Deep Language Model Representation of Document Clustering

Shakhboz Abdulazizov
B00870379
Dalhousie University
Halifax NS, Canada
sh873370@dal.ca

Bhuvaneshwari Basquarane
B00853122
Dalhousie University
Halifax NS, Canada
bh563857@dal.ca

Rakshit Makan
B00883651
Dalhousie University
Halifax NS, Canada
r.makan@dal.ca

Abstract—**Document clustering is an important problem in Machine Learning which combined with NLP language models can be useful in exploratory analysis of document corpora. In this project, we try to understand how different clustering algorithms in combination with dimensionality reduction and a state-of-the-art Deep Language Model over a text corpus taken from the Abstract of research papers in a domain of machine learning, natural language processing and deep language models. In this project, we are comparing the clustering algorithms with SBERT Embeddings and evaluate the result based on the clusters of the documents.**

1. Introduction

As the computing has already become inevitable part of daily life, the amount of digital data in form of documents is increasing massively. This trend led the problem of finding desired document easily and fast among huge number of available data. Although the documents are in natural language which is easy for human understanding, when the information reaches a critical mass, the time of data clustering and selecting is very high. However, state of art deep language models which can understand textual data in contextual manner similar to human brain combined with appropriate clustering algorithms could achieve document clustering accuracy close to human performance.

In this project we used deep language model SBERT [1] to produce contextualized word embeddings in order to cluster the documents with existing clustering algorithms like KMeans, HDBScan, and DBScan. This project also present how SBERT pretrained model has been a better encoder of documents than BERT base and its various pooling schemes.

The data-set which we are trying to utilize is a collection of research papers from the MALNIS lab library. The data format is of CSV that consists of the following headers: Bibliography Type, Identifier, Author, Title, Journal, Month, Year, URL, Abstract. The collection holds one thousand papers in the area of machine learning, natural language processing and deep language models. The idea behind this approach is, we will use deep language models specifically to process our text corpus from the different research papers abstract section and project them onto a vector embedded space.

Clustering the documents was the main goal for this project. In our investigation we found that the density based clustering algorithms works better than early centroid based. HDBScan[2] is one of the density based clustering algorithm we tested which was developed by Campello, Moulavi, and Sander. Along with this algorithm we ran few iterations with KMeans and DBScan clustering. We experimented mainly with the BERT base (768 hidden Layer), which produces a high dimension dense vector for the document.

Before passing these vector through clustering algorithm we added a dimensionality reduction layer. In this project we tested principle component analysis (PCA)][3] with KMeans and Used Uniform Manifold Approximation and Projection (UMAP)[4] with DBScan and HDBSCan. This project majorly cover the performances of the clustering algorithms with respect to the Deep language models and represent results in form of user study and the visual 2-D clusters using dimentionality reduction technique.

The following sections will provide the Background, Methodology, and Results of the investigation in detail. We will then briefly analyze these results, and the study as a whole in the Discussion section.

2. Background

2.1. Language Modelling

Language models are a broad set of functions in natural language processes which provide solutions to various language related problems. These problems can include text prediction, translations, and audio to text conversion. In the scope of this paper, language models will be used to represent text documents as word embeddings, this process is also known as topic modelling. Some earlier topic modelling techniques will be described before discussing the chosen models, in order to gain insight of the evolution of such models.

2.2. Topic Modelling

An important discussion piece throughout a topic modelling publication by Luhn as early as 1958 was that machines only ever understand words as pure physical objects [5]. Machines are unable to map meaning to words, only count them. This gives rise to an important fundamental concept through many natural language processing models; a machine must learn to handle and process text through numerical representations.

In a summary for topic models by Nenkova & McKeown, they define topic word models as finding significant words within documents which may be used to represent the document as a whole [6].

Luhn also provided the idea that each word in a document has some significance, and this can be measured by the word's presence throughout the document [5]. The presence of an individual word in a document can be measured by its frequency.

Dunning was later able to create a more robust topic model based on the ideas from Luhn [7]. Although Luhn was able to provide this foundation, it was found that significant words were not necessarily common words, and by missing those, the frequency models may not find an accurate summary of the analyzed document. Dunning's model involved statistical analysis to find significance in less common words which the previous models may have missed [7]. This introduces one class of topic models, known as frequency driven topic models.

2.3. Frequency Driven Topic Models

Frequency driven models assign weights to words in order to derive representations that best summarize a given document [8]. Words with greater weights are considered more important while summarizing the document. Stop words are once again removed for these models, as they are common yet do not provide much meaning to the overall topic of the document.

Early frequency models were flawed as they had no way of compensating for stop words other than completely removing them. This expands not to just stop words, but also any other word in the analyzed documents which were common, but did not effectively contribute to the overall meaning of the document. Thus a more robust frequency driven model was developed, known as TF-IDF [8]. This model will be briefly discussed.

2.3.1. TFIDF Model. TF-IDF is a robust frequency driven topic model as it is able to compensate for the previously discussed common word problem [9] TF-IDF determines the weights of words based on their frequency throughout a document. As mentioned, the goal of this is to find important words which can be used to summarize the given text. TF-IDF introduces a punishment to weights of words that are common across the corpus being investigates [9]. This means that commonly occurring words, such as stop words will not have relatively lower weights, despite having a high frequency throughout a given document. Overall, TF-IDF is able to assign weights to words according to their importance, and these important words can be used to represent the given document as a whole [9].

2.4. Attention Models

Attention models take a different route of deriving text representation. There is a deep history which will be briefly discussed in order to introduce the Transformer model. A key component of most attention models is the generation of word embeddings which contain more context to the processed text data. In theory, contextualized word embeddings would better represent text compared to models which summarize text simply by its frequency. This is because frequency does not necessarily capture the meaning or context of most text.

The attention model which is most relevant to this study is known as the Transformer. The Google Brain publication known as Attention is All You Need presents this model [10]. Originally proposed to solve an inefficiency observed in recurrent neural networks during sequential text processing, the Transformer has become a foundation for many attention models due to its encoder stack [10]. Though only one part of the model is used, the encoder stack, contains a multi-head self-attention layer, which contributes to its ability to generate contextual word embeddings based on input text [10]. This model plays an important role in the BERT model, as will be discussed throughout the paper.

2.5. BERT

BERT, or Bidirectional Encoder Representations from Transformers, is a model highlighted in a recent paper published by researchers at Google AI Language [10]. It caused a stir in the Machine Learning community by presenting state of the art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others.

BERT's key technical innovation is applying the bidirectional training of a Transformer to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training.

2.6. SBERT

SBERT[1] is a modification of the pretrained BERT network that uses siamese and triplet network structures to derive semantically meaningful sentence embeddings. These embeddings can be compared using cosine-similarity methods, in order to organize the corpus.

2.7. K-Means

Kmeans being one of the most widely known clustering algorithm, it is fast and the simplest unsupervised learning algorithms which solve the well known clustering problem. The process follows a way to classify a given data set through a certain number of clusters (assume k clusters). The main idea is to define k centers, one for each cluster. However it is a partitioning algorithm rather than clustering as it partitions the data-set into as many parts as requested through trying to minimize inter-partition distances [11]. The algorithm works in the way to process the training data, the pre-processed data-set starts with a first group of randomly selected centroids, which are used as the starting points for every cluster, and then performs iterative calculations to optimize the positions of the centroids.

2.8. DBSCAN

DBScan is a density based algorithm and considers dense regions as clusters. DBSCAN groups together the points that are closely packed together, assuming points that lie in low density regions as ’noise’. It needs a minimum cluster size and a distance threshold epsilon as user-defined input parameters. To start with the algorithm, it requires two parameters which are ϵ (eps) and the minimum number of points required to form a dense space ’min distance’. The process begins with an arbitrary starting point that is neither been visited nor defined before. This point is ϵ-neighborhood which is discovered as output, and if it contains significantly many points, a cluster is started. Otherwise, the point is labeled as noise. Consider this point might later be found in a sufficiently sized ϵ-environment of a different point and therefore made part of a cluster. If a point is found to be a dense part of a cluster, its ϵ-neighborhood is also part of that cluster. Thus, all points that are found within the ϵ-neighborhood are appended, as is their own ϵ-neighborhood when they are also dense. This process continues until the density-connected cluster is completely found. Then, a new offbeat point is retrieved and processed, leading to the discovery of a further cluster or noise.

2.9. HDBSCAN

Hierarchical Density-Based Spatial Clustering of Applications with Noise is a relatively new robustness algorithm which allows varying density clusters. In addition to being better for data with varying density, it’s also faster than regular DBSCAN. HDBSCAN needs the minimum cluster size as single input parameter [12]. To begin with the algorithm, we require epsilon (eps), minimum number of points to create a dense space ’min distance’ and the minimum size of the cluster assigned to the variable ’min cluster size’. The execution starts with series of steps, at first we have to transform the space according to the density or sparsity where a point with at least ’min samples’ points whose distance with respect to the point is below the threshold value of epsilon and it is determined as Core point, then build the minimum spanning tree of the distance weighted graph. Later construct a cluster hierarchy of connected components by sorting the edges of the tree by distance. Finally, Condense the cluster hierarchy based on minimum cluster size is created and we extract the stable clusters from the condensed tree.

3. Methodology

3.1. Preprocessing

As described in previous section the text has been extracted from the PDFs of research paper of MALNIS lab, due to which there are multiple anomalies in the data. We have to clean and pre-process the data before we can pass it through the deep language models. For this task we are going to use the NLTK python library, as well as regular expressions to clean the corpus. The first step in cleaning included dropping the duplicates, removal of numbers and special characters.

3.1.1. Stop Words. Unlike other language models, removal of stop words leads to better results with the BERT model. Although the pretrained BERT model is provided in a black box, our theory for this is that stop words are not removed during training. This may be because stop words provide additional sentence information, when they are embedded within the context of the sentence. Another study focused on measuring BERT performance scores has found supporting results as they claim that stop words receive as much attention as non-stop [13].

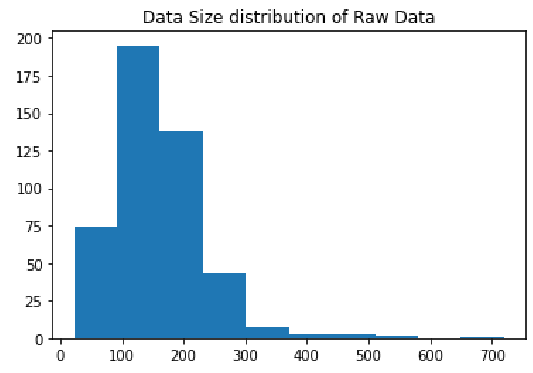


Figure 1: Document Length histogram

3.1.2. BERT word limit. BERT has a word limit of 512 words which can be problematic for large size of documents. Fortunately we were dealing with only the Abstract of research paper which had an average of 225 words in all documents as shown in Figure 1. There were few documents those were crossing this limit so we had to truncate them to 400 words.

3.2. Deep Language Model

For Deep Language model we started our research from learning about Recurrent neural networks, which are majorly used for sequence or time series data. The limitation of this model is that they cannot able to capture the long term dependency as talked in [14]. Then we explored LSTM (Long Short Term Memory)[15] networks which provided the base for seq2seq model [16] and attention models [16]. After covering the ground for understanding the BERT, as shown in figure 2 , we finally covered transformer model[10], which creates a base for the BERT [17]. The below figure will show the building block the we covered to understand and implement BERT.

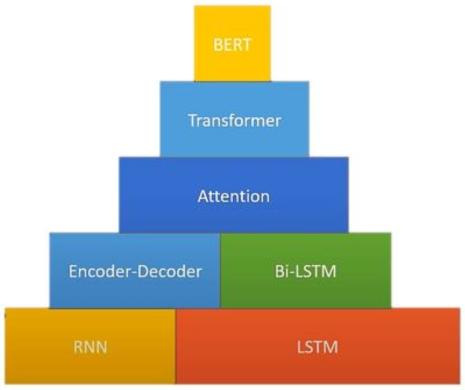
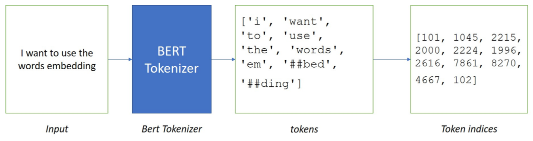


Figure 2: Learning blocks for BERT

3.2.1. BERT. The BERT as a model can have multiple use cases in the various tasks of NLP. For our project we were mainly focused on extracting the features of our data set. In case of natural language, to make it machine understandable we need to convert that text into vectors. BERT can be used for converting the text into contextual word embeddings using the available pretrained models. To create these pretrained models the stack of encoders was trained on variety of data sets with two objectives, firstly, predicting the masked words in the sentences and secondly, predicting the next sentence given the first sentence.

To get contextual word embedding from the BERT, we need to do few preprocessing of your text before we can pass it through the model. Firstly, we need to convert our text into tokens. These tokens are specially designed for the BERT input to tag the words with an index value that BERT uses to maintain it's vocabulary. BERT has limited vocabulary of 30k, which create problems while tokenize the words as it breakdown the words into ngrams and tag them with a special token. As you can see in figure 3 the word *embedding* is getting divided into multiple meaningful word which BERT understands. This might can be seen as the limitations for domain specific data where most of the words are new for BERT and it would lead to misinterpretation of the sentence.

The output of the tokenizing process will give two added tokens [CLS], at the beginning of the sentence, and [SEP], at the end of the sentence. These tokens act like flags in the model and have special index of 101 and 102. BERT models expect all the inputs to be of same length, which should be



```
['i', 'want',
'to', 'use',
'the', 'words',
'em', '##bed',
'##ding']
```

```
[101, 1045, 2215,
2000, 2224, 1996,
2616, 7861, 8270,
4667, 102]
```

Figure 3: BERT tokenization process

less than 512. Since, our data set has an average of 200-250 words and maximum of 600 words as you can see in figure 1, we truncated the data to 400 words which cost us minimal loss of data and to equalize the length we filled the remaining tokens with zero. Along with this matrix we had to also provide an attention vector for each document. This vector will mark all the non-zero tokens as 1 and rest as 0. After applying all the processing, we got two matrices of size:

$$\#documents * tokens = 466 * 503$$

For getting the contextualized embedding we used BERT-BASE pretrained model which is a 12-layer model with 768 hidden layers. To harness the features from these embeddings, the paper[10] provides multiple pooling options for us to test. For this project, we adopted to test three of them, [CLS] token embeddings, Mean of Last hidden layer, and Concatenation of Last 4 hidden layers. The output of the BERT model provided an of 503(number of tokens) x 768 for each document (figure 4) in our dataset and this will be the output for all the 12 layers of BERT model.

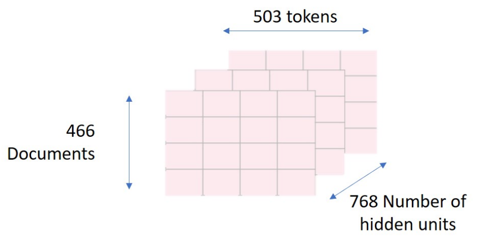


Figure 4: BERT Model Output for each layer

3.2.2. SBERT. is another model that we tried for this project. It is a modification of the BERT pretrained model designed to produce the sentence embedding which can be used for the finding semantic similarity between the two sentence. According to the paper, the sentence embedding produced by the BERT models does able to perform good on similarity tasks. Even the embedding like GLoVe can produce better results that this. There are three pooling methods defined in paper. We took the Mean Pooling for this task as it was proven to be better than the other two that were CLS token and MAX. We experimented with BERT base and it's pooling option and SBERT.

3.2.3. Cosine Similarity Task. In order to capture the nature of embeddings being produced by the mentioned model, a task was deployed involving comparing embeddings from different texts using cosine similarity. Abstracts were selected by hand into two subsets, A, and B, from the corpus. Set A contained pair of text that were observed to have similar topics, while set B was contains pair of text with topics different from A. The abstracts are then embedded using the discussed models, then cosine similarity is performed to compare the embeddings of each abstract. In order for this experiment to hold statistical significance, it was performed over multiple trials.

3.3. Dimensionality Reduction

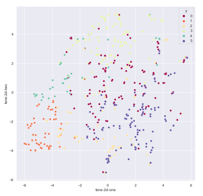
An issue with many language processing tasks is that word embeddings requires thousands of features to represent a single sentence. This poses some issues, such as slow training times or difficulty in finding optimal solution. This is known as the curse of dimensionality. Dimensionality reduction is the process of decreasing the number of features to the most relevant ones.

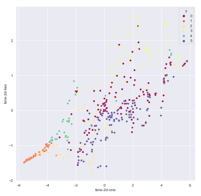
Dimensionality reduction methods are convenient for researchers as they will allow for visual analysis of the clusters which are easily readable to the human eye [18]. Another factor of convenience associated with dimensionality reduction techniques is that they will reduce the overall complexity of a given word embedding by a large factor, making clustering a more efficient task [18]. This is crucial as the word embeddings are usually represented in very high dimensions by the deep language models. For our purpose of reducing the dimensions of the word embeddings, we have implemented a technique known as Principal Component Analysis, or PCA.

PCA is a dimensionality reduction technique which aims to find project data onto lower dimensional hyperplanes, while keeping the maximum amount of variation in the data [3]. It is adapative to any type of data presented to it, which makes is very useful for processing word embeddings [3].

PCA is a linear dimension reduction technique which makes it a bad choice for a data which can be non-linear in structure. To address this fact, we took t-Distributed Stochastic Neighbor Embedding (t-SNE)[19] in account which is a unsupervised, non-linear technique. The shape of a t-sne plot majorly depend upon the choice of it's parameters like perplexity and number of iterations. Being too sensitive to these parameters, it makes t-sne a bad fit for this project. The UMAP is the technique we used for the density based clusters. We tried multiple iterations of t-sne with different values of perplexity and number of iteration and the results differ a lot. Figure 5 shows plot for perplexity at 5,6, and 7 for 3000 iterations and it can been seen that for a single increase of point there is huge change in the plot.

Uniform Manifold Approximation and Projection (UMAP) is another non-linear dimension reduction technique that we used in this project. It uses stochastic gradient decent approach to understand data structure to maintain the global as well as local structure as cited in [4]. To implement this we are taking number of components as 2, learning rate as 10 and metric as cosine. Other parameters


(a) perplexity = 5


(b) perplexity = 6

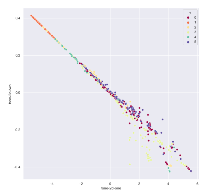

(c) perplexity = 7

Figure 5: t-sne with different Perplexities

like n_neighbour and min_dist are getting optimized by maximizing the silhouette score.

All above dimension reduction techniques were considered throughout testing of the model, in order to find which brings the most valuable and efficient representation of the documents.

3.4. Clustering and hyper-parameter tuning

For clustering part of the problem we are going to use KMeans, HDBScan and DBScan algorithms with the aforementioned Deep Language Model and compare the results in order to find out which clustering and reduction techniques works best for Deep Language Model. The hyper parameter tuning is a important part of clustering. The parameters of the clustering algorithm can be sensitive to features of the data and using default parameter can leads to bad results which might not make any sense at all.

3.4.1. Elbow Method for KMeans. The elbow method is a useful technique for determining the optimal number of clusters to choose for an algorithm such as K-means. For a tuning starting point, we employed this technique as we had no prior knowledge of the optimal number of clusters in our dataset. The elbow method captures the variation in the dataset and as soon as the number of clusters exceeds the actual number of groups in the data, the added information will drop sharply. This is because at that point, it is just subdividing the actual groups. Assuming this happens, there will be a sharp elbow in the graph of explained variation versus clusters. The elbow plots for all the variations of BERT are shown in figure 6.

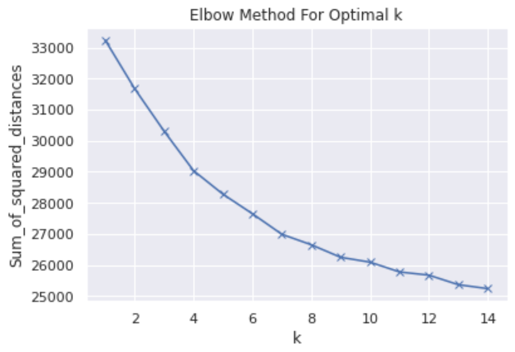


Figure 6: Elbow method to estimate number of clusters

3.4.2. Nearest Neighbour for Epsilon. Most of the machine learning algorithms like DBSCAN and HDScan in this case, we have several parameters which accounts for better cluster computation. One such parameter is epsilon(eps) that needs to be optimised systematically. To define the optimal value for eps, will consider two arbitrary points who are close neighbors only if the distance between the two points is below the threshold epsilon. To compute the value for epsilon, we have implemented an algorithm to find the best suitable number, the calculation begins from euclidean distance on each pair of data x and y by implementing distance function. A function of matrix is used on the results distance that needs to be normalized into the matrix which is calculated after distance transform data into a multidimensional vector. This technique of normalization is used to facilitate the search for the k nearest neighbors on each line of the distance calculation. After computing the distance and the matrix, the initialization is done to count the number of rows in the data. Further searches carried out three nearest neighbors of each matrix line spacing for sorting is done in ascending for each result set the closest distance to the neighboring points. Later, sorting the results of each line of the nearest neighbor made a plot with the x-axis and y-axis is the object and the distance k nearest neighbors. Once plotted, we would find major changes in the curve corresponds to the significant Epsilon value at different levels of density for the given dataset.

3.4.3. Silhouette Maximization Optimizer. In DBScan and HDBScan algorithms, after getting the optimal point for epsilon we wanted to optimize the other parameters of the algorithm. Since we are using the DBscan and HDBscan in combination of UMAP, there was a need to optimize them in sync. To do so we used silhouette score and tried to maximize it to get some directions in which we can try other parameters. We used hyperopt library which uses the method tree-structured Parzen estimator(TPE) to minimize our objective function. We declared the search space for 'min distance' and 'n neighbors' which are hyper-parameters of UMAP and arrive at the final best values for our hyper-parameters after 25 evaluations.

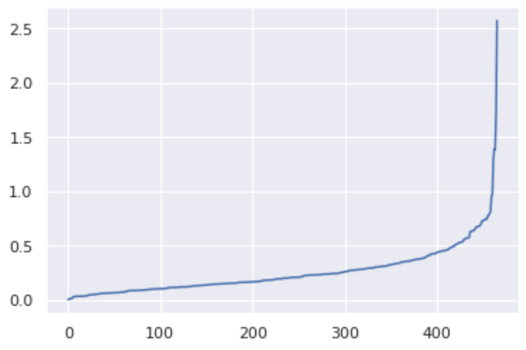


Figure 7: Nearest Neighbour plot for finding Epsilon Value for HDBScan and DBSCan

For HDBscan we optimized minimum cluster size and keeping other parameters as folow: min_samples = 1, cluster_selectio_epsilon = 0.3, metric = 'cityblock', cluster_selection_method='leaf'

Table 1 will provide the summary of the iteration.

Clustering Algorithm	Maximum silhouette Score Achieved	silhouette Score of final implementation
DBscan	0.801	0.25
HDBscan	0.84	0.20

TABLE 1: Silhouette Score Achieved v/s Final

Upon using the parameters of maximum silhouette score, the results were not making sense, hence we have to reduce the score in order to get intuitive results.

4. Results

For this project the main goal was to test out the Deep Language model with existing clustering algorithms and try to evaluate them based on semantic similarity clusters . We are using a data that does not have a benchmark so its hard to say which how many cluster can be present in the data. Also, considering the fact that all the research papers belongs to similar domain there was a possibility that might fall under same cluster. We first evaluated the different variants of BERT (CLS Token, Mean Pooling and concatenating last four layer)and SBERT (Mean Pooling) model. We assessed them based on the ability to find similarity and dissimilarity between the sentences. The results shows that the SBERT with mean pooling easily surpassed the BERT. We have captured the difference in these models by looking at the color pattern of cosine similarity heat maps as shown in Figure 8. There are two things we noticed over here. Firstly, the range of the map and the color density variations in them. SBERT shows the highest range and color intensity changes as compared to all the other models. So, for this reason SBERT is taken as the default model for generating sentence embedding.

4.1. Cosine Similarity

Table 2 shows the average cosine similarity scores for each model during the cosine similarity task discussed in the methodology section. For SBERT, the average score for similarity between embeddings for abstracts in A and B were 94, and 87.

For CLS token variation of BERT the similarity scores for abstracts in A and B were 94, and 87. For the mean last layer variation of BERT the similarity scores for abstracts in A and B were 92, and 88. For the concatenated last layer variation of BERT the similarity scores for abstracts in A and B were 90,and 87.

4.2. Clustering

This project report showcase three clustering algorithms that are KMeans, DBScan and HDBscan. We use these algorithms in combination of dimensionality reduction schemes

BERT Model	Similarity Set A	Similarity Set B
SBERT	75	50
CLS Token	94	87
Mean of Last Layer	92	88
Concat Last 4 Layers	90	87

TABLE 2: Cosine Similarity Range for BERT variants

as we notices that providing high dimensional inputs the density based algorithms were giving only one cluster. To avoid this we introduced UMAP with density based model and PCA/t-SNE with KMeans. The results of 2-D plots of these cluster are given in figure 9. The density based cluster in combination with UMAP out performed the centroid based KMeans in the results and detection of buckets. The plots enable us to understand the grouping of each clustering. We spend most of out time to do multiple iteration for each of the algorithm to fine tune the parameter and produce the better result. The table 3 shows the final cluster we found for each model and the size of each cluster can be seen in Figure 10.

KMeans with PCA - 6 Clusters	
Clusters	**Topics in clusters**
Cluster 0	Active Learning, Deep Learning, Interactive Analysis
Cluster 1	Radio-Graphs, X-Rays and Healthcare
Cluster 2	Summarizing, Social Media Data and IR
Cluster 3	Deep Learning and Deep Language Models
Cluster 4	Mixed
Cluster 5	Clinical Data and healthcare

KMeans with t-SNE - 3 Clusters	
Cluster 0	Deep language Model and NLP tasks
Cluster 1	Deep language Model and NLP tasks
Cluster 2	Active Learning, Deep Learning, Interactive Analysis
Cluster 3	Mixed
Cluster 4	Mixed
Cluster 5	X-Ray, healthcare Related Data

HDBScan with UMAP - 4 Clusters	
Cluster 0	Topic Modelling, Summarizing
Cluster 1	Clinical data, Healthcare
Cluster 2	X-Rays, Clinical data, Healthcare
Cluster 3	Topic Modelling, Classification
Cluster 4	Language Models
Cluster 5	Active Learning, Deep Learning, Language Models
Cluster 6	Mixed
Cluster 7	BERT
Cluster 8	Visualization
Cluster 9	Embeddings
Cluster 10	Interactive Analysis
Cluster 11	Active Learning, Summarizing

DBScan with UMAP - 6 Clusters	
Cluster 0	Classification, Deep Language Model, Summarizing
Cluster 1	Summarizing, Social Media Data, BERT
Cluster 2	Information retrieval, Embeddings.
Cluster 3	X-Rays, Clinical data, Healthcare
Cluster 4	Active Learning
Cluster 5	Visualization

TABLE 3: Clustering buckets

5. Discussion

5.1. Frequency Topic Models

The main model implementation in this investigation were various forms of the BERT encoder model. While introducing topic models, a brief summary of frequency topic models was provided, mainly the TF-IDF approach. Although not implemented, these models were briefly discussed in order to further develop the history of topic models thus far. A frequency model was not implemented as it was hypothesized early on that such implementations lose contextual meaning while analyzing text. Due to this hypothesis, we thought that a model such as BERT was a better choice, as it is able to retain contextual meaning while processing text data. In order to further this investigation, future work includes comparing the performance frequency topic models on the tasks provided, with the performance of the various implemented BERT models.

5.2. BERT

As it is clear from table 1, the BERT model implementations struggled while determine text was not similar. Given these inaccurate results, BERT still provided an essential benchmark to the study. This is because the SBERT model, which is discussed below, was implemented from the base BERT model. We recognize BERT as an important stepping stone for this project, and within additional study in this area.

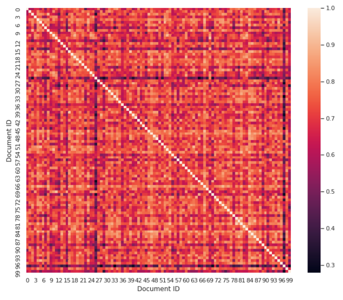
5.3. SBERT

According to table 1, although SBERT typically yielded a lower similarity score compared to the other BERT implementations, it was the only model that accurately captured deviations in the similarities within the set of papers that did not share a topic. This finding shows that the SBERT may be a more competent model in determining if a paper is not actually similar.

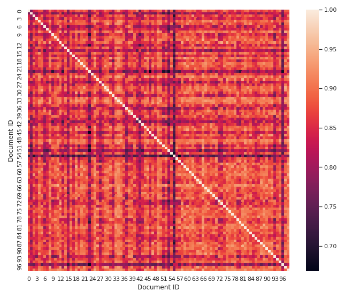
Additionally, as mentioned previously, the SBERT model outperformed the BERT variations while clustering documents by topic. This is because during training, SBERT is tuned to capture similarities between documents, using cosine similarities [1]. This indicates that SBERT will naturally have a better performance during semantic similarity tasks.

The results from figure 4 are also supported by the findings shown in [1], as they discuss that both averaging the last layers of the BERT model, and using [CLS] tokens can yield relatively bad sentence embeddings. From this research, it is evident now that these methods are not optimal for a study which focuses on finding strong embeddings.

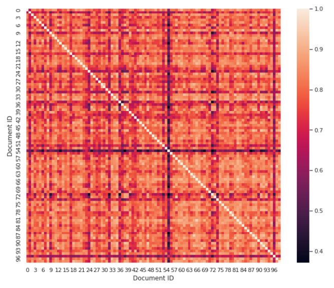
Strong sentence embeddings are important, as this project involves summarizing abstract sections which are made up of at least a few sentences. Our results support this claim, as SBERT was found to have the best performance on the tasks out of all models tested.



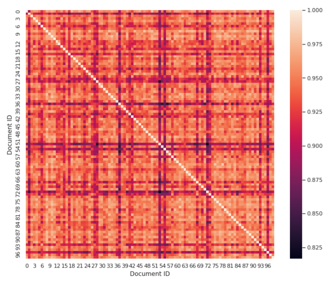
(a) SBERT



(b) Last 4 layers

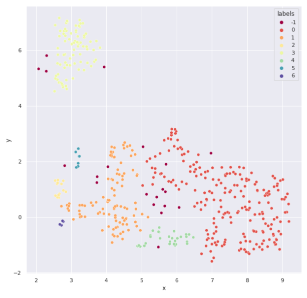


(c) Mean Of Last Layer

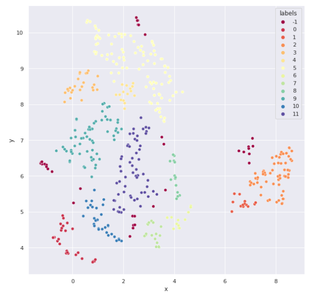


(d) CLS Token

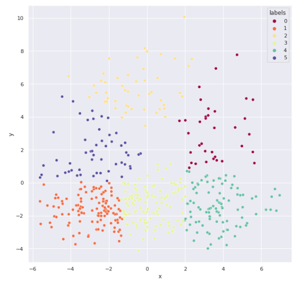
Figure 8: Cosine similarities Heat Maps



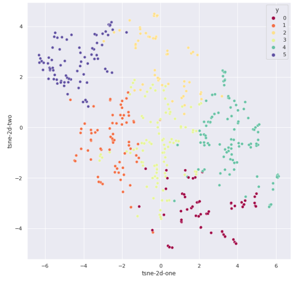
(a) DBScan with UMAP



(b) HDBScan with UMAP

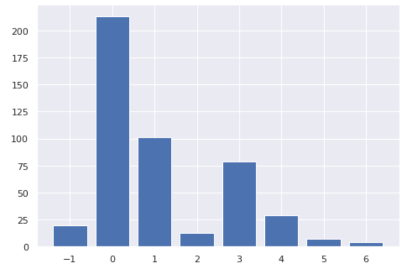


(c) KMeans with PCA

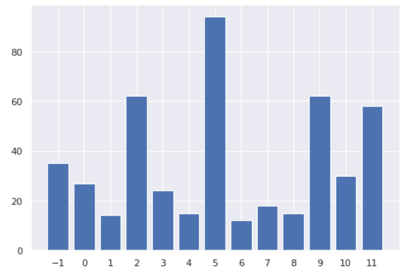


(d) KMeans with t-SNE

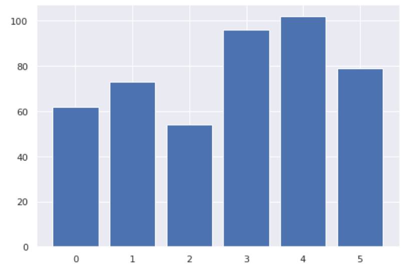
Figure 9: Clusters in 2-D plots



(a) DBScan with UMAP



(b) HDBScan with UMAP



(c) KMeans with PCA

Figure 10: Number of Clusters

These SBERT related findings all contribute to the decision that SBERT is a very robust model for the specific goals of this project.

5.4. Clustering

5.4.1. Kmeans. We started with KMeans to set the benchmark for the results. The results for Kmeans with PCA in table 3 shows decent performance in bucketing the documents. Also we tested with more number of clusters but the cluster were not making much of sense, hence we decided to keep it to 6. We also tried using t-SNE but we never found a reliable result with any combination of perplexity and number of iteration. The results also resonates the same, as the clusters created with mixture of the topics.

5.4.2. Density based Clustering. The main issue with KMeans is that we have to define before hand how many cluster do we need in the data. Also, the KMeans could not detect the outliers in the data sets. To overcome this, we tried our data set with density based clustering like HDBScan and DBScan. These two algorithms helped us discovering new patterns those were not possible in KMeans. As shown in figure 10 and table 3 there is clear difference in number of cluster and type of topics in a clusters.

The UMAP is being is used for reducing the dimensions of the BERT embeddings. We tried few iterations with high dimension data with both density based clustering algorithm but we found that they were not compatible with high dimension data. The reason for this is still up for the discussion and more investigation is required in this. We adopted this over PCA and t-sne as cited in [4] the algorithm is capable of preserving the global as well as local structure while reducing the dimension from high to low. But one thing that makes UMAP difficult to use is the stochastic nature, which can create problem to reproduce the same results every time.

5.5. Limitations

There are a number of limitations to the implementations of this study. This section will present the most pressing limitations to the study.

5.5.1. Limited Input Size. Both BERT models had a limited input size of 512 words. The model would thus not accept input text that was not truncated or summarized, if the original text exceeded this word limit. This is limiting as longer input text is not able to be processed within this study without a lot of preprocessing. Due to the fact that we only used abstracts, we did not experience significant data loss, but this is a potential issue for analysis of larger texts in future work.

5.5.2. Limited Vocabulary. Both BERT models were trained to have only a vocabulary size of around 30000 words. This was limiting especially while processing research papers that referenced domain specific concepts and techniques that were not known by the model.

5.5.3. Slow Processing Time. Specific to the BERT model, there was a high computational requirement in order to run the model. During the experiments, we observed the model to take around 45 minutes to only process around 466 sentences. In order to compensate for this high computational requirement, we had to input sentences individually. This was quite taxing on a general use laptop, which was used for most of the model processing.

5.5.4. Unlabelled Dataset. We were limited to the fields from the dataset that was used throughout the study. This dataset did not include true labels for each individual document. These labels would simply be the overall subject of the given paper. The knowledge of these true labels would help further validate the results. Unfortunately, because labels were not initially presented in the data set, developing that field would involve having to manually read through each paper in order to label it.

5.6. Future Work & Implications

The study is constrained to only four model variants. We originally planned on implementing more models, but time limitations reduced that number. Given more time to work on this investigation, more variants and models could be tested. This would broaden the perspective of which language model can perform the best on the tasks presented. Thus, a large part of the future work includes testing other deep language models, such as InferSent, Universal Sentence Encoder, XLNet and Open-GPT3.

The BERT variants were also implemented out of the box. Due to this, there was not a lot of experimentation with the model's hyperparameter during the study. Future work involves tuning these hyperparameters in order to make BERT understand domain specific papers. The general hypothesis for this future work is that BERT may produce better embeddings given the study's dataset, which would directly improve clustering.

Other future work includes collecting more research papers for the dataset, as well as developing more tasks to test the models. This would further validate each models robustness within the context of the study, as well as obtaining a deeper understanding of each models strengths and weaknesses.

6. Conclusion

The goal of this study was to find which deep language model was able to best produce sentence embeddings on a given research paper corpus so that it could understand semantics. These embeddings would then be clustered in order to assess the overall topics within the corpus using of KMeans, DBScan and HDBScan. This study is important as quickly processing large corpus's of research papers has the potential to help researchers find papers that match their specific needs during research. This will ultimately increase the efficiency of the background research process for many fields.

Due to limited time, only 4 variants of a pretrained BERT model was implemented, BERT base with mean pooling, BERT base with concatenating last four layers, BERT base with [CLS] tokens, and SBERT. It was found that SBERT was able to best generate sentence embeddings for clustering. For documents clustering with KMeans, DBScan, and HDBScan algorithm. It is found the density based clustering works best with SBERT deep language model and HDBScan has little edge over the DBScan in performance. Due to various limitations, the future work for this study involves implementing various other deep language models, improving the dataset, and implemented more testing benchmarks for the model.

6.1. Gitlab Link for code

The code for this project will be available in git lab https://git.cs.dal.ca/makan/ml_winter_term_project

References

[1] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019.

[2] C. Malzer and M. Baum, "A hybrid approach to hierarchical density-based cluster selection," in *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2020, pp. 223–228.

[3] I. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2016.

[4] L. McInnes, J. Healy, and J. Melville, “Umap: Uniform manifold approximation and projection for dimension reduction,” 2018.

[5] H. P. Luhn, “The automatic creation of literature abstracts,” *IBM Journal of Research and Development*, 1958.

[6] A. Nenkova and K. McKeown, “A survey of text summarization techniques,” *Mining Text Data*, 2012.

[7] T. Dunning, “Accurate methods for the statistics of surprise and coincidence,” *Computational Linguistics*, 1994.

[8] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut, “Text summarization techniques: A brief survey,” 2017.

[9] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information Processing & Management*, 1988.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017.

[11] L. McInnes, J. Healy, and S. Astels, “Comparing python clustering algorithms,” 2016. [Online]. Available: https://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html

[12] C. Malzer and M. Baum, “A hybrid approach to hierarchical density-based cluster selection,” *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, Sep 2020. [Online]. Available: http://dx.doi.org/10.1109/MFI49285.2020.9235263

[13] Y. Qiao, C. Xiong, Z. Liu, and Z. Liu, “Understanding the behaviors of bert in ranking,” 2019.

[14] P. S. Y Bengio and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, 1994.

[15] S. Hochreiter and J. Schmidhuber, “Lstm can solve hard long time lag problems,” *Advances in neural information processing systems*, 1997.

[16] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” 2014.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2019.

[18] E. Sherkat, E. E. Milios, and R. Minghim, “A visual analytics approach for interactive document clustering,” vol. 10, no. 1, 2019.

[19] L. Van Der Maaten, “Accelerating t-sne using tree-based algorithms,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3221–3245, 2014.