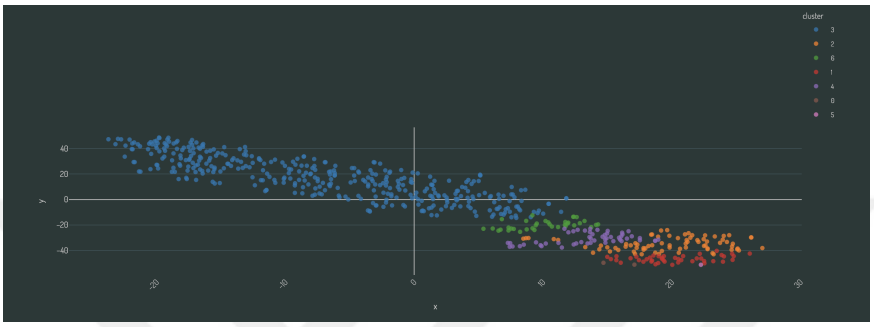

Figure 6.46: TSNE Scatter Plot of 7 clusters With GloVe Tf-Idf Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 30

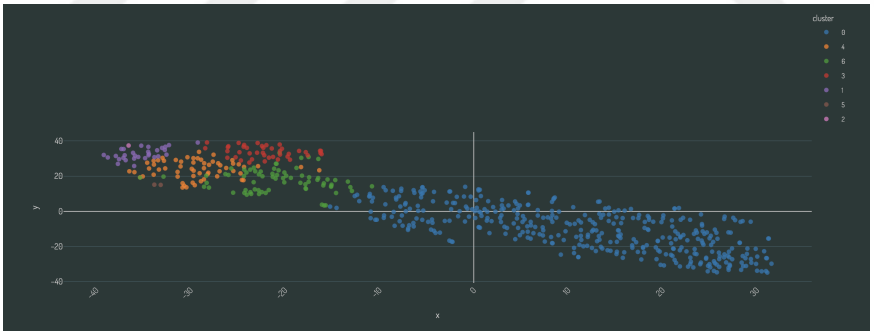

Figure 6.47: TSNE Scatter Plot of 7 clusters With GloVe Tf-Idf Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 30

Both Agglomerative (Figure 6.46) and K-Means (Figure 6.47) scatter plots show clusters as if there is one main cluster and 6 other separated clusters. The formed clusters are similar to the ones in Word2Vec method with tf-idf and Agglomerative Clustering. However, the clusters are less dense and more noisy. So, it is fair that GloVe has smaller silhouette score for Agglomerative Clustering. For the K-Means clustering,

main cluster of Word2Vec is noisier but better separated. So, their scores are actually similar to each other for K-Means Clustering. Actually for averaging methods, GloVe also has given noisy visualisations. So, it is obvious that GloVe sees data set to be noisy.

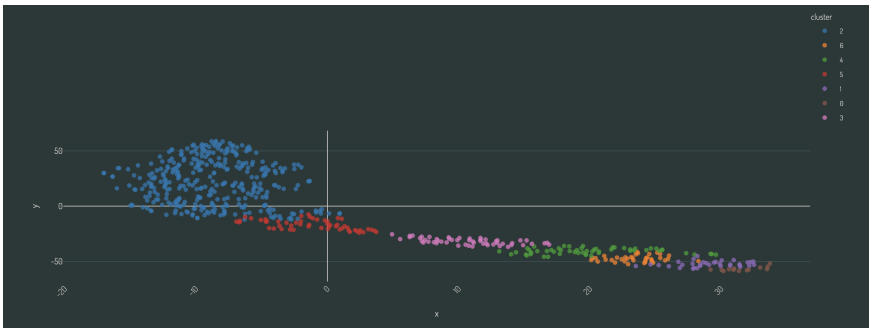

Figure 6.48: TSNE Scatter Plot of 7 clusters With Fasttext Tf-Idf Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 30

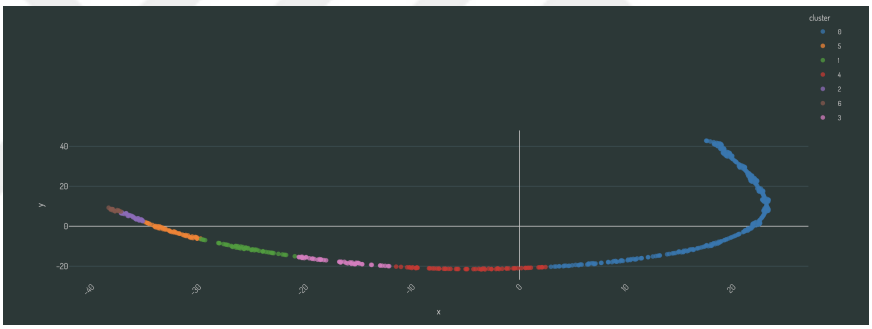

Figure 6.49: TSNE Scatter Plot of 7 clusters With Fasttext Tf-Idf Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 30

Fasttext also expects a main cluster but denser for Agglomerative Clustering (Figure 6.48). The other clusters seemed to be linear, so less variant inside the cluster. In compare to Word2Vec, well separated clusters are formed, where the Word2Vec clusters are denser. So, Word2Vec gains points for silhouette score for this and the main cluster density gives points to Fasttext. For K-Means, Fasttext expects a linear and not-varying, so hard-to-separate data. This is similar to the averaging method. Thus, Fasttext predicts a less varying data point distribution.

In general methods with tf-idf predicts one main cluster and smaller and equally dis-

tributed small clusters. This works for all three methods, where Fasttext predicts denser main cluster, Word2Vec predicts smaller clusters to be denser and GloVe predicts more distributed kinds of clusters.

Table 6.9: Best Rand Index Results Table with Term Frequency - Inverse Document Frequency Combination Strategy, Agglomerative Clustering

Method	**Option**	**Score**
word2vec	tf-idf	0.0487
glove	tf-idf	0.0329
fasttext	**tf-idf**	**0.0538**

Table 6.10: Best Rand Index Results Table with Term Frequency - Inverse Document Frequency Combination Strategy, K-Means Clustering

Method	**Option**	**Score**
word2vec	tf-idf	0.0344
glove	tf-idf	0.0272
fasttext	**tf-idf**	**0.0648**

Rand indices for the both Agglomerative Clustering and K-Means clustering are too low. The silhouette scores prove that data is separated better than averaging methods but the prediction accuracy is almost zero. The shapes are good but the context is not captured well with tf-idf weighting. Actually, it is not false to say that weighted averaging breaks the pattern discovered by word embeddings. However, even if the rand index results are very low, the scatter plots according to the best rand index create well-separated clusters in general, so it is worth to analyse them as well.

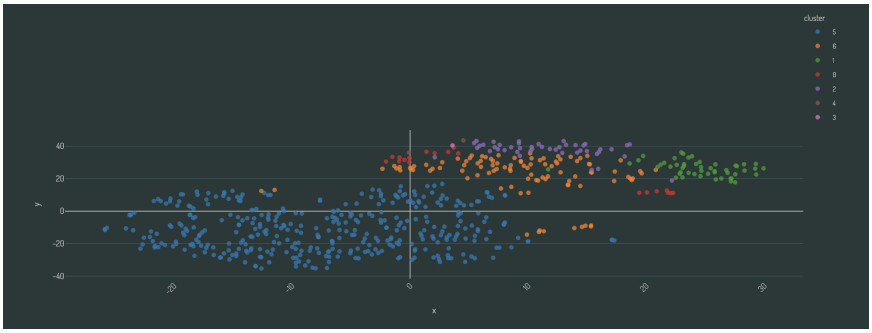

Figure 6.50: TSNE Scatter Plot of 7 clusters With Word2Vec Tf-Idf Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30

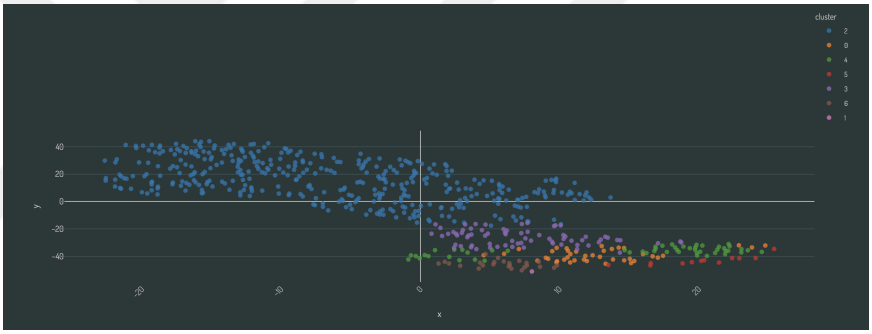

Figure 6.51: TSNE Scatter Plot of 7 clusters With Word2Vec Tf-Idf Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30

Agglomerative Clustering (Figure 6.50) with best rand score gives a noisier data point distribution than the one having the best silhouette scores (Figure 6.44). Actually, there is small difference between rand indices of both trials and it shows that Word2Vec tf-idf gives bad rand index independent of distribution of data points. The Agglomerative Clustering results have more well-separated clusters and K-Means Clustering results are noisier. So, the former has bigger rand index which means some points of a few points go in correct clusters with Agglomerative than K-Means.

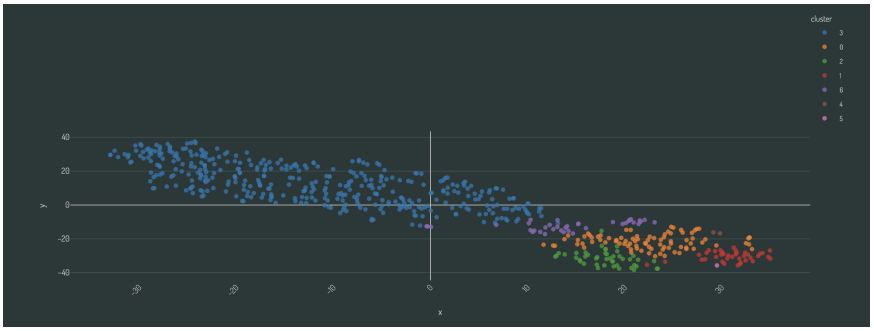

Figure 6.52: TSNE Scatter Plot of 7 clusters With GloVe Tf-Idf Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30

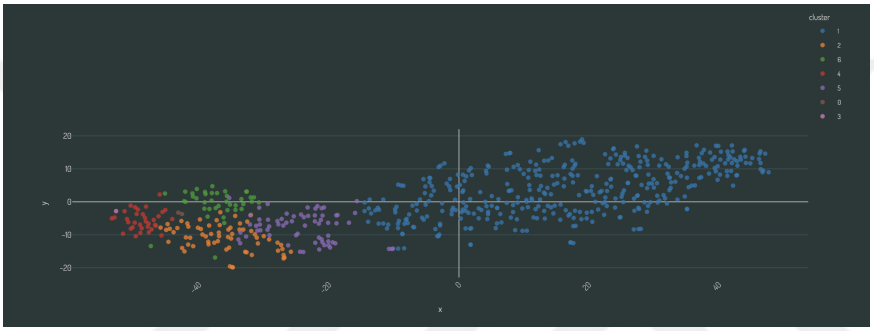

Figure 6.53: TSNE Scatter Plot of 7 clusters With GloVe Tf-Idf Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30

GloVe with the best rand index results are very similar to the results of the Glove having best silhouette score: non-dense but well-separated clusters, which can be the reason that GloVe has the lowest rand index value.

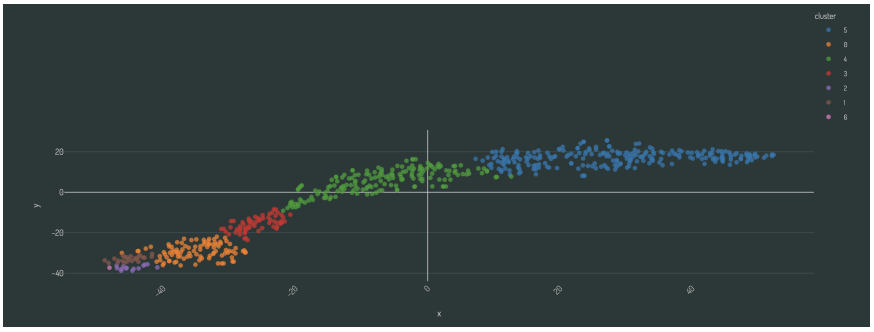

Figure 6.54: TSNE Scatter Plot of 7 clusters With Fasttext Tf-Idf Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 30

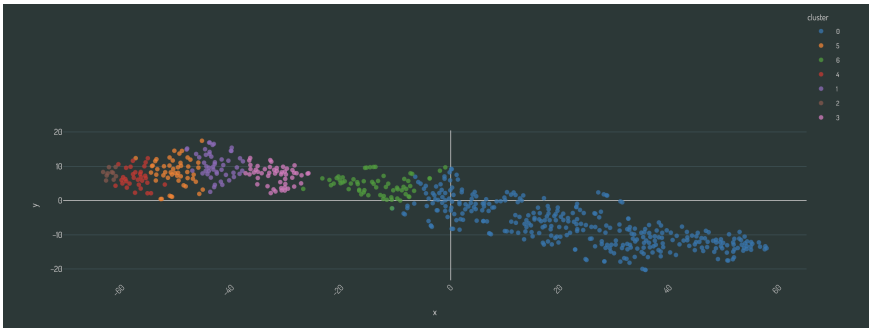

Figure 6.55: TSNE Scatter Plot of 7 clusters With Fasttext Tf-Idf Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 30

Fasttext results give good clustering visualisation for both Agglomerative and K-Means Clustering. They seem to be better than GloVe and Word2Vec. Also, the rand index scores are higher. So, a correlation between the better clustering visualisation and rand index in terms of methods with tf-idf combination strategy. However, the too smaller values of rand index in contrary to bigger silhouette scores is a mystery to this analysis.

6.1.3 Averaging Combination Strategy Extended

Embedding methods that use the averaging as a combination strategy have a lot of methods to analyse so, they are analysed in two different sections by dividing the methods according to the rand indices. The ones having better rand indices was in the Section 6.1.1 and the ones having lower rand indices are analyzed in this section. . So, it can be expected that the accuracy of these methods are low in terms of given labels for documents.

Table 6.11: Best Silhouette Score Results Table with Averaging Combination Strategy, Agglomerative Clustering Extended

Method	Option	5 Clusters	10 Clusters	15 Clusters	20 Clusters
bert	**sentence-avg**	**0.7618**	**0.6964**	**0.5237**	**0.5110**
roberta	sentence-avg	0.4839	0.3152	0.2882	0.2672
gpt2	sentence-avg	0.4536	0.3938	0.3660	0.3516
skip thoughts	sentence-avg	0.0408	0.0254	0.0247	0.0076

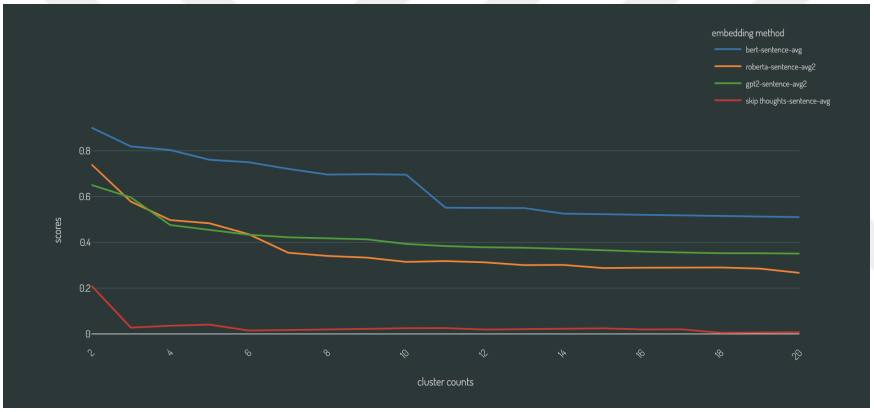

Figure 6.56: Best Silhouette Score Results Graph with Averaging Combination Strategy and Agglomerative Clustering Method Extended

According to Table 6.11 and Figure 6.56, BERT is the best embedding method in terms of silhouette score; the RoBERTa and GPT2 have similar scores and Skip Thoughts give a very low silhouette score. BERT has a stable silhouette score graph. Outliers are not expected much. Also, the score is high, so BERT must have had dense and/or well-separated clusters. RoBERTa and GPT2 has similar silhouette scores. However, Doc2Vec has the lowest silhouette scores. The reason can be about outliers or predicting data to be inseparable.

Table 6.12: Best Silhouette Score Results Table with Averaging Combination Strategy, K-Means Clustering Extended

Method	Option	5 Clusters	10 Clusters	15 Clusters	20 Clusters
bert	**sentence-avg**	**0.5718**	**0.4870**	**0.4475**	**0.4293**
roberta	sentence-avg	0.1560	0.0800	0.0496	0.0336
gpt2	sentence-avg	0.0553	0.0248	0.0090	0.0125
skip thoughts	sentence-avg	0.0362	0.0316	0.0190	0.0267

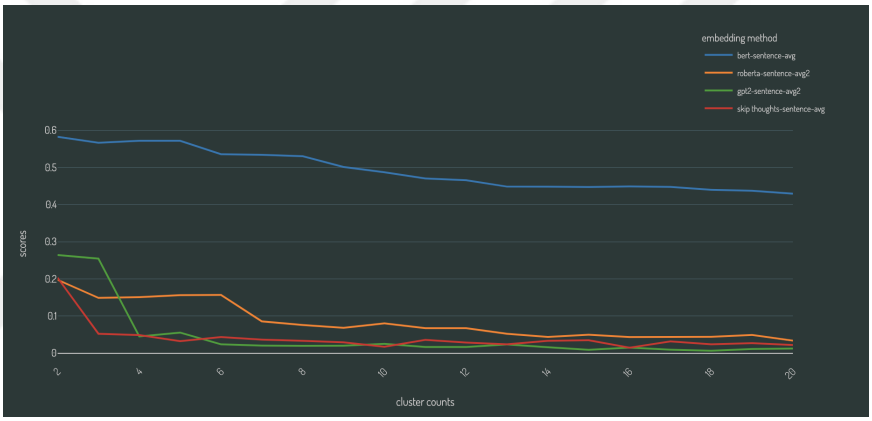

Figure 6.57: Best Silhouette Score Results Graph with Averaging Combination Strategy and K-Means Clustering Method Extended

K-Means results are lower than the ones with Agglomerative Clustering (Table 6.12 and Figure 6.57). Actually, only the BERT has a worthwhile silhouette score. The other ones are generally close to 0. Among them, RoBERTa has the best performance. Also, these three methods are expected to have outliers since their scores decrease as the cluster count increases. On the other hand, BERT has slightly worse but more stable scores in this than the ones with Agglomerative Clustering. So, the BERT can be the only source of information, the others are not expected to be well clustered with K-Means algorithm.

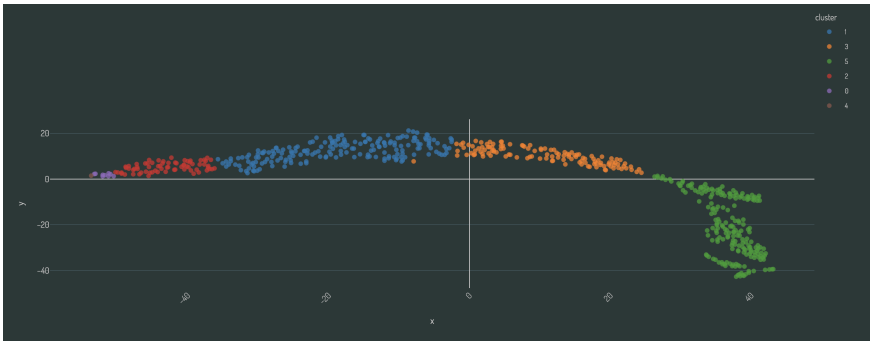

Figure 6.58: TSNE Scatter Plot of 7 clusters With BERT Averaging Extended Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 40

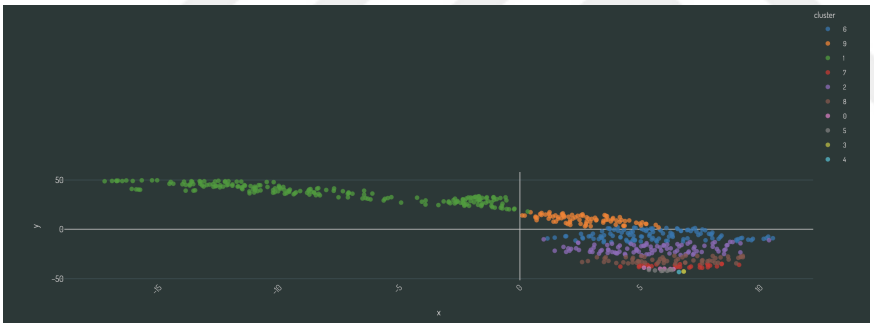

Figure 6.59: TSNE Scatter Plot of 7 clusters With BERT Averaging Extended Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 40

BERT has well separated and less variant clusters for Agglomerative and K-Means both (Figure 6.58, 6.59). But for the K-Means, there is more noise inside clusters. So, BERT can be thought as giving a good clustering performance for the best silhouette scores.

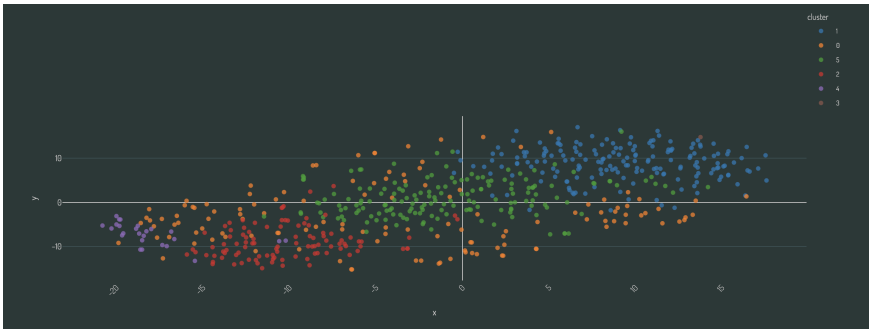

Figure 6.60: TSNE Scatter Plot of 7 clusters With RoBERTa Averaging Extended Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 40

With K-Means, data points are visualised as one inseparable cluster and clusters are not well-formed (Figure 6.60). So, RoBERTa seems useless with K-Means clustering. Also, for the Agglomerative Clustering, there is noise, and dense and well-separated clusters do not form. However, it has a relatively better silhouette score for Agglomerative Clustering since there are clusters forming at least.

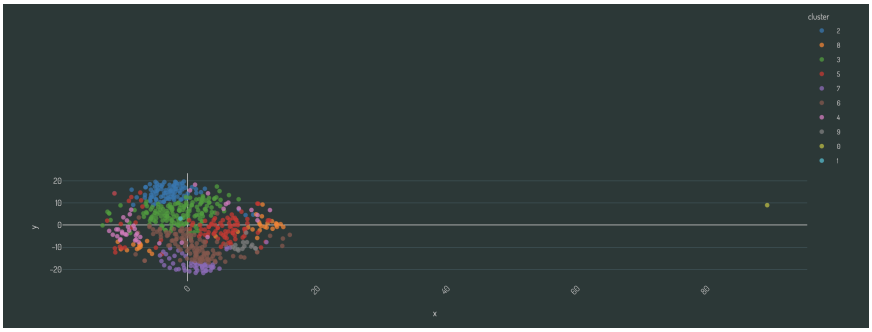

Figure 6.61: TSNE Scatter Plot of 7 clusters With RoBERTa Averaging Extended Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 40

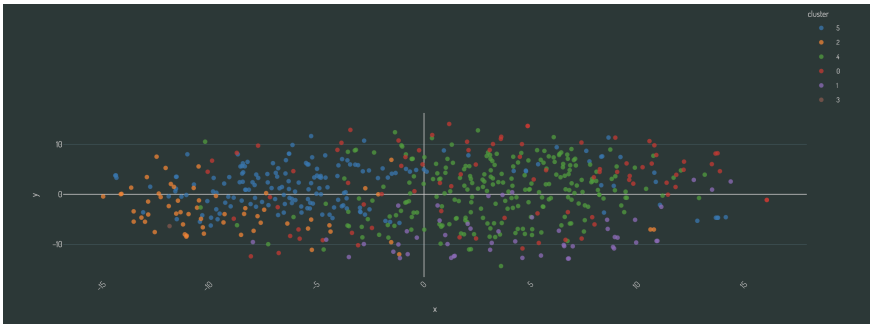

Figure 6.62: TSNE Scatter Plot of 7 clusters With GPT2 Averaging Extended Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 40

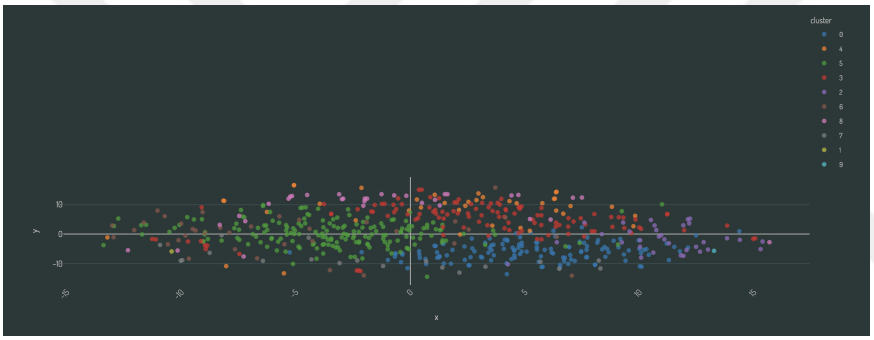

Figure 6.63: TSNE Scatter Plot of 7 clusters With GPT2 Averaging Extended Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 40

GPT2 is as noisy as RoBERTa. So, the silhouette scores are relatively low according to BERT (Figure 6.61, 6.62). The points clustered with Agglomerative Clustering has a higher score than the ones clustered with K-Means. However, Figure 6.62 and Figure 6.63 seems to be equally bad. Sometimes, data cannot be represented well because of data loss from dimensionality reduction when the dimension of actual data vector is very high (768 in this case). There can be noise but having the same noise, not having outliers and having very different silhouette scores can be explained with this.

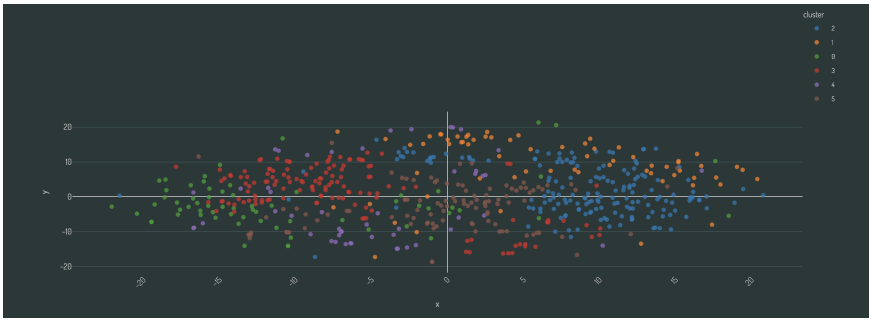

Figure 6.64: TSNE Scatter Plot of 7 clusters With Skip Thoughts Averaging Extended Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 40

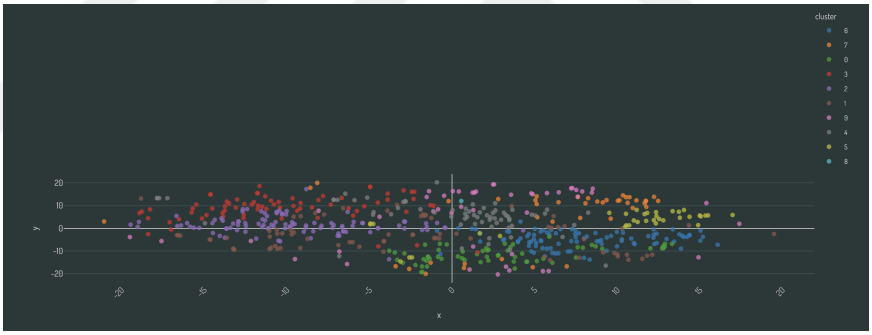

Figure 6.65: TSNE Scatter Plot of 7 clusters With Skip Thoughts Averaging Extended Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 40

Skip Thoughts have the lowest silhouette scores and there are the noisiest and non-dense clusters in it. It should be noted that Skip Thought vectors are trained with only one trial, so with no hyper-parameter optimization. Thus, the Skip Thoughts have formed the less well separated clusters if there is any cluster.

Table 6.13: Best Rand Index Results Table with Averaging Combination Strategy, Agglomerative Clustering Extended

Method	Option	Score
BERT	sentence-averaging	0.0094
RoBERTa	sentence-averaging	0.0836
GPT2	sentence-averaging	0.1083
Skip Thoughts	**sentence-averaging**	**0.2244**

Table 6.14: Best Rand Index Results Table with Averaging Combination Strategy, K-Means Clustering Extended

Method	Option	Score
BERT	sentence-averaging	0.0164
RoBERTa	sentence-averaging	0.0845
GPT2	sentence-averaging	0.1003
Skip Thoughts	**sentence-averaging**	**0.1628**

Rand indices are very close to zero. This means that the methods in this category does not do well in terms of accuracy according to given labels to some portion of documents. Also, there is a counter relation between the silhouette score and rand index. This can be because the data actually is not separable or methods having better silhouette scores capture different patterns from the data. It is possible that data is inseparable since different methods, some of which have good rand indices as well, predicted data to be inseparable. If silhouette scores capture different patterns from the data, this can mean that there are clusters that requires to form from the not annotated, i.e. labeled, documents. They can have a pattern and this will not be captured by the rand index values. Actually, there is one more possibility. These methods all use external, big models to be trained on the existing data set. So, the previous weights, or embeddings can be dominant and the local context cannot be captured.

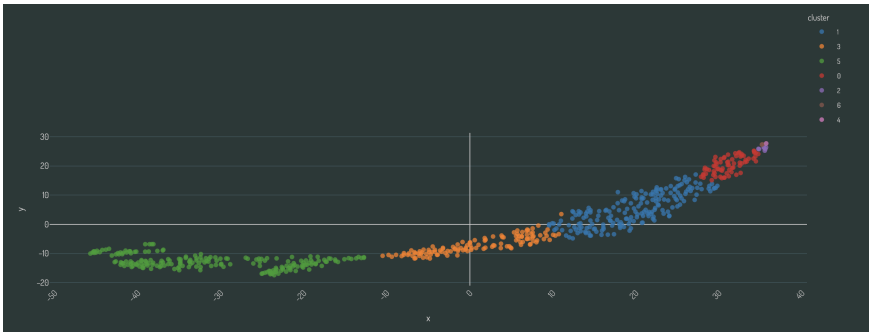

Figure 6.66: TSNE Scatter Plot of 7 clusters With BERT Averaging Extended Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 40

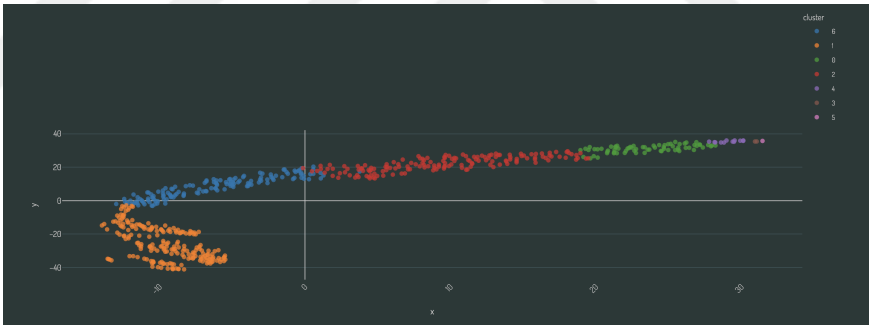

Figure 6.67: TSNE Scatter Plot of 7 clusters With BERT Averaging Extended Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 40

Both plots (Figure 6.66 and Figure 6.67) are similar to the ones with best silhouette scores. Also, there is a linearity in the data similar to Fasttext results with the best silhouette scores. Actually, clusters are forming in here as well but the accuracy is the lowest among all the methods in this section.

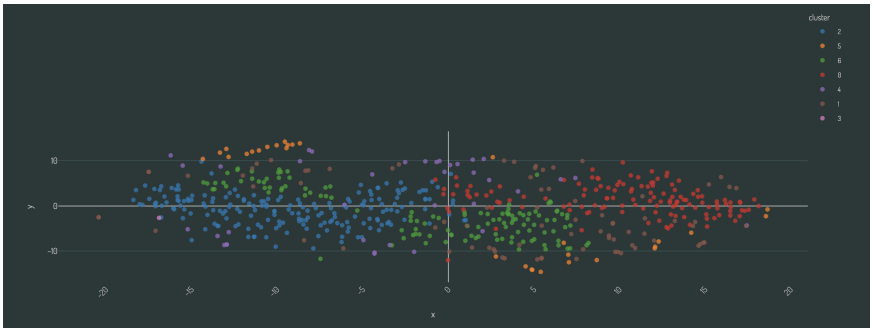

Figure 6.68: TSNE Scatter Plot of 7 clusters With RoBERTa Averaging Extended Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 40

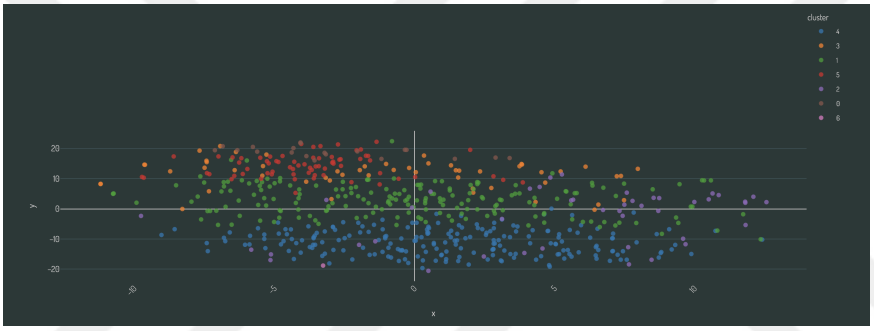

Figure 6.69: TSNE Scatter Plot of 7 clusters With RoBERTa Averaging Extended Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 40

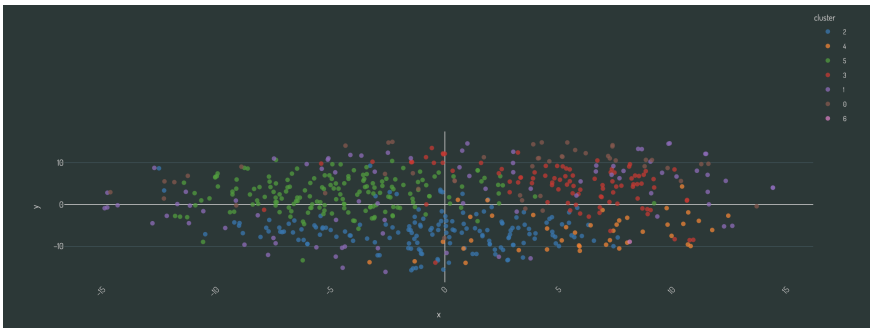

Figure 6.70: TSNE Scatter Plot of 7 clusters With GPT2 Averaging Extended Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 40

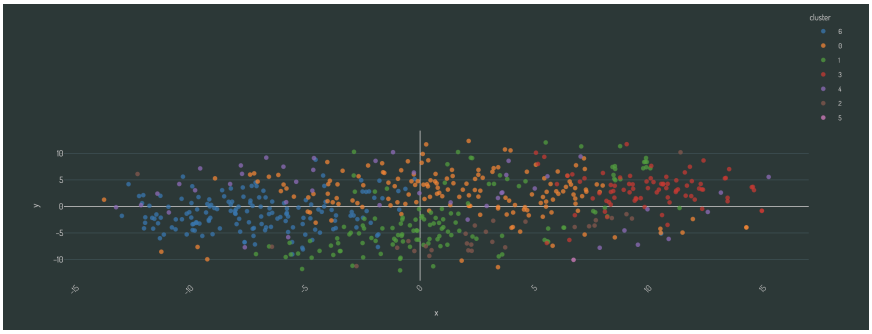

Figure 6.71: TSNE Scatter Plot of 7 clusters With GPT2 Averaging Extended Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 40

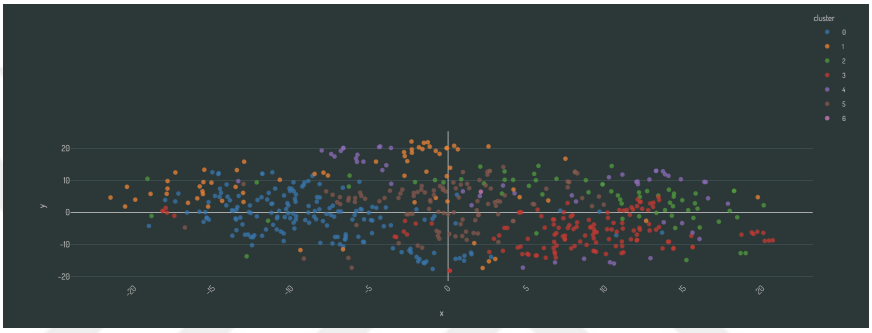

Figure 6.72: TSNE Scatter Plot of 7 clusters With Skip Thoughts Averaging Extended Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 40

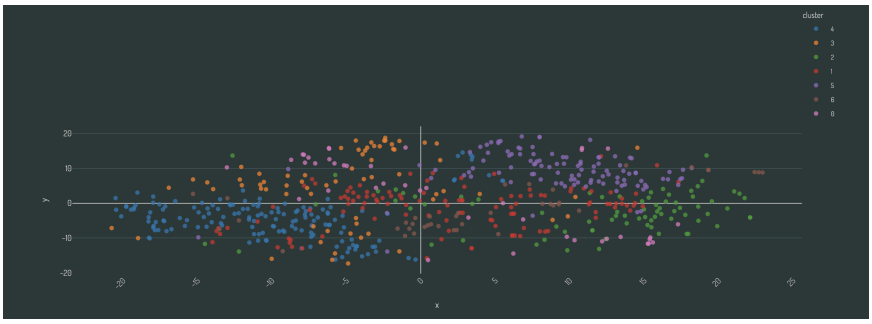

Figure 6.73: TSNE Scatter Plot of 7 clusters With Skip Thoughts Averaging Extended Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 40

RoBERTa, GPT2 and Doc2Vec all have noisy results, so one will not expect good cluster accuracy and their accuracy is very low. But the Skip Thoughts have a better

accuracy, where the difference between RoBERTa and GPT2 can be ignored. Skip Thoughts has a relatively smaller network than others and the local structure of data set affects much, so this can be the reason that accuracy is better with Skip Thoughts.

6.1.4 Best Results With Other Clustering Metrics

Until now, silhouette score and rand index are used for measuring the clustering quality. In this section, two best embedding methods in terms of the quality of clusters they create: Fasttext with averaging and Sentence BERT with sentence averaging. As in the previous sections, the clustering results is evaluated and analysed with two different clustering metrics: Davies-Bouldin Index and Homogeneity Score.

Table 6.15: Best Davies-Bouldin Index Results of Best Embedding Methods, Agglomerative Clustering

Method	Option	5 Clusters	10 Clusters	15 Clusters	20 Clusters
fasttext	avg	0.7107	0.6531	0.6577	0.6931
sentence bert	sentence-avg	0.5926	0.6916	0.6991	0.6979

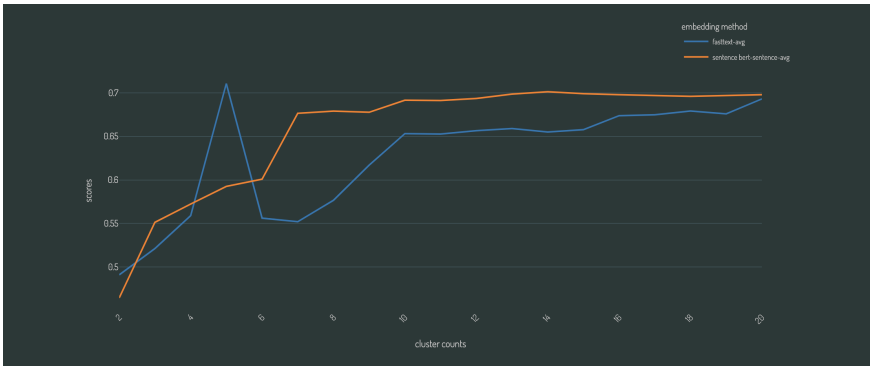

Figure 6.74: Best Davies-Bouldin Index Line Graph of Best Embedding Methods, Agglomerative Clustering

For Agglomerative Clustering, the Davies-Bouldin Index values are shown in Table 6.15 and Figure 6.74. For Davies-Bouldin Index, smaller values are better and the smallest value is zero. In the given table and figure, the index values seem good

enough and the scores for both of the methods are close to each other. So, their clusters will have similar quality. In order to choose a cluster count to analyse, 6 seems good since Sentence BERT has an elbow point at 6 clusters and Fasttext has a local minima there.

Table 6.16: Best Davies-Bouldin Index Results of Best Embedding Methods, K-Means Clustering

Method	Option	5 Clusters	10 Clusters	15 Clusters	20 Clusters
fasttext	avg	2.7143	2.3064	2.1682	2.1068
sentence bert	sentence-avg	2.8851	2.7020	2.5238	2.2841

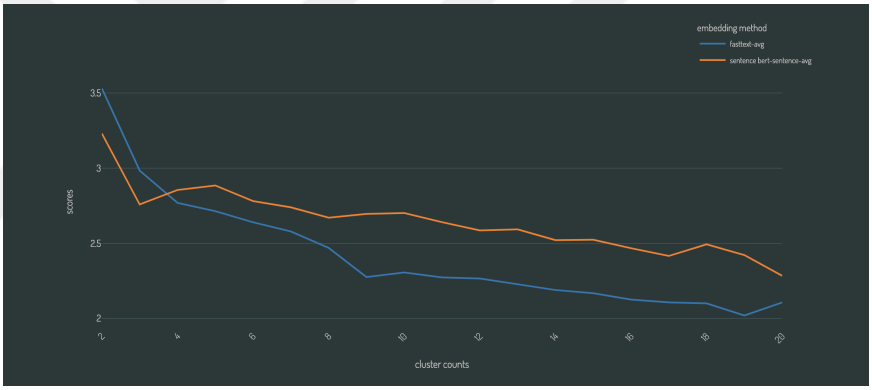

Figure 6.75: Best Davies-Bouldin Index Line Graph of Best Embedding Methods, K-Means Clustering

For K-Means clustering, the Davies-Bouldin Index values are shown in Table 6.16 and Figure 6.75. The values are lower according to the ones in Agglomerative Clustering. This means that the Agglomerative Clustering results is going to give better clustering visualisation as well separated and denser clusters. For Fasttext results clustered with K-Means, 9 clusters point is a local minima for Davies-Bouldin Index. For the Sentence BERT, 8 clusters point is a local minima. So, these points can be used as cluster counts of scatter plots.

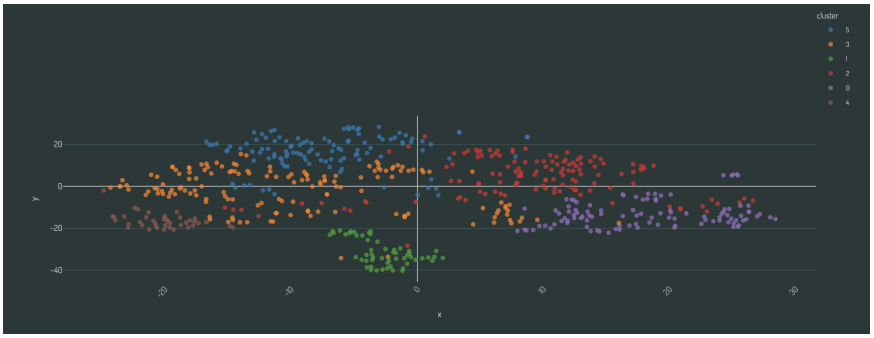

Figure 6.76: TSNE Scatter Plot of 6 clusters With Fasttext Averaging Having Best Davies-Bouldin Index Strategy and Agglomerative Clustering Method, perplexity: 30

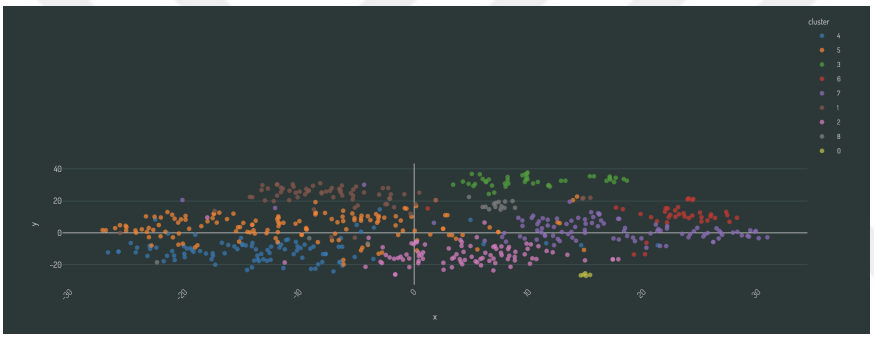

Figure 6.77: TSNE Scatter Plot of 9 clusters With Fasttext Averaging Having Best Davies-Bouldin Index Strategy and K-Means Clustering Method, perplexity: 30

In Agglomerative Clustering result (Figure 6.76), the clusters are formed and separated very well but the clusters are not dense. The clusters can be seen very obvious and clusters are equally crowded. Data points in this trial, which has the best Davies-Bouldin scores, are separated better than the trial chosen as having best silhouette score for Fasttext embedding method. On the other hand, the K-Means results (Figure 6.77) are very similar to the ones with Agglomerative Clustering except that the former has more intertwined clusters which can be the reason of lower scores. Looking at the both results, except the little noise or non-denseness, there are clusters formed well.

Figure 6.78: TSNE Scatter Plot of 6 clusters With Sentence BERT Sentence Averaging Having Best Davies-Bouldin Index Strategy and Agglomerative Clustering Method, perplexity: 30

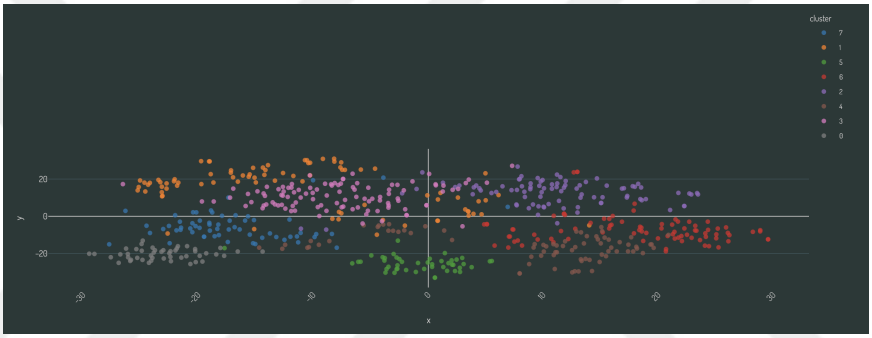

Figure 6.79: TSNE Scatter Plot of 6 clusters With Sentence BERT Sentence Averaging Having Best Davies-Bouldin Index Strategy and K-Means Clustering Method, perplexity: 30

Sentence BERT results with Davies-Bouldin Index (Figure 6.78 and Figure 6.79) is similar to Fasttext results in terms of cluster visualisation but this one is noisier. So, the Fasttext results are slightly better

Table 6.17: Best Homogeneity Score Results of Best Embedding Methods, Agglomerative Clustering

Method	**Option**	**7 Clusters**
fasttext	avg	0.6886
sentence bert	sentence-avg	0.6631

Table 6.18: Best Homogeneity Score Results of Best Embedding Methods, K-Means Clustering

Method	Option	7 Clusters
fasttext	avg	0.7081
sentence bert	sentence-avg	0.7011

According to the homogeneity score they cannot beat each other, where Fasttext is better with K-Means clustering and Sentence BERT is better with Agglomerative Clustering6.17 and Table 6.18). Homogeneity scores are labeled metric as rand index. So, we can expect better predictions about the trials since this index includes human help in contrary to Davies-Bouldin Index. When it is compared with the rand index, there are better scores. Also, the K-Means results are over 0.70 and over Agglomerative ones for Homogeneity Score.

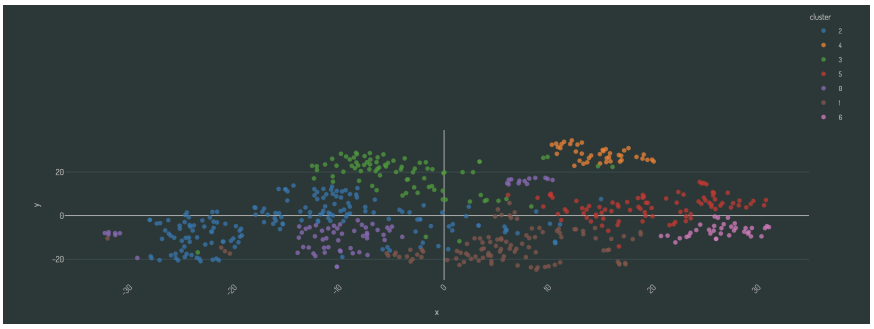

Figure 6.80: TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Homogeneity Score Strategy and Agglomerative Clustering Method, perplexity: 30

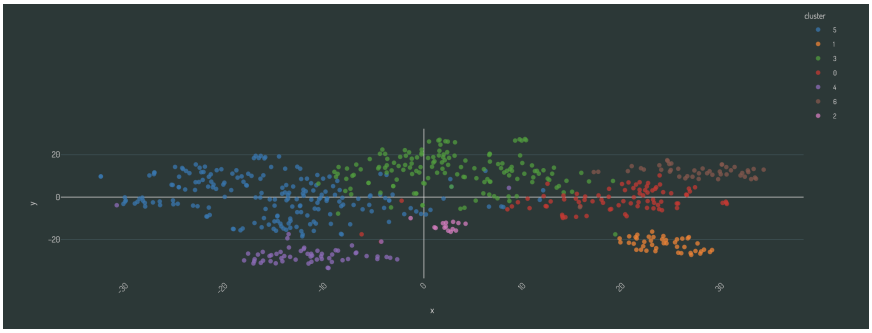

Figure 6.81: TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Homogeneity Score Strategy and K-Means Clustering Method, perplexity: 30

As in the Davies-Bouldin part, good clusters are formed in here as well, the intra cluster distances are even high but lower than the ones with Davies-Bouldin Index, especially for K-Means clustering results plot (Figure 6.81). Actually, K-Means clustering results are also denser than all. This can be taken as the closest visualisation to ideal.

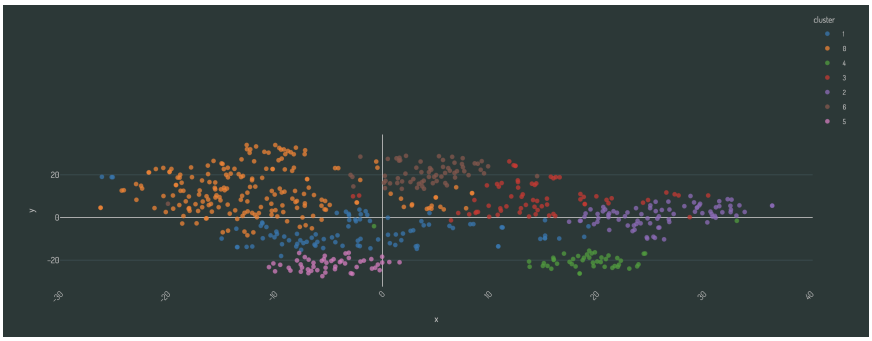

Figure 6.82: TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Homogeneity Score Strategy and Agglomerative Clustering Method, perplexity: 30

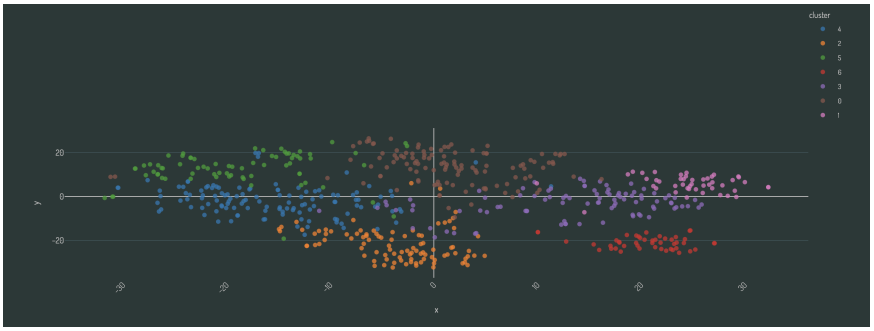

Figure 6.83: TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging
Having Best Homogeneity Score Strategy and K-Means Clustering Method, perplexity: 30

As the Fasttext visualisations having the best Homogeneity score, the Sentence BERT results include well separated and less noisy clusters (Figure 6.82 and Figure 6.83). K-Means result of this has slightly better clusters and this can be the second the best clustering after Fasttext with K-Means clustering method.

To conclude, there are mostly better clustering visualisation results with Davies-Bouldin and Homogeneity than Silhouette and Rand scores for these two methods.

6.1.5 Clustering Quality Change With Parameter Optimization

Optimization with different trials is the backbone of this thesis and there is a two-phase optimization process. In the first part, hyper-parameters of embedding methods are optimized as the objective function is average silhouette score. Average silhouette score is the mean of silhouette scores for cluster counts from 2 to 20, where different clustering methods are used with their default parameters. The second part is the optimization of clustering method parameters. For each clustering method clustering parameters are optimized for each cluster count separately, where silhouette score for the given cluster count is the objective function.

In this section, the improvements of average scores (defined as average silhouette score for cluster counts from 2 to 20 for a given embedding method, combination

strategy and clustering method) through different trials are analysed. The average score for each trial of first optimization part is taken and averaged for a given embedding method, combination strategy and clustering method, where each trial has one average score. In this calculation, clustering method parameters are in their default values. This values are called "Avg. of Avg. Scores" and used as a baseline for performances. Then the result having best average score is taken instead of average in "Avg. of Avg. Scores". This is called "Best of Avg. Scores". These two values are compared and the improvement is written in Rise1 (Table 6.19). Next, clustering method optimizations are applied and each silhouette score for each cluster count is optimized. As a metric, only best trial according to the average silhouette score is chosen and called "Best of Optimized Avg Scores". This value is compared with "Best of Avg. Scores" and the rise in the score is in the Rise2 column in Table 6.19. The overall increase is the composite increase of Rise1 and Rise2. In the experiments the trials are chosen according to "Best of Optimized Avg Scores".

According to table, there are improvements more than 100% for Word2Vec and Fasttext with averaging method with optimization of hyper-parameters of embedding methods. Fasttext has the most number of parameters, so it is not a coincidence it has such an improvement. Doc2Vec, GloVe, RoBERTa and GPT2 have a medium improvement. Also, they have a good number of hyper-parameters, but they do not improve as much as the Word2Vec, whose hyper-parameter count is similar to these methods. This can be because of the limits of methods or lack of trials. Methods with tf-idf combination strategy do not improvement as much as the ones with averaging combination strategy. The difference between the rises certainly shows that the potential of tf-idf to be optimized is not as good as averaging. Sentence BERT and SciBERT also show less improvement from the most methods. Even if their rand indices are very high the silhouette score values are not very good and do not improve much. So, they can be thought as creating non-well clustered clusters and even have more accuracy since they are capturing the patterns in data in a more different way than word embedding methods. 'sentence-avg' and 'sentence-avg2' refers to two different training ways for the corresponding embedding methods as explained before. According to the comparison between these two combination strategy, they have similar improvements and similar scores at hand, where the 'sentence-avg2' has slightly

better scores. Improvements according to the clustering method are also similar between K-Means and Agglomerative clustering methods mostly. As a special case, BERT has one hyper-parameter and it seems it cannot affect the scores at all.

Table 6.19: Improvements with Optimizations

Method	**Method Option**	**Clustering Method**	**Avg of Avg Scores**	**Best of Avg Scores**	**Best of Opt Avg Scores**	**Rise1(%)**	**Rise2(%)**
word2vec	avg	Agglomerative	0.1105	0.24	0.2941	117.2373	22.5265
word2vec	avg	K-Means	0.1309	0.2872	0.2907	119.4441	1.2108
word2vec	tf-idf	Agglomerative	0.5077	0.5578	0.6808	9.8704	22.0538
word2vec	tf-idf	K-Means	0.5247	0.5529	0.576	5.3791	4.1768
glove	avg	Agglomerative	0.1024	0.1476	0.219	44.1315	48.324
glove	avg	K-Means	0.1218	0.1758	0.1832	44.3456	4.2107
glove	tf-idf	Agglomerative	0.4599	0.511	0.672	11.1185	31.5115
glove	tf-idf	K-Means	0.4672	0.4952	0.5386	5.9941	8.7444
glove	pre-prepared	Agglomerative	0.0805	0.0961	0.2163	19.4706	125.0223
glove	pre-prepared	K-Means	0.0937	0.1131	0.1176	20.7255	3.9722
fasttext	**avg**	**Agglomerative**	**0.1851**	**0.5227**	**0.5877**	**182.4369**	**12.4225**
fasttext	avg	K-Means	0.2031	0.5362	0.5418	164.0236	1.0393
fasttext	tf-idf	Agglomerative	0.5039	0.6018	0.6814	19.4327	13.2152
fasttext	tf-idf	K-Means	0.516	0.6201	0.6239	20.1783	0.6161
bert	sentence-avg	Agglomerative	0.527	0.527	0.6391	0.0	21.2822
bert	sentence-avg	K-Means	0.53	0.5302	0.4929	0.0233	-7.0291
roberta	sentence-avg	Agglomerative	0.2183	0.3651	0.3891	67.2521	6.5617
roberta	sentence-avg	K-Means	0.0592	0.0693	0.0871	17.1324	25.6856
roberta	sentence-avg2	Agglomerative	0.1605	0.2645	0.37	64.8055	39.9031
roberta	sentence-avg2	K-Means	0.0523	0.0602	0.0848	15.2111	40.9259
gpt2	sentence-avg	Agglomerative	0.2036	0.2808	0.3831	37.9612	36.4348
gpt2	sentence-avg	K-Means	0.0156	0.0217	0.0487	39.2794	124.1613
gpt2	sentence-avg2	Agglomerative	0.2272	0.2665	0.4166	17.2968	56.339
gpt2	sentence-avg2	K-Means	0.0132	0.0173	0.0454	31.0714	162.6333
scibert	sentence-avg	Agglomerative	0.1219	0.1253	0.1649	2.7934	31.5687
scibert	sentence-avg	K-Means	0.0803	0.0815	0.0893	1.5356	9.6244
scibert	sentence-avg2	Agglomerative	0.1234	0.1275	0.1703	3.3574	33.5719
scibert	sentence-avg2	K-Means	0.08	0.0809	0.089	1.2056	9.8863
sent bert	sentence-avg	Agglomerative	0.0817	0.0883	0.1572	8.0544	77.9748
sent bert	sentence-avg	K-Means	0.0849	0.093	0.0982	9.6312	5.5771
doc2vec	doc-embedding	Agglomerative	0.1959	0.2413	0.4147	23.1422	71.8703
doc2vec	doc-embedding	K-Means	0.2117	0.2685	0.2747	26.8413	2.3306

This turns us into the Rise2 values, which are the improvements via clustering method hyper-parameters. Agglomerative Clustering improvements are very high. This is mostly because the Agglomerative Clustering results create more outliers than K-Means results and silhouette score is sensitive to outliers. The most improvements are in GPT2 and Glove with pre-trained embeddings. THe GPT2 does not have good silhouette score averages and this increase can be a coincidence but GloVe with pre-trained embeddings improvement moves the method from very low to medium among methods.

6.2 Experimenting With Other Data Sets with Chosen Methods

In order to understand the power of trials, it is applied to other data sets: one is very smaller and one is very larger data sets.

6.2.1 Cappadocia Papers Data Set

Cappadocia paper collection is the data set that is collected in the context of this thesis. It contains papers that are related with Cappadocia area. It has 58 documents and the 27 of them labeled.

Table 6.20: Best Silhouette Scores Results of Best Embedding Methods, Agglomerative Clustering, Cappadocia Dataset

Method	Option	5 Clusters	10 Clusters	15 Clusters	20 Clusters
fasttext	avg	0.5544	0.5868	0.6249	0.6249
sentence bert	sentence-avg	0.1591	0.1486	0.1247	0.1177

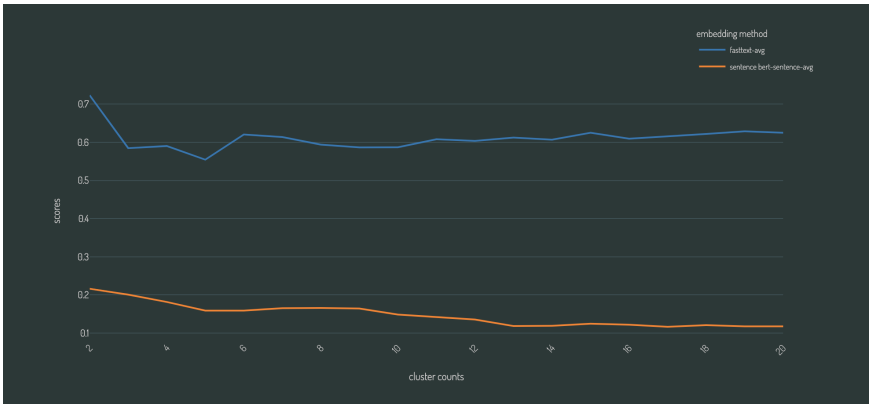

Figure 6.84: Best Silhouette Score Line Graph of Best Embedding Methods, Agglomerative Clustering, Cappadocia Dataset

In general, scores are similar to the ones with Multi-Disciplinary Papers Dataset, the scores are in medium level. According to silhouette scores with Agglomerative Clustering (Table 6.20 and Figure 6.84), Fasttext has a so higher than Sentence BERT that Fasttext promises a better clustering result. They are both stable mostly in terms of scores over cluster counts. This means that there are less outliers in the results of clustering.

Table 6.21: Best Silhouette Scores Results of Best Embedding Methods, K-Means Clustering, Cappadocia Dataset

Method	**Option**	**5 Clusters**	**10 Clusters**	**15 Clusters**	**20 Clusters**
fasttext	avg	0.5567	0.5833	0.6251	0.6319
sentence bert	sentence-avg	0.1652	0.1385	0.1202	0.1242

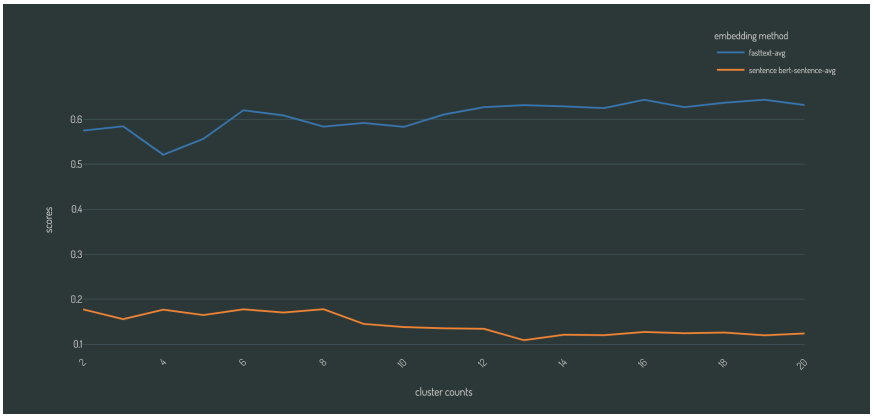

Figure 6.85: Best Silhouette Score Line Graph of Best Embedding Methods, K-Means Clustering, Cappadocia Dataset

K-Means results (Table 6.21 and Figure 6.85) also has stability over cluster counts and the scores are similar. Because less outliers are expected, the difference between Agglomerative and K-Means clustering results diminishes.

Table 6.22: Best Davies-Bouldin Index Results of Best Embedding Methods, Agglomerative Clustering, Cappadocia Dataset

Method	**Option**	**5 Clusters**	**10 Clusters**	**15 Clusters**	**20 Clusters**
fasttext	avg	0.7127	0.7324	0.7022	0.7650
sentence bert	sentence-avg	0.7294	0.7209	0.7658	0.8389

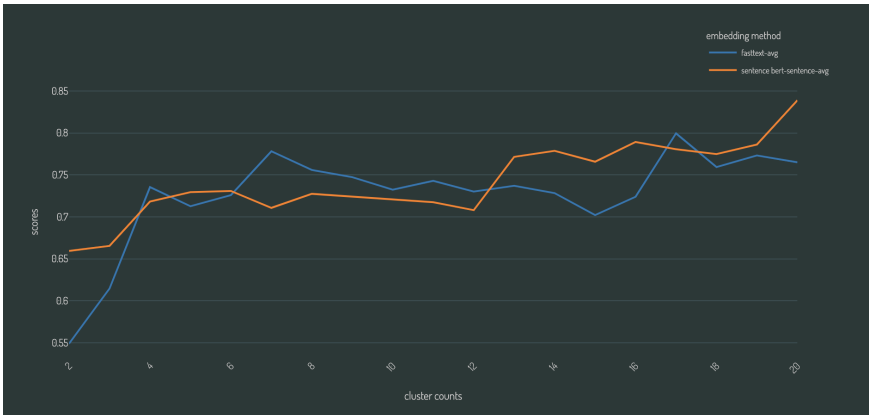

Figure 6.86: Best Davies-Bouldin Index Line Graph of Best Embedding Methods, Agglomerative Clustering, Cappadocia Dataset

According to Davies-Bouldin indices for Agglomerative Clustering, the clustering quality is closer to zero but not very close. So, the clustering separation and density of clusters will be good relatively.

Table 6.23: Best Davies-Bouldin Index Results of Best Embedding Methods, K-Means Clustering, Cappadocia Dataset

Method	**Option**	**5 Clusters**	**10 Clusters**	**15 Clusters**	**20 Clusters**
fasttext	avg	1.4460	1.3767	1.1580	0.9600
sentence bert	sentence-avg	1.6678	1.2874	1.1119	1.0745

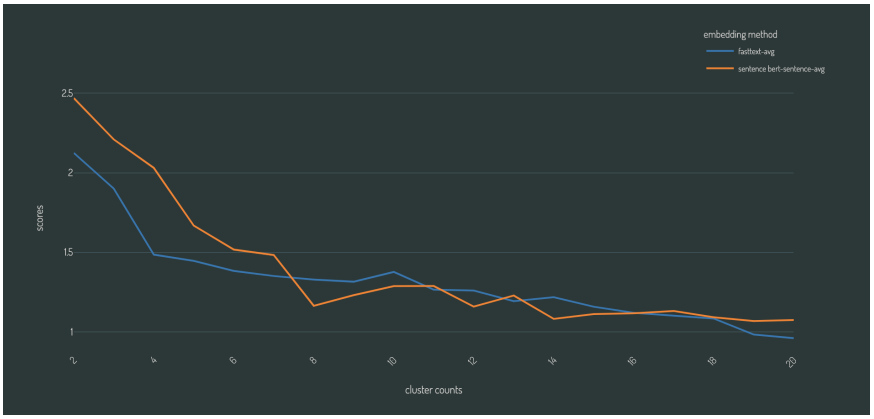

Figure 6.87: Best Davies-Bouldin Index Line Graph of Best Embedding Methods, K-Means Clustering, Cappadocia Dataset

In contrary to results with best silhouette score , there is a difference between results of Agglomerative and K-Means cluster with best Davies-Bouldin Index. Also, Agglomerative ones predict the clusters to be better as the cluster count increases, the K-Means predictst the opposite. The stability is also less, so this is a possible sign of low quality of clusters.

Table 6.24: Best Rand Index Results of Best Embedding Methods, Agglomerative Clustering, Cappadocia Dataset

Method	**Option**	**7 Clusters**
fasttext	avg	0.8557
sentence bert	sentence-avg	0.7458

Table 6.25: Best Rand Index Results of Best Embedding Methods, K-Means Clustering, Cappadocia Dataset

Method	**Option**	**7 Clusters**
fasttext	avg	0.8493
sentence bert	sentence-avg	0.7596

Even if there are some medium, low results in and differences between the clustering

results in terms of the best silhouette score and davies-bouldin index, the best rand indices for both Agglomerative and K-Means clustering methods are similar and very high (Table 6.24 and 6.25). This means that the scatter plots of these methods will be very close to the ideal.

Table 6.26: Best Homogeneity Score Results of Best Embedding Methods, Agglomerative Clustering, Cappadocia Dataset

Method	**Option**	**7 Clusters**
fasttext	avg	0.7850
sentence bert	sentence-avg	0.7430

Table 6.27: Best Homogeneity Score Results of Best Embedding Methods, K-Means Clustering, Cappadocia Dataset

Method	**Option**	**7 Clusters**
fasttext	avg	0.8252
sentence bert	sentence-avg	0.7664

Homogeneity scores are not as high as the rand index values but they are also close to 1. K-Means Clustering show slightly better performance while predicting the clusters according to homogeneity score. So, the homogeneity results are also expected to be good enough for this data set.

All the scatter plots are drawn as the perplexity is 10 and cluster count is 4. The reason is that there are few papers to work on, the cluster count required to be small and the 4 is the count of labels and the constant cluster counts of rand and homogeneity scores. When the cluster count is 4, perplexity would be fair to be 10 since it is the approximate measure of neighbors of a data point.

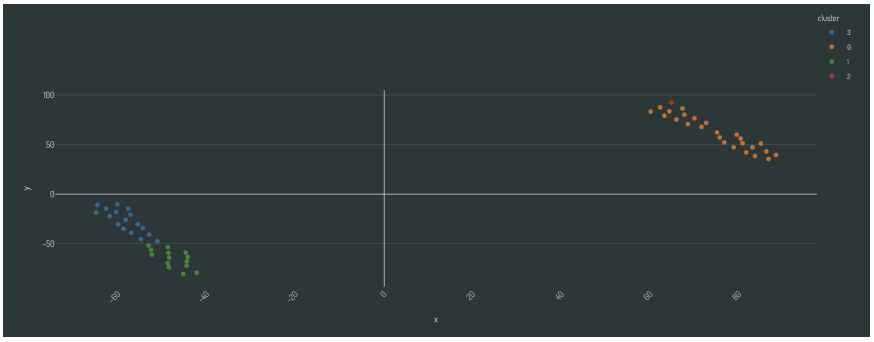

Figure 6.88: TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 10

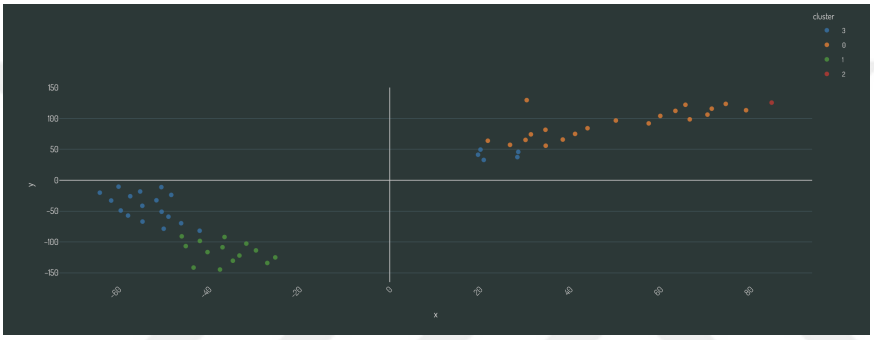

Figure 6.89: TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 10

Fasttext with best silhouette score is expected to distinguish data into two clusters if the visualisation in Figure 6.88 and Figure 6.89 are taken into consideration. For the four clusters, data cannot be distinguished enough.

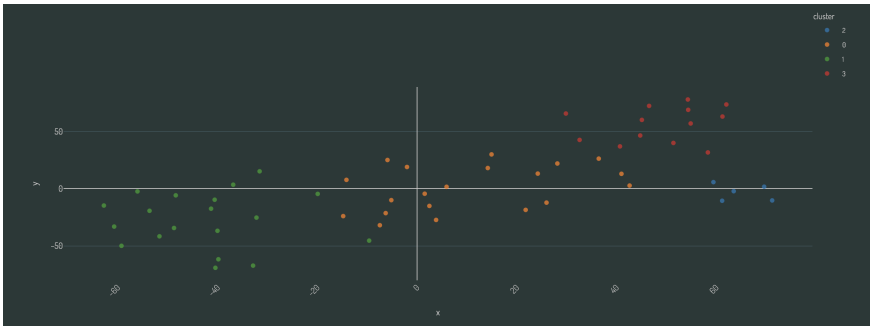

Figure 6.90: TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Davies-Bouldin Index Strategy and Agglomerative Clustering Method, perplexity: 10

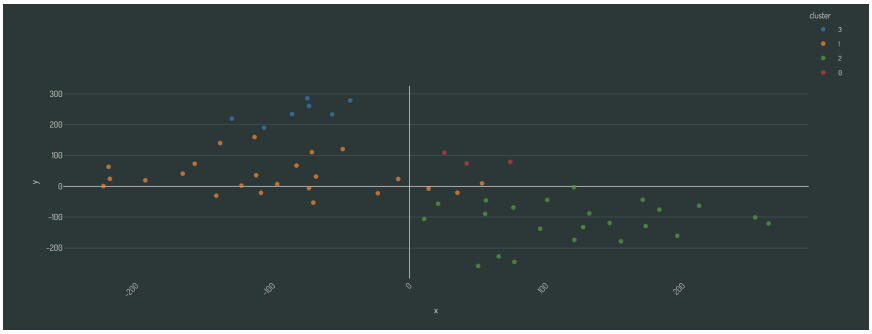

Figure 6.91: TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Davies-Bouldin Index Strategy and K-Means Clustering Method, perplexity: 10

Fasttext with best Davies-Bouldin Index results seem to be noisier where their silhouette scores over different cluster counts are not stable. However, the clusters are not inter-twined much. So, the result is slightly good.

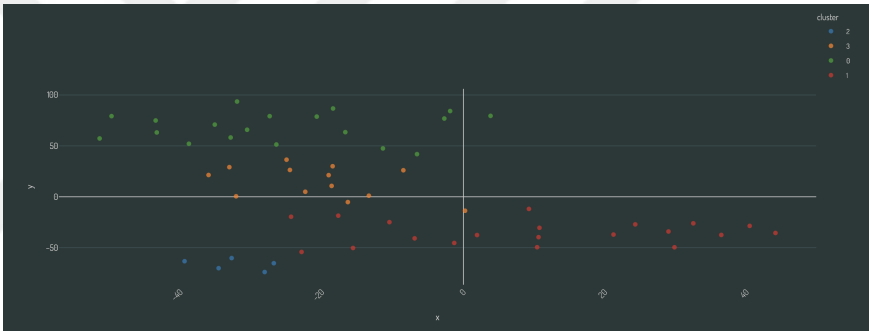

Figure 6.92: TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 10

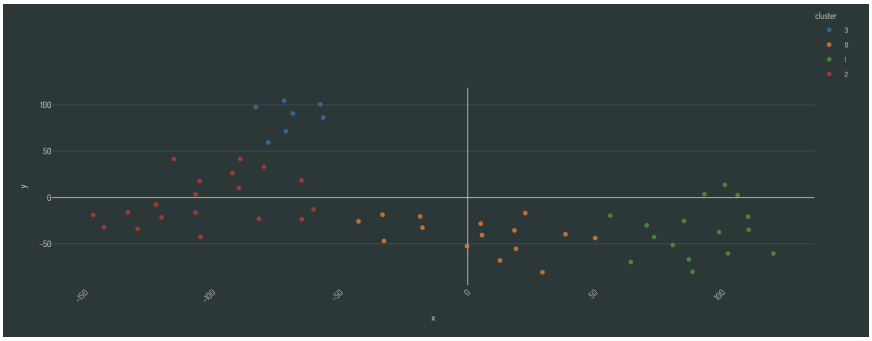

Figure 6.93: TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 10

It is said that, rand index for given labeled papers is similar to the ideal, since the accuracy is very high. It predicts clusters that are well separated, equally distributed but a little bit noisy and not dense. This may be because of the structure of data set since the ideal is expected to be similar to this. These results give us a good intuition about the distribution of documents in data set. The same idea applies both for K-Means and Agglomerative clustering. The both expect similar structures and their scores are very close to each other as well.

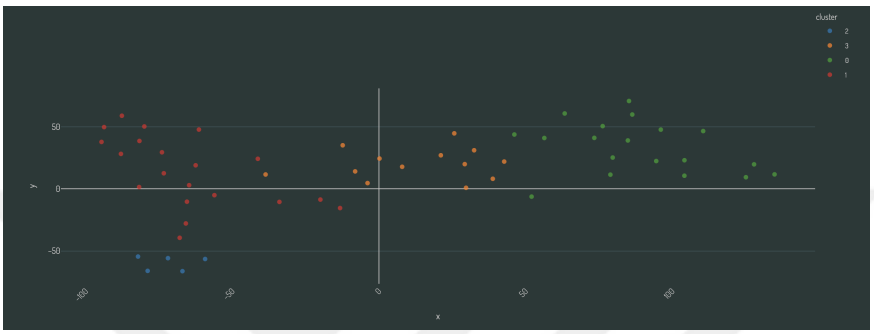

Figure 6.94: TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Homogeneity Score Strategy and Agglomerative Clustering Method, perplexity: 10

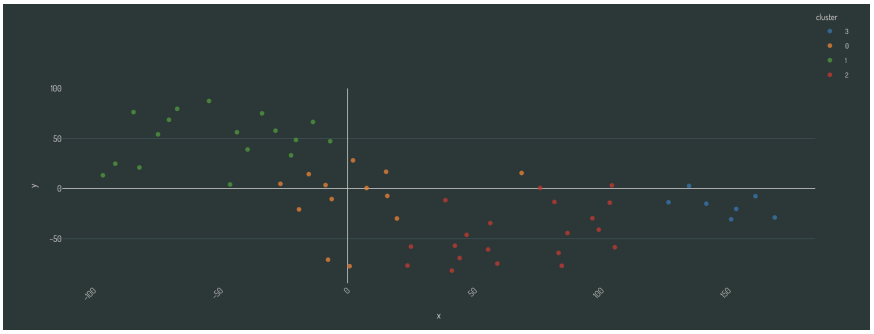

Figure 6.95: TSNE Scatter Plot of 4 clusters With Fasttext Averaging Having Best Homogeneity Score Strategy and K-Means Clustering Method, perplexity: 10

Homogeneity scores are lower than the rand indices for clustering methods. But the scatter plots show similar structure: non-dense but well-separated, not intertwined clusters.

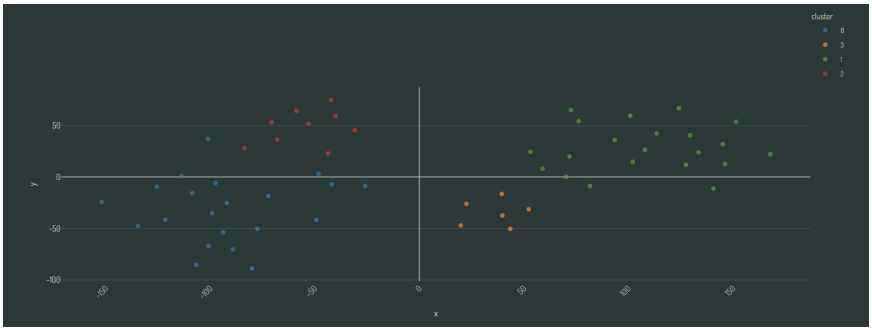

Figure 6.96: TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 10

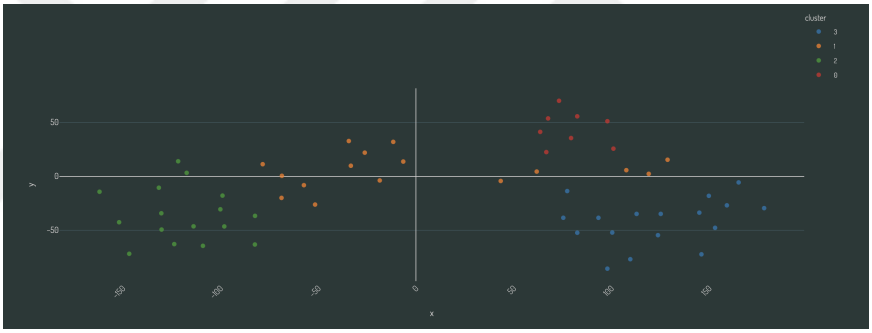

Figure 6.97: TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 10

Sentence BERT results with best silhouette score and Agglomerative Clustering have a clustering result similar to the one with Fasttext and best rand index value. So, the reason silhouette score is low for Sentence BERT result is because noise increase the silhouette score but the clusters are well-separated and intertwined data points do not exist. K-Means results visualisation is not as good as Agglomerative Clustering result in terms of silhouette score.

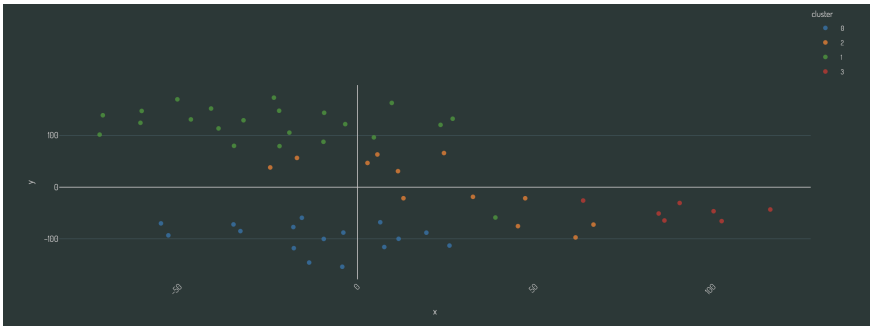

Figure 6.98: TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Davies-Bouldin Index Strategy and Agglomerative Clustering Method, perplexity: 10

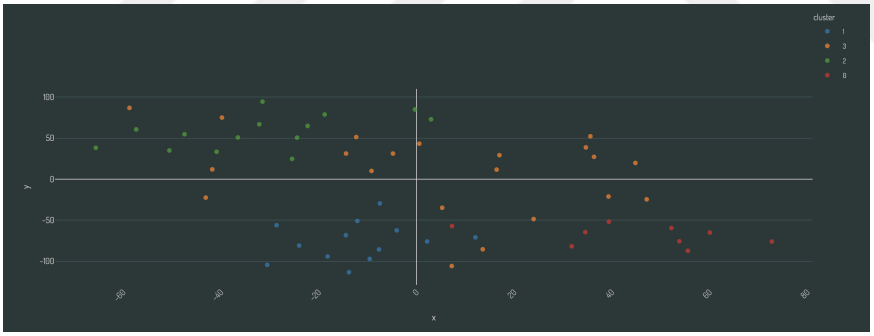

Figure 6.99: TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Davies-Bouldin Index Strategy and K-Means Clustering Method, perplexity: 10

Davies-Bouldin Index results do both seem noisy and do not have well-separated clusters even if the scores seem better.

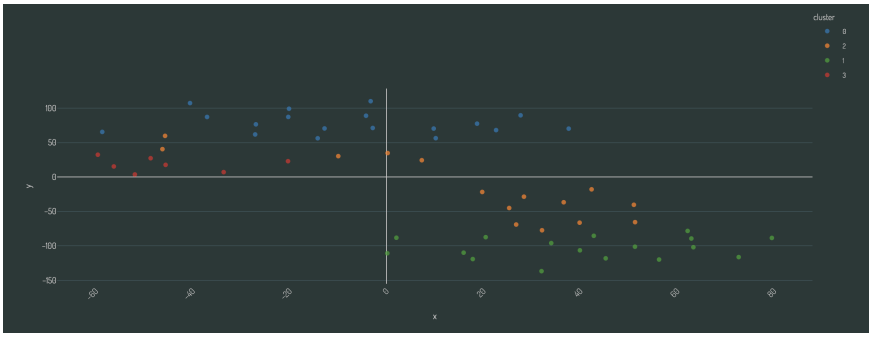

Figure 6.100: TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Rand Index Strategy and Agglomerative Clustering Method, perplexity: 10

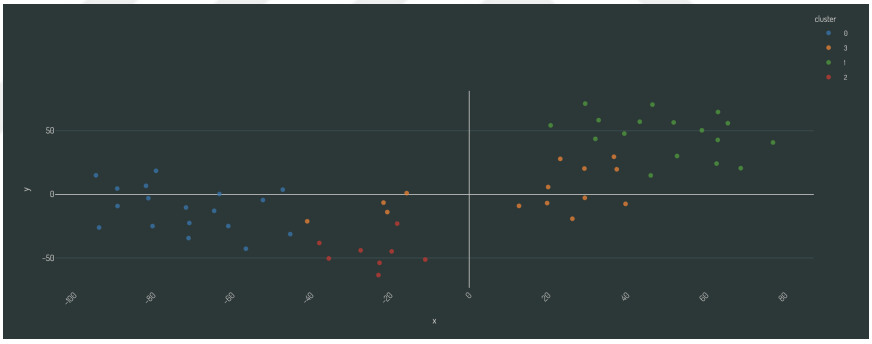

Figure 6.101: TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Rand Index Strategy and K-Means Clustering Method, perplexity: 10

Rand index values of Sentence BERT results are lower than the ones of Fasttext. So, one can expect lower quality but the value of the score is high. The yellow cluster has points inside the border of other clusters. This can be the points that decrease the rand indices.

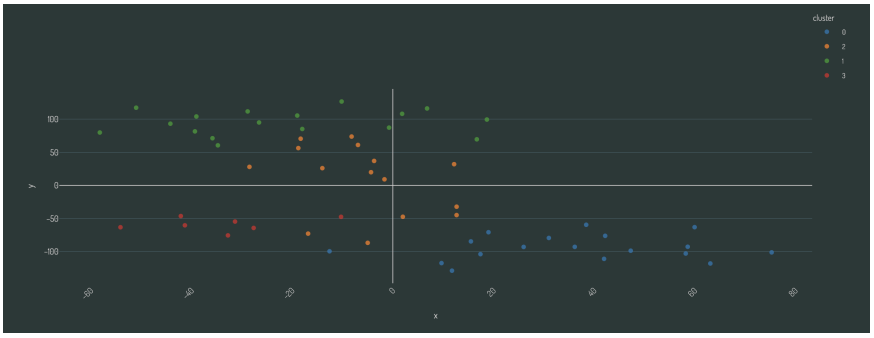

Figure 6.102: TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Homogeneity Score Strategy and Agglomerative Clustering Method, perplexity: 10

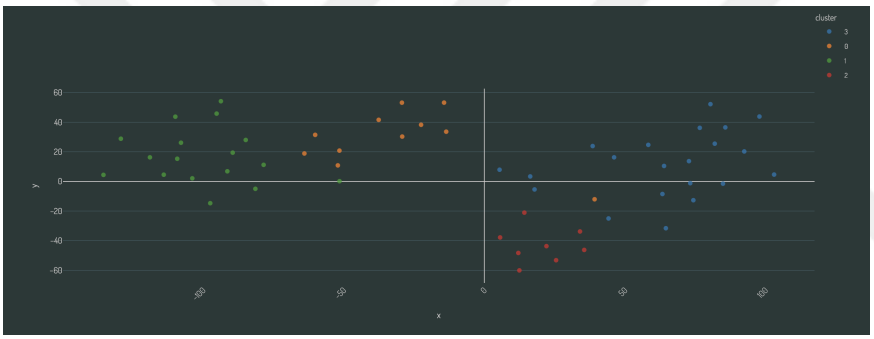

Figure 6.103: TSNE Scatter Plot of 4 clusters With Sentence BERT Sentence Averaging Having Best Homogeneity Score Strategy and K-Means Clustering Method, perplexity: 10

Homogeneity scores are also lower and Fasttext shows a better distribution than Sentence BERT, so the Fasttext is better in visualisation and scores.

About the data, it can be said that there is a noisy data that cannot be clustered perfectly. The methods do very well in terms of rand index and homogeneity scores which are the accuracy measures for labeled papers. Lastly, the methods work on the smaller data set and give better accuracy while do not give good enough clusters visually.

Table 6.28: Improvements with Optimizations with Cappadocia Dataset

Method	**Method Option**	**Clustering Method**	**Avg of Avg Scores**	**Best of Avg Scores**	**Best of Opt Avg Scores**	**Rise1(%)**	**Rise2(%)**
fasttext	avg	Agglomerative	0.3129	0.591	0.611	88.851	3.3782
fasttext	**avg**	**K-Means**	**0.3112**	**0.5984**	**0.607**	**92.3015**	**1.4265**
sentence bert	sentence-avg	Agglomerative	0.1222	0.1342	0.1472	9.8203	9.6575
sentence bert	sentence-avg	K-Means	0.1073	0.1243	0.1437	15.894	15.578

The improvements of according to averages in Fasttext is higher as in the Multidisciplinary Papers dataset. However, the increase with the clustering method parameters are not that high. This can be because the limit is achieved in terms of silhouette score. Sentence BERT silhouette scores are lower than then the ones in Fasttext and improvements are also lower according to hyper-parameters but the clustering parameters make a better improvement than Fasttext. But the scores of Sentence BERT is very low and possibility of having coincidental increases are also high. In overall, the framework we built has increased the silhouette scores of methods with at least 20% and at most 95%.

6.2.2 COVID19 Papers Data set

COVID19 Dataset is large dataset having abstract of over 30.000 papers used for further analysis of best methods chosen: fasttext and sentence bert. There are no labels, so only the silhouette and davies-bouldin indices are used as a metric. Also, in the previous trials, only less than ten clusters are analysed, so until 10 clusters the trials are run in this part.

Table 6.29: Best Silhouette Score Results of Best Embedding Methods, Agglomerative Clustering, Covid19 Dataset

Method	Option	2 Clusters	5 Clusters	8 Clusters	10 Clusters
fasttext	avg	0.8486	0.7104	0.6811	0.6803
sentence bert	sentence-avg	0.3574	0.2979	0.2735	0.2644

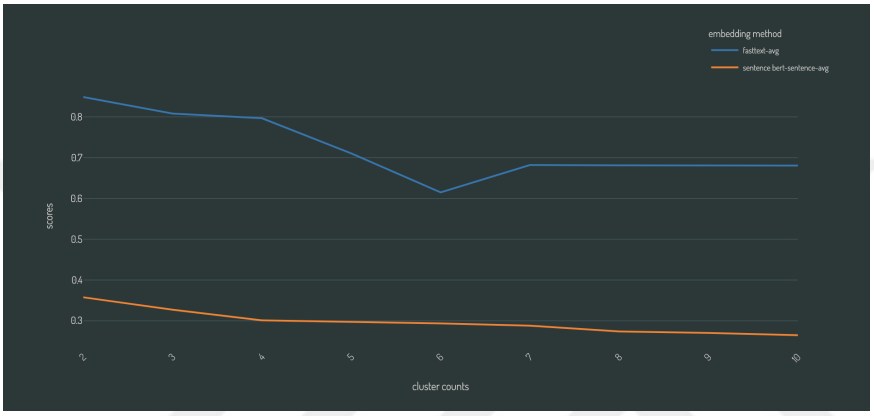

Figure 6.104: Best Silhouette Score Results of Best Embedding Methods, Agglomerative Clustering, Covid19 Dataset

Fasttext has a rather good performance on Covid19 dataset with best silhouette score and Agglomerative CLustering as in the other data sets. The documents are expected to be well clustered enough. On the other hand, Sentence BERT has relatively bad performance but the silhouette score over the cluster counts is more stable. So, we can expect less outliers but more intertwined clusters with Sentence BERT.

Table 6.30: Best Silhouette Score Results of Best Embedding Methods, K-Means Clustering, Covid19 Dataset

Method	Option	2 Clusters	5 Clusters	8 Clusters	10 Clusters
fasttext	avg	0.2094	0.2251	0.1657	0.1367
sentence bert	sentence-avg	0.1321	0.0691	0.0505	0.0508

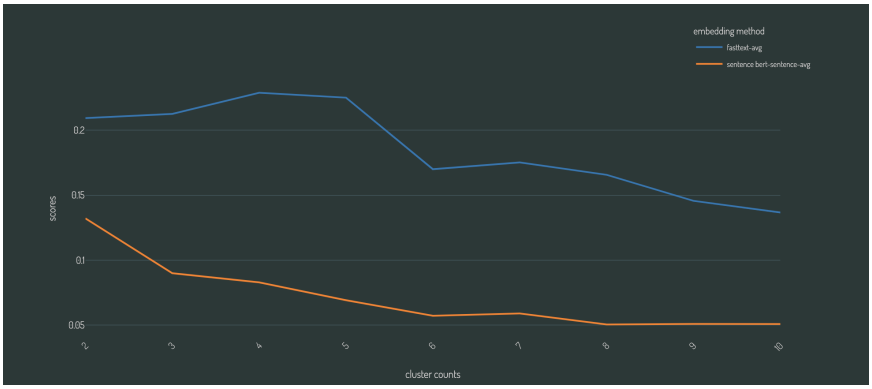

Figure 6.105: Best Silhouette Score Results of Best Embedding Methods, K-Means Clustering, Covid19 Dataset

Both methods, Fasttext and Sentence BERT- with best silhouette score and Agglomerative Clustering has a bad performance. This shows that K-Means clustering may not be split.

Table 6.31: Best Davies-Bouldin Index Results of Best Embedding Methods, Agglomerative Clustering, Covid19 Dataset

Method	**Option**	**2 Clusters**	**5 Clusters**	**8 Clusters**	**10 Clusters**
fasttext	avg	0.1780	0.3360	0.3190	0.3521
sentence bert	sentence-avg	1.4736	0.5944	0.5955	0.6032

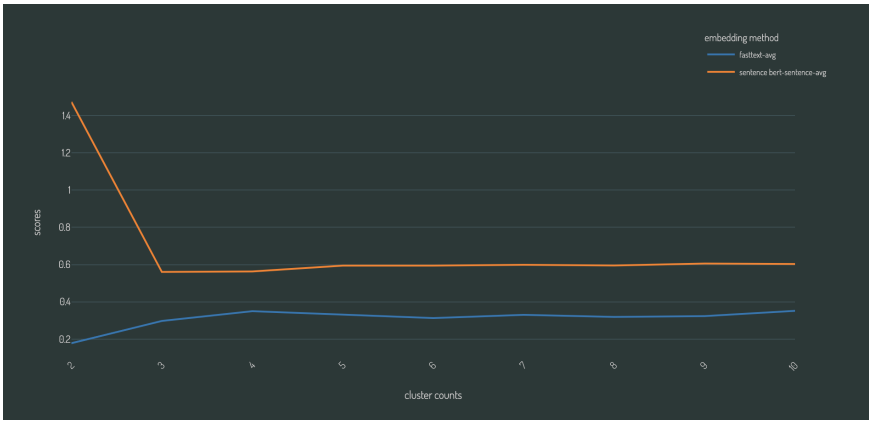

Figure 6.106: Best Silhouette Score Results of Best Embedding Methods, Agglomerative Clustering, Covid19 Dataset

The results with best Davies-Bouldin index with Agglomerative Clustering, Sentence BERT creates good scores and the Fasttext has better results. It is better than both datasets for Fasttext. So, results with best DB index of Fasttext can be thought as the best clustering result for trials of this dataset.

Table 6.32: Best Davies-Bouldin Index Results of Best Embedding Methods, K-Means Clustering, Covid19 Dataset

Method	**Option**	**2 Clusters**	**5 Clusters**	**8 Clusters**	**10 Clusters**
fasttext	avg	2.0753	0.2.387	2.3470	2.0964
sentence bert	sentence-avg	3.1720	4.3118	4.1425	4.1814

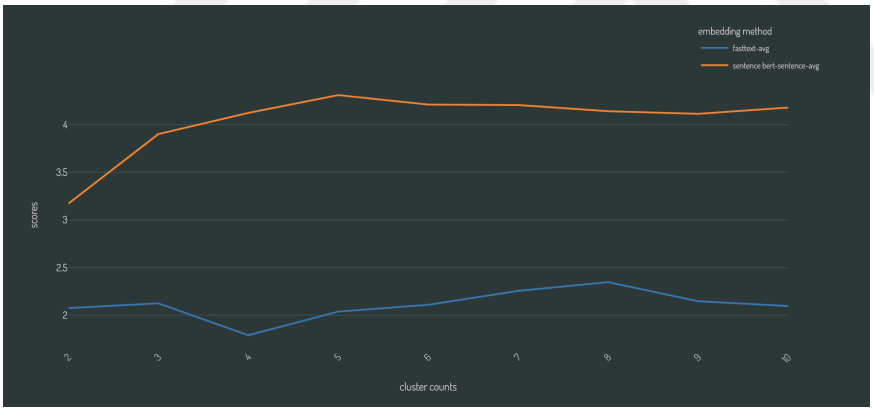

Figure 6.107: Best Silhouette Score Results of Best Embedding Methods, K-Means Clustering, Covid19 Dataset

The results with best Davies-Bouldin index with K-Means Clustering have rather worse results. It can be said that K-Means do not separate the data well as in the best silhouette score. Also, the data is fluctuating slightly, so there can be some outlier points as well.

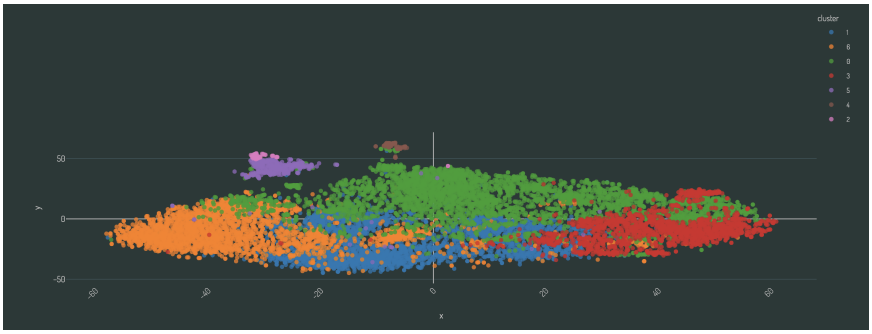

Figure 6.108: TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 50

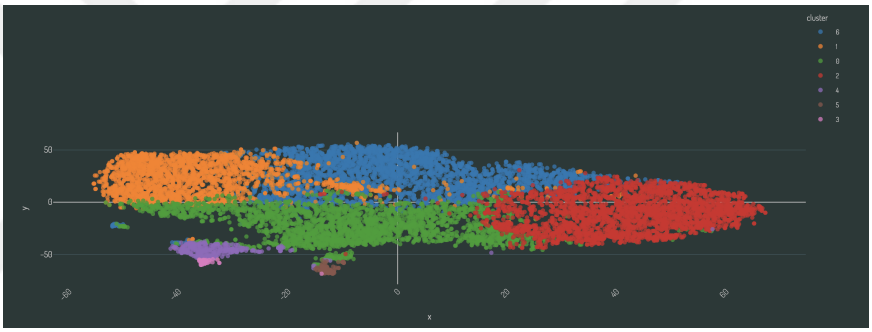

Figure 6.109: TSNE Scatter Plot of 7 clusters With Fasttext Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 50

Fasttext method with best silhouette score has better silhouette scores with Agglomerative clustering than K-Means clustering. However, according to visualisation, there are outliers and lots of points seen at the area of other clusters. The outliers may have increased the score but the silhouette scores do not change much over cluster counts. Also, no more outliers in the Agglomerative Clustering are seen than K-Means clustering. Data visualisation may be not so close to actual data in higher dimension, as well. This anomaly can be solved by a labelled indices but there is no label for documents in this dataset to use.

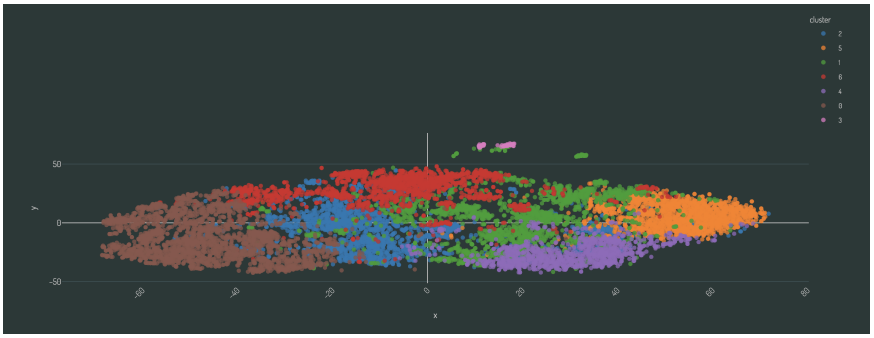

Figure 6.110: TSNE Scatter Plot of 7 clusters With Fasttext Averaging
Having Best Davies-Bouldin Index Strategy and Agglomerative Clustering Method, perplexity: 50

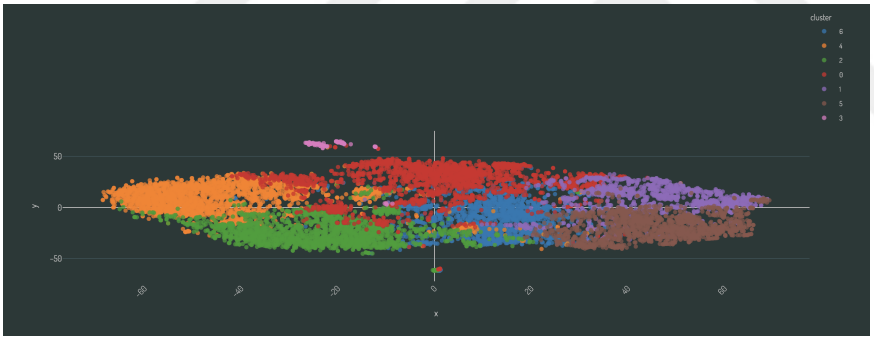

Figure 6.111: TSNE Scatter Plot of 7 clusters With Fasttext Averaging
Having Best Davies-Bouldin Index Strategy and K-Means Clustering Method, perplexity: 50

Davies-Bouldin Index creates more separated clusters visually and its performance for Fasttext is very good for Agglomerative Clustering but the noise inside clusters is more than the ones with best silhouette score. K-Means also has similar structure but more smooth. So, DB-Index creates more separated clusters for Fasttext.

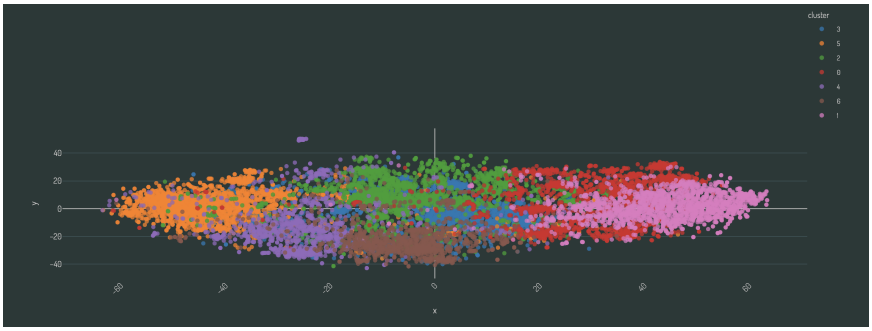

Figure 6.112: TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Silhouette Score Strategy and Agglomerative Clustering Method, perplexity: 50

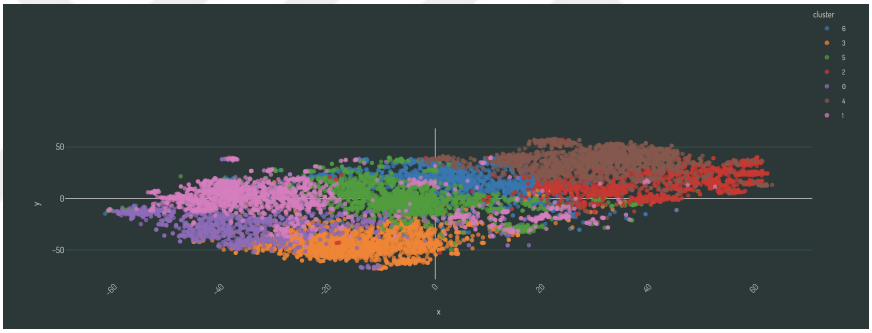

Figure 6.113: TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Silhouette Score Strategy and K-Means Clustering Method, perplexity: 50

Sentence BERT creates mostly noisy clusters for silhouette score and both clustering methods. Actually, the silhouette score is low for the Sentence BERT for both the Agglomerative and K-Means clustering methods. This shows that Sentence BERT does not predict very well-clustered clusters. However, in other data sets, the separation was not as good as Fasttext for Sentence BERT but the labeled results, or accuracy was so good that it is the best or the closer to the best.

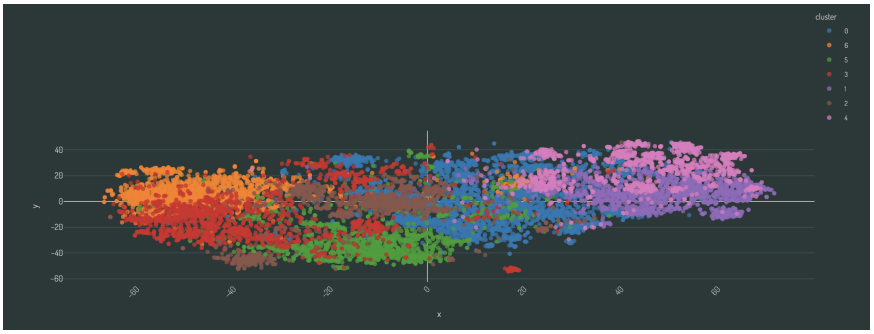

Figure 6.114: TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Davies-Bouldin Index Score Strategy and Agglomerative Clustering Method, perplexity: 50

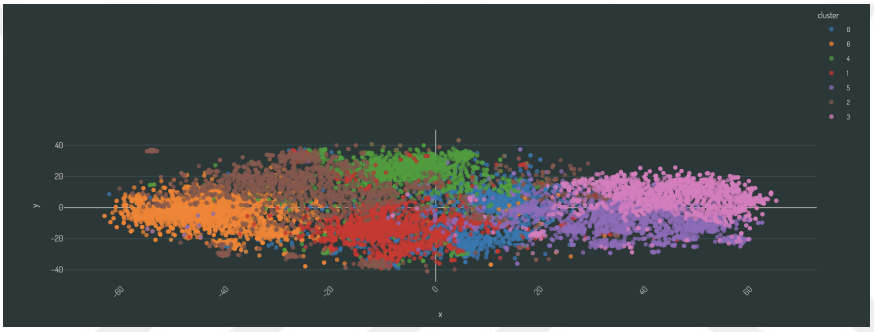

Figure 6.115: TSNE Scatter Plot of 7 clusters With Sentence BERT Sentence Averaging Having Best Davies-Bouldin Index Score Strategy and K-Means Clustering Method, perplexity: 50

The results with best Davies-Bouldin score is noisier. K-Means decreases this noise but this time, clusters are not separated well.

As a result, the method proposed in this thesis worked on a large dataset, a medium data set and a small data set. All the datasets contain papers in some context. Similar results are obtained such that the documents are clustered to some extent but it is sufficient to have good enough clustering results that is tested with labeled clustering metrics (rand index and homogeneity score).

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

A collection of papers of academicians from Cost Action CA18110 project is a starting point for this thesis. It is analysed so that one can get a good understanding even if the reader does not have much knowledge about the collection. In order to achieve that, text representation techniques, clustering methods and clustering validity indices are utilized and the hyper-paramaters of text representation techniques and parameters of clustering methods are optimized according to clustering validity index. Also, the results are evaluated in detail and best results are tested with other similar data sets. As a result, several outcomes are obtained.

- Clustering quality is increased with the hyper-parameter optimization of embedding methods and optimization of clustering method parameters have increased the clustering quality, which is measured with silhouette score. Even if there are not so many trials, improvements have been more than 200% for some cases. This means that the clustering quality can be increased with parameter tuning with a few trials -around 20-.

- Secondly, clustering measures are evaluated in terms of their performance to measure clustering quality by comparing the results of different clustering metrics with each other. As an example, searching over silhouette score improves the clustering quality but this may not always mean the clustering accuracy according to ground truth determined by humans. However, searching for a well separated, dense clusters via silhouette score helps us approximate better accuracy even if the best results of both are not exactly equal. This is found by comparing silhouette scores with rand index and homogeneity scores. Also, the silhouette score and Davies-Bouldin score, which both do not require ground

truth, are compared according to the visualisation results. For example, the scores are compared in terms of their deviation in the case of outliers.

- Performances of embedding methods are measured individually for the task of clustering with the help of different clustering methods. They are compared in terms of different metrics in order to detect the best algorithm. As mentioned, the metrics are also analysed in terms of their strengths while doing that selection. The chosen best methods are tested with other two data sets as well.
- Embedding methods are evaluated in terms of how their average silhouette score -which is the objective function of optimization methods- increased. The rise of the score over different trials for each embedding and clustering method is analysed and a good improvement for some methods is observed. The more hyper-parameters method has, the more increase in the score has been achieved (Fasttext vs BERT and BERT vs SciBERT).
- The documents are evaluated individually by looking at the visual support from the tool. It has been observed that the papers from one writer had similar relationships, clusters are formed after enough trials with the right method. As an example, the main dataset of the thesis -a collection of papers of academicians from Cost Action CA18110 - have distinguished the electric, computer science fields from anthropology and archaeology. Also, Cappadocia papers have two main clusters that one has the papers about tourism and the other has the ones with history, archaeology and geology of the region. These good results helped detect the ground truth labels.
- Data sets having different sizes and being from different contexts are observed and the changes in the clustering quality and accuracy are observed. Being scientific papers related to some general topic and from different areas of interest is in common in data sets. So, their inseparability issues are analysed and tried to be solved.

For a further study, the following extensions and changes can be applied:

- Document embeddings can be created via more wise combination strategies rather than averaging or weighted averages of less text chunks such as words.

- Using abstraction methods, a common topic can be constructed for documents in each cluster. In this way, clusters can be more meaningful.
- tf-idf could be used as a similarity measure instead of a combination strategy.
- A clustering method only related to document clustering could be utilized[20].
- Ground truth labels could be used for a meta-learning and the objective function could be constructed in this way for optimization.

Using the experiment tool, the experiments are automatized and results are reproducible, where the actual results are preserved as well. So, one can re-apply the experiments in this thesis and more by using the tool, or they can reconstruct the results by using data sets in this paper.

REFERENCES

[1] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[4] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," *Advances in neural information processing systems*, vol. 28, 2015.

[5] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations. corr abs/1802.05365 (2018)," *arXiv preprint arXiv:1802.05365*, 1802.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[7] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, pp. 1188–1196, PMLR, 2014.

[8] K. Premalatha and A. Natarajan, "A literature review on document clustering," *Information Technology Journal*, vol. 9, no. 5, pp. 993–1002, 2010.

[9] P. Willett, "Recent trends in hierarchic document clustering: a critical review," *Information processing & management*, vol. 24, no. 5, pp. 577–597, 1988.

[10] R. Bekkerman, H. Raghavan, J. Allan, and K. Eguchi, "Interactive clustering of text collections according to a user-specified criterion.," in *IJCAI*, vol. 7, pp. 684–689, 2007.

[11] R. Bekkerman and M. Sahami, "Semi-supervised clustering using combinatorial mrfs," in *ICML-06 Workshop on Learning in Structured Output Spaces*, 2006.

[12] S. Basu, M. Bilenko, and R. J. Mooney, "A probabilistic framework for semi-supervised clustering," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 59–68, 2004.

[13] B. Liu, X. Li, W. S. Lee, and P. S. Yu, "Text classification by labeling words," in *Aaai*, vol. 4, pp. 425–430, 2004.

[14] R. Bekkerman, R. El-Yaniv, and A. McCallum, "Multi-way distributional clustering via pairwise interactions," in *Proceedings of the 22nd international conference on Machine learning*, pp. 41–48, 2005.

[15] T. Wei, Y. Lu, H. Chang, Q. Zhou, and X. Bao, "A semantic approach for text clustering using wordnet and lexical chains," *Expert Systems with Applications*, vol. 42, no. 4, pp. 2264–2275, 2015.

[16] A. Islam, E. Milios, and V. Kešelj, "Text similarity using google tri-grams," in *Canadian Conference on Artificial Intelligence*, pp. 312–317, Springer, 2012.

[17] V. H. A. Soares, R. J. Campello, S. Nourashrafeddin, E. Milios, and M. C. Naldi, "Combining semantic and term frequency similarities for text clustering," *Knowledge and Information Systems*, vol. 61, no. 3, pp. 1485–1516, 2019.

[18] T. Walkowiak and M. Gniewkowski, "Evaluation of vector embedding models in clustering of text documents," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pp. 1304–1311, 2019.

[19] A. K. Singh and M. Shashi, "Vectorization of text documents for identifying unifiable news articles," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 7, 2019.

[20] V. Mehta, S. Bawa, and J. Singh, "Weclustering: word embeddings based text clustering technique for large datasets," *Complex & intelligent systems*, vol. 7, no. 6, pp. 3211–3224, 2021.

[21] H. Wang, C. Zhou, and L. Li, "Design and application of a text clustering algorithm based on parallelized k-means clustering.," *Rev. d'Intelligence Artif.*, vol. 33, no. 6, pp. 453–460, 2019.

[22] M. M. Fard, T. Thonet, and E. Gaussier, "Deep k-means: Jointly clustering with k-means and learning representations," *Pattern Recognition Letters*, vol. 138, pp. 185–192, 2020.

[23] L. Abualigah, A. H. Gandomi, M. A. Elaziz, A. G. Hussien, A. M. Khasawneh, M. Alshinwan, and E. H. Houssein, "Nature-inspired optimization algorithms for text document clustering—a comprehensive analysis," *Algorithms*, vol. 13, no. 12, p. 345, 2020.

[24] Y. Poulakis, C. Doulkeridis, and D. Kyriazis, "Autoclust: A framework for automated clustering based on cluster validity indices," in *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 1220–1225, IEEE, 2020.

[25] D. G. Ferrari and L. N. De Castro, "Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods," *Information Sciences*, vol. 301, pp. 181–194, 2015.

[26] G. K. Zipf, "Selected studies of the principle of relative frequency in language," in *Selected Studies of the Principle of Relative Frequency in Language*, Harvard university press, 2013.

[27] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[28] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[29] A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[30] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[31] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[33] T. Mikolov, J. Kopecky, L. Burget, O. Glembek, *et al.*, "Neural network based language models for highly inflective languages," in *2009 IEEE international conference on acoustics, speech and signal processing*, pp. 4725–4728, IEEE, 2009.

[34] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[35] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, "Improving language understanding by generative pre-training," 2018.

[36] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[37] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[38] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[39] J. Han, J. Pei, and H. Tong, *Data mining: concepts and techniques*. Morgan kaufmann, 2022.

[40] J. Bergstra, D. Yamins, D. D. Cox, *et al.*, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," in *Proceedings of the 12th Python in science conference*, vol. 13, p. 20, Citeseer, 2013.