Machine Learning for Big Data Assignment 4

Aamir Shoeb Alam Khan
Masters of Applied Computer Science
Dalhousie University
Halifax, Nova Scotia
am754815@dal.ca

Noah Sealy
Masters of Computer Science
Dalhousie University
Halifax, Nova Scotia
noah.sealy@dal.ca

1. Background

1.1. Text Summarization

Since the creation of the printing press to the vastness of the Internet today, hundreds of millions of text characters are printed daily, through social media, news articles, or even assignments like this. Text data is an effective and efficient way to easily convey information to other people all around the world. With the constantly increasing amount of text data available, text and natural language processing become more and more prevalent.

Within the world of text and natural language processing, text summarization is an elegant solution in solving problems which relate to parsing through a large corpus of text. Text summarization models are able to save a significant amount of time by clustering words in order to represent them as semantics, topics, and more.

Throughout this paper, we will explore various text summarization models, specifically a variant of the Latent Semantic Analysis model and a variant of the Latent Dirichlet Allocation model. We will then compare the performances of the two models, in order to evaluate which model has the advantage in the process of text summarization. The models will both run on text documents which will be discussed in the following sections.

1.2. Available Methods

This subsection will briefly describe various text processing options from literature review, before moving onto the chosen models themselves. The models that will be discussed here are topic word models, frequency driven models, and sentence clustering models.

1.2.1. Topic Word Models.

Nenkova & McKeown provide a summary of topic word models for text summarization, in their discussion referring to the work of Duhn [9] as well as Dunning [5] for advancements in the topic [10]. Topic word models are based on finding significant words within documents which may be used to represent the document as a whole.

Luhn, provided the idea that each factor of a document has some sort of measurable significance [9]. Luhn's investigation proposed that the significance of each word in a document can be found in their individual frequencies throughout the document. Additionally, from these significant words can be measured significant sentences [9]. In order to avoid stop words from looking significant merely due to frequency, they are removed. Thus, based on the idea that important words get repeated, these significant words would be able to appropriately summarize the document.

Dunning further expanded on Luhn's idea of word significance in order to create a more robust model. Dunning utilized more advanced statistical analysis to find significant words which the previous models may have missed. Dunning comments on an issue of some frequency based models such as Luhn's was that very significant words may be missed, which could later sabotage the final summary of the document [5].

Though only two examples were given of this type of model, topic word models can be used to summarize a document based on some significance measure of the words and sentences it contains. An important discussion piece throughout Luhn's publication was that machines only ever understand words as pure physical objects [9]. 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 numeric representations.

1.2.2. Frequency Driven Models.

According to Allahyari *et al.*, frequency driven models involve the assignment of weights to word in order to derive representations which best summarize a given text [2]. This means that words with greater weights will represent the given document in order to summarize the overall text.

Early frequency driven model approaches used various probability methods to assign weights based on word likelihood throughout the given text. Though these models would find representations of the text, they were flawed in the sense that they did not have strong compensation for stop words, which had a very high probability, but low

meaning related to the summary of the text. Thus a more robust frequency driven approach was developed, known as TF-IDF [2].

TF-IDF is better than the previous frequency driven approach because it is able to compensate for the disregard of stop words found in previously discussed probability models [11]. Similar to Luhn's topic word model [9], TF-IDF will weight words based on their frequency throughout the document, as those words are in some way significant compared to others, and can thus be used to summarize the text. A key difference is that TF-IDF will decrease the weight of words based on their frequency across the corpus being investigated [11]. In general, this fixes an issue which was apparent in both early topic models [9] and frequency driven models [2]; stop words will now have low weights despite having a high frequency through a document. TF-IDF is a robust model which is able to assign the correct weight to words according to their significance in order to properly use them to represent a summary of the text [11].

1.2.3. Sentence Clustering.

Sentence clustering involves grouping sentences within the input text by some function of their similarity or distance [10]. A commonly used similarity function for these models is known as cosine similarity [12].

In general, a clustering model using cosine similarity compares sentences in a high dimensional vector space. While comparing the numeric vector representations of sentences, the cosine of their angle is used as a similarity function [12]. This is due to the nature the cosine of two angles, which will yield 1 for identical vectors and 0 for orthogonal vectors. Thus, a cosine similarity closer to 1 indicates that the vector representations of the sentences are similar. Sentences which are considered similar by means of cosine similarity are thus grouped together by the clustering model, and are used to represent different topics of the text. These topics found by the model thus provide a summarization of the topics covered throughout the input documents, with the relative size of the cluster indicating topic importance throughout the document. [10].

1.2.4. Moving Forward.

Although we did not choose to investigate these models, we believe they are still important to briefly address in order to show a more developed understanding of what text summarization has to offer. It is also important to note that this is just the tip of the iceberg for text summarization methods, we only actually investigate a sub class of models known as extractive models. In general, extractive models function by finding a way to take a summary of an input document, which is used as the representation of that document [2]. The following sections will discuss this investigation further.

1.3. Choosing the Models

Though the models discussed provide elegant solutions to finding the significance of sentences throughout the topic in order to summarize it, the investigation chose other models in order to summarize text. The models chosen were latent semantic analysis (LSA) and latent Dirichlet allocation (LDA). The models will be discussed in terms of theory and implementation throughout the document. The models were chosen for the investigation as they both involve deducing latent variables given the input text.

LSA and LDA also both utilize the concept of latent variables throughout their implementation. This refers to underlying variables that are not necessarily physically presented in the models outputs. We thought that it would be interesting to delve into the various types of latent variables, and find which may be more effective to target. These latent variables may allow the model to represent summaries of text closer as a human would compared to the other models; as meaning is sometimes hidden or abstract in natural language. As fully understanding natural language involves a lot of natural human thinking, models such as these may be more robust than those which use concepts such as frequency or similarity measures in purely mathematical spaces.

1.4. Choosing the Documents

The documents were taken from Aeon, which is a charity website devoted to collecting articles on various topics, ranging from psychology to sports [1]. Each document is greater than 30 paragraphs, and was converted to a .txt file containing only the contents throughout. The first document was titled "Here's to naps and snoozes", and related to sleep in teenagers. The second document was titled "Soccer broke my brain", and related to concussions in sports. The third document was titled "Unboxing mental health", and related to mental health awareness.

Though to a human each document has clear topics, it may not be so clear to the trained models. These documents were chosen as they have underlying themes which relate; the brain is related to sleep, concussions, and mental health. Sleep is related to napping as well as mental health, in the same way that mental health could be closely connected to mental health. We will attempt to train our models with not just train on binary documents which have clear set topics, rather loose themes and topics that intertwine with one another. This will make hyperparameter tuning very important and perhaps more interesting than normal.

2. Models

2.1. Latent Semantic Analysis

2.1.1. Theory.

Unlike the text summarization models discussed, latent semantic analysis processes text data with the intention of producing a representation based on the texts latent features. In this case, latent refers to a concept which cannot be measured by a machine, the underlying topic

of sentences and documents. Overall, it is an unsupervised method that aims to summarize text by presenting the topics of the documents presented. By doing so, a brief description of the document can be provided which may be easier to understand by a human compared to measurements relating to frequencies and weights of text in a document.

There are key processes which occur during the execution of LSA which should be noted. In sequential order, those processes are building a term sentence matrix, weighing each entry in the matrix, singular value decomposition, and calculating a topic representation for each document from the results of the singular value decomposition.

Term Sentence Matrix.

The raw text data of the corpus to be processed by an LSA model will first need to be converted to a term sentence matrix, A. This is an n * m matrix, where n represents the rows containing each word in the input text's dictionary, and m represents the columns containing each sentence from each document provided in the corpus to be processed [2]. This term sentence matrix will act as a representation of the corpus in order to be processed by the LSA model.

As there are quite certainly more words than sentences in the typical document, the columns in the term sentence matrix will hold more non-occurrences of words than occurrences, resulting in a sparse matrix [7]. Each column can also be used as a vector representation of each sentence within the corpus, as values will only be greater than zero when a word occurs in that sentence. Mapped out on a Euclidean space, and these vectors now represent words in a space which a natural language processing algorithm can understand, thus showing the translation power of term sentence matrices.

Term Sentence Weights.

As mentioned, each occurrence of a word in a sentence will correspond to a value greater than zero in the sentence word matrix, a(ij), where i is the corresponding word and j is the corresponding sentence [10]. This section will briefly elaborate on the derivation of those values, which from here will be referred to as weights.

The weight of the corresponding word and sentence must be derived from some term frequency function, such as those discussed previously, in order to fully represent the significance of each word in a sentence. A common method of setting these weights is the previously discussed TF-IDF frequency driven model, as it will prefer words which seem significant to the sentence being analyzed, but has the capability of punishing stop words [11]. Evidently, if a word i does not occur in sentence j, word i's weight will be set to zero.

Singular Value Decomposition.

Singular value decomposition is a form of dimensionality reduction that is carried out on the term sentence matrix. This results in three matrices which are typical in this process, U, Σ, and V^t. Matrix U has words as rows and latent topic representations as columns [3]. Matrix Σ is a diagonal matrix where each nonzero weight represents the weight of the topic of the selected index [2]. Lastly, matrix V^t has sentences as rows and latent topic representations as columns [3].

SVD is used in order to further model the latent semantics among the words and sentences in the document [3]. As Gong & Liu discuss, words which are closely related, or synonyms, typically appear in the same types of sentences, potentially having relating frequencies throughout the texts [7]. Due to this inherent similarity among the documents and the nature of the vector representations from the term sentence weights, these related words will be found close together when analyzed on an n-dimensional space [7].

Topic Representations.

Once the input text has been processed with SVD to results in three matrices to represent latent features, sentence selection is required in order to truly summarize the text. Sentence selection provides a cleaner summarization of the findings of LSA compared to simply summarizing the document with the matrices. According to Alpaslan & Cicekli, there are many methods of sentence selection [3], but as this paper has already discussed many LSA topics in terms of the investigation by Gong & Liu, it will briefly summarize the sentence selection method they proposed.

The chosen method of sentence selection to discuss is a simple solution which utilizes two of the three matrices produced from SVD. As mentioned, V^t contains weights corresponding to given topics and sentences in the documents. Greater weight values indicate stronger correlations between the topics and sentences [3]. In other words, topics important to a sentence will have a higher weight.

Greater weights in Σ correspond to topics that are more important to the document analyzed [7]. With the use of the two matrices V^t and Σ, an algorithm can take the most significant topics, and choose the most significant sentences corresponding to the selected topic. The number of topics and sentences chosen is simply an input parameter set during implementation.

Topic representation is crucial to LSA in order to bring meaning to the term sentence weight assignments and SVD which occur before it. Gong & Liu show just one of many sentence selection methods, which ultimately summarize documents according to the calculated significance of underlying latent topics which are found throughout the analyzed documents.

2.1.2. Implementation.

Implementation of the LSA model can be broken down into the preprocessing of the text data and the model implementation itself. This implementation was done in Python, using libraries such as Pandas, NLTK, and Sklearn. Various parts of the code will be discussed throughout, but a full running representation of the code can be found in the additional files attached with this assignment.

Preprocessing Text Data.

Preprocessing of the input data is crucial to a proper execution of a model, this applies just the same for text data. In sequential order, preprocessing involved converting the text data to a data frame, removing punctuation, lower case conversion, stop word removal, tokenization, lemmatization,

and finally creating a bag of words representation. Each of these processes will be briefly discussed in relation to the code. Tokenization through bag of words representation were all taken care of using Sklearn's TF-IDF Vectorizer, so those topics will be discussed together.

Data Frame Conversion.

As discussed, the text data was converted to a .txt file from its source website [1]. This text file was first read, tokenized by sentence, and stored as a dictionary in Python. This dictionary was then passed into a Pandas data frame. The text would be stored within a data frame until it was needed by the model. It should also be noted here that the three text files chosen for analysis were never combined, rather implemented separately, but running the model only on the selected file..

Punctuation Removal.

Any punctuation found within the text data would be irrelevant to the LSA model, so removal was necessary. In order to do so, a regular expression was used which eliminated any punctuation that was found in the text data, and replacing it with an empty character. Throughout the development process, this regular expression method was found to outperform various punctuation removal libraries in terms of the amount of punctuation removed.

Lowercase Conversion.

In order to standardize the text itself, every word needed to be converted to have a consistent casing. As words relate differently to a computer compared to a reading human, the seemingly small difference between, for example, "Pineapple" and "pineapple" may alter results as the model will process the words as different. In order to avoid this potential error, all words to converted to lower case before being processed by the model.

Stop Word Removal.

Words contained in the NLTK library's stop words import were removed from the text data. Stop words are words which typically act as words to fill the structure of the sentence, but not the semantics; any stop words which were thus removed. Throughout each text file, this step of preprocessing usually reduced the amount of words contained in the data set by some factor.

Sklearn TD-IDF Vectorizer.

The Sklearn library's TD-IDF Vectorizer import allowed for the implementation of a bag of words for the dictionary of the input text, the weight assigned to each word in the bag of words were weights according to a TF-IDF model. The TF-IDF Vectorizer also took input other preprocessing options which allowed for the majority of the preprocessing to occur in one place.

The other measure of preprocessing that occurred within the TF-IDF Vectorizer included further word vectorization, in order to prepare the data as input for the model. Lemmatization was also included, which further standardized the word data, similar to lower case conversion.

LSA Model Implementation.

```
svd_model = TruncatedSVD(
    n_components=n_topics,
    algorithm='randomized')

svd_model.fit(x)
```

Figure 1: Sklearn's SVD library used for the LSA model implementation.

```
Topics in LSA model:
Topic #0: health mental disord
Topic #1: disord anxieti social
Topic #2: system diagnost categor
```

Figure 2: LSA model output after processing "mental_health.txt" with three expected topics.

The LSA model was implemented using Sklearn library's TruncatedSVD import. As shown in figure 1, the LSA model would take in the expected amount of topics as input, and return a decomposed matrix showing the most significant topics, and the sentences relating to those topics. The output shown in figure 2 is an example of the output of this model, which was run on the "mental_health.txt" file with the expected amount of topics set to three.

2.2. Latent Dirichlet Allocation

2.2.1. Theory.

LDA is similar to LSA in the sense that its goal is to find latent features throughout the analyzed text which will be used to represent topics. The topics found from LDA will be used to summarize the text, as seen in LSA. Other than that LDA and LSA do not have many similarities in terms of their models; where LSA uses algebra to find the significance of topics, LDA will use an array of probability distributions in order to generate documents to relate them back to the original corpus.

Before discussing implementation, there are a few key concepts to LDA which will be discussed, in relation to sequence with respect to the model itself. Those concepts all relate to the generative distribution of a document, which can be used to represent the LDA model itself [4].

$$p(D;\alpha,\beta) = \Pi_{d=1}^{M} p(\theta_d|\alpha) * \Pi_{n=1}^{N_d}(p(z_{dn}|\theta_d) * p(w_{dn}|z_{dn},\beta))$$

This mixture model will give the probability of a document being generative with LDA [4]. D represents the new document, M represents the amount of documents to

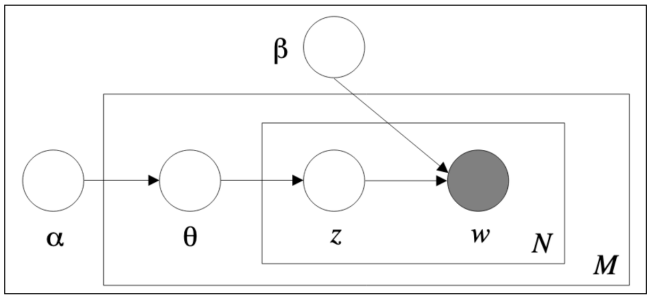

Figure 3: LDA mixture model diagram. M corresponds to the number of documents. N corresponds to the number of words in document M. [4]

be generated, and N_d represents each word in the generated document d [4]. This generative formula can be generalised to figure 3, which shows the foundation of the LDA model..

According to Blei et al., figure 3 has three levels, the first corresponding to α and β, the second corresponding to θ_d, and the last corresponding to z_dn and w_dn [4]. A brief discussion for the LDA model will follow this level break down to further analyze the generative model. A brief summary of similarity measures for the model and the original document will then be provided to conclude the theory discussion.

Figure 3, Level 1.

At level 1 of figure 3, there is α and β. These are parameters which are fed into the model, and sampled at the corpus level [4]. This means that these parameters will remain constant while generating documents for the LDA model.

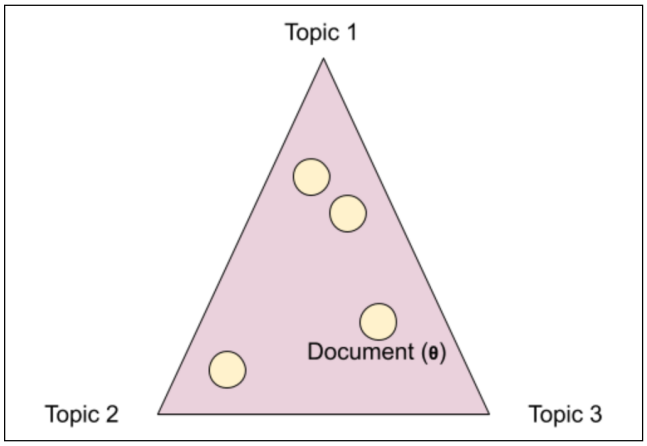

Figure 4: Example dirichlet distribution, with k topics in k-1 dimensional space.

The parameter of is α, which feeds into a special distribution used in order to sample θ later on, which is known as the Dirichlet distribution. The Dirichlet distribution allows for the sampling of a random variable θ, which can be represented as a k-1-dimensional simplex [8], as shown in figure 4. The dimension k in the case of the generative model shown in figure 3 is the number of anticipated topics. Geometrically, each corner of the simplex will represent a topic, while documents will distribute among the space according to their relative similarity to each topic. For instance, a document relating to basketball may reside closer to a corner which represents the topic of sports.

The parameter α will control the sampling distributions, controlling the modal peaks throughout the geometric representation. While α is greater than one, points are more likely to reside in the centre of the simplex, and while α is less than one they are more likely to reside in the corners [8]. Due to the each topic being in a corner of the simplex, in order for documents to relate to them a lower α may prove better in implementation of the LDA model.

Figure 3, Level 2.

The second level of figure 3, and the generative model in general, involves sampling θ [4]. θ is sampled for each new document that is generated, and can be thought of as representing the topic of words within the document. θ is sampled directly from the Dirichlet distribution, and can be thought of as a multinomial distribution relating to the available K topics which the model is trying to summarize under.

When a random variable is sampled from θ, as we will see from z_{dn} in the next level, the value of that variable will represent a topic. This variable will be sampled for each word in the document, in order to generate a document which is determined from the distributions provided. Thus sampling θ from the Dirichlet distribution denoted as α, is the process of choosing the probabilities of topics to be used in the next generated document, which will determine the overall topic and summary of that document.

Figure 3, Level 3.

The last level of figure 3, and the generative LDA model, interacts with sampling the random variables z_{dn} and w_{dn} from θ and a distribution which assigns available words to available topics parameterized by β. It should be noted that another Dirichlet can be used as this distribution, and it would belong in the same level as θ sampling [8]. This sampling is done with every new word in each generated document.

As discussed, z_{dn} represents the topic which the generated word will be represented by. z_{dn} will be sampled directly from θ with the conditions discussed prior.

w_{dn} is sampled depending on what z_{dn} sampled from θ. We can see from this sampling that the Dirichlet distributions truly contain distributions of distributions. Depending on which topic was chosen, w_{dn} will sample from another multinomial distribution that is divided by the topics.

Similarity Measures.

The resulting distribution parameterized by β distribution will act as final output. The LDA model which is producing topics which are closest to the original models will be selected as the final model. This final distribution can be represented as a list of words which acts as a dictionary of words per each given topic. This list of words is the text summary, and it will be up to researchers to assign each of

the lists to one of the topics [4] in order to provide further meaning to each list, and the summarization in general.

2.2.2. Implementation.

Preprocessing Text Data.

In order to keep the LSA and LDA model implementations as consistent as possible in order to make a just comparison, the text preprocessing was exactly the same for both models. As seen in the LSA implementation section, in sequential order, preprocessing involved converting the text data to a data frame, removing punctuation, lower case conversion, stop word removal, tokenization, lemmatization, and finally creating a bag of words representation. Steps regarding tokenization through creating the bag of words was also processed through Sklearn's Count Vectorizer.

Evidently, the data for the LDA model was not preprocessed using the TF-IDF Vectorizer as was the case for the LSA model. As discussed by Blei et al., TF-IDF has some shortcomings due to its text reduction nature, which could provide issues to the LDA model; thus TF-IDF is not used here [4]. The Count Vectorizer is used just the same as the TF-IDF Vectorizer though, except instead of producing a TF-IDF representation of the dictionary, a bag of words with word frequencies is instead provided. This bag of words was sufficient for the LDA implementations, as the model's distributions take no consideration of word position or context.

As the preprocessing steps were otherwise exactly the same for all other factors, please refer to the LSA preprocessing text data implementation section if more detail on any of the discussed methods is needed.

```
lda = LatentDirichletAllocation(
    n_components=num_topics)

lda.fit(x)
```

Figure 5: Sklearn's LDA library used for LDA model implementation.

```
Topics in LDA model:
Topic #0: approach transdiagnost person
Topic #1: treatment disord health
Topic #2: mental health anxieti
```

Figure 6: LDA models output after processing "mental_health.txt" with three expected topics.

LDA Model Implementation.

The LDA model was implemented using the Sklearn library's LatentDirchletAllocation import. As shown in figure 5, the LDA model would take in as input the expected amount of topics, similar to the LSA model implementation. The model would return a multinomial distribution that could be represented as a number of words per topic. Figure 6 shows an example output of the LDA model, which was run on the "mental_health.txt" file with the expected amount of topics set to three. The output was also set to only show the top three words for each topic.

3. Results

3.1. Latent Semantic Analysis

3.1.1. Hyperparameter Tuning.

As discussed throughout the theory section, LSA is a algebraic method of decomposing a matrix representation of the text data. As the foundation of the model is based from a algebraic formula, there are not many parameters which need to be set. The only parameter of interest during implementation of the LSA model was the number of topics to select from the SVD results..

In order to optimize this parameter, with the goal of finding the best summary of each text produced by the model, we implemented an exhaustive grid search to test various numbers of topics. The number of topics tested ranged from 2 to 10. This exhaustive search was done separately on each of the text files during implementation of the model.

3.1.2. Result Interpretations.

In order to evaluate the model, the metric of topic coherence was used as well as our own subjective judgement of which topic looked best. The results of the LSA model will be discussed in terms of these evaluations.

Topic Coherence.

Table 1: LSA model results for "mental_health.txt" for 2 topics.

Topic	Words
Topic #1	health mental disorder problem treatment
Topic #2	disorder anxiety social generalised personality

Table 2: LSA model results for "sleep.txt" for 3 topics.

Topic	Words
Topic #1	sleep people judgment time habit
Topic #2	nap time good sleeping still
Topic #3	asked still sleeping even something

Topic coherence is a metric which can be used to assess the quality of topics produced by the model. The topics themselves are evaluated at an individual level, in order to measure the semantic similarity of the words within [6]. The average coherence score was taken among topics for each iteration. The Gensim KeyedVectors library was used along with a word to vector representation of Google News headlines to evaluate the topics produced by the investigated topics. The Gensim coherence metric was called TC-W2V.

Table 3: LSA model results for ”concussion.txt” for 9 topics.

Topic	Words
Topic #1	brain concussion player athlete year
Topic #2	brain broke boxer injury webster
Topic #3	player first concussion soccer female
Topic #4	game player soccer headache played
Topic #5	year athlete id symptom concussed
Topic #6	like sport play wasnt wanted
Topic #7	play ball head never concussion
Topic #8	back wanted field sport even
Topic #9	could disease body black like

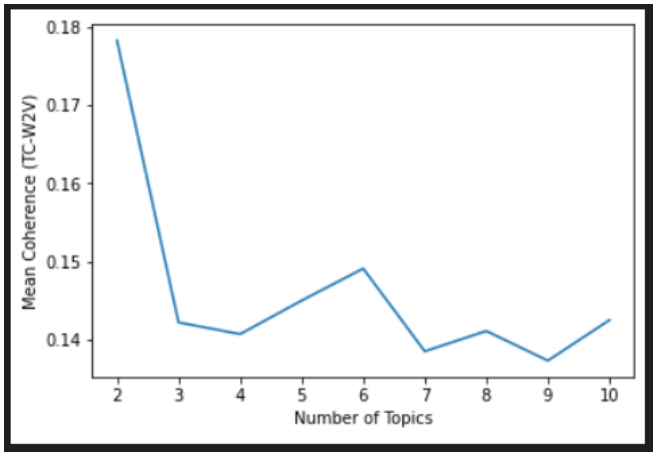

Figure 7: Coherence scores for LSA model on various number of topic parameters, shown is ”mental_health.txt” example. The hyperparameter (number of topics=2) with the highest coherence score (TC-W2V=0.1782) was selected for further investigation of the model.

The output for the LSA model for the three analyzed files are shown in table 1, 2, and 3 respectively for the chosen hyperparameters. Figure 7 shows a visual representation of the coherence scores given hyperparameters for the ”mental_health.txt” file in order to provide a demonstration of how hyperparameters were tested. The hyperparameters which would yield the highest coherence measure would be selected, as otherwise parsing through each run of each model would be very time consuming.

As shown in said tables, LSA had the best performance evaluation for ”mental_health.txt” is 2 topics. Following the same metric, ”sleep.txt” was set to 3 topics, and ”concussion.txt” set to 9 topics.

Subjective Measures.

As previously discussed, each topic output is simply a string of words, assigned to a topic. In this sense, a topic is just a group of words which co-occur in such a way that some significance is extracted by the model. As a human interpreter of this output, these topic assignments are not enough. The final method of model evaluation had a human element, where we as researchers would give names and representation to these strings of words.

The names of the topics would be solely based on the words within. These names or summaries are used to further summarize the text, as there was now a set of topic words used to describe the text. For example, given topic #1 for the LSA's output for ”mental_health.txt” found in Table 1, we may be inclined to assign those words to the topic of ”mental health”, as it contains various words relating to mental health in general. Given topic #2 of that same Table, it may relate to the different forms of mental illness, such as generalised anxiety, social anxiety, and personality disorder.

As previously discussed, humans and computers process text data in very different ways. While computers can execute advanced algorithms and mathematical operations on high dimensional text data, they are unable to understand it in its base form, as a language. Due to this fact, the best summarization model requires some human insight in deciding how well the model is able to capture latent variables, such as the topic of the discussion within the document.

3.1.3. Model Limitations.

The LSA model proposed by Gong & Liu which was discussed throughout the theory section for LSA, has a few known limitations. The most pressing limitation discussed through literature relating to this variant of LSA implementation is that a one sentence summary of a topic may be insufficient to fully show its meaning [10]. Since the development of the variant of Gong & Liu's LSA model, researchers have developed various improvements which allow for compensation of this limitation [10]. Future improvements to this investigation may want to delve into these improvements in order to try to compensate for the shortcomings which are present in the implemented version of LSA.

3.2. Latent Dirichlet Allocation

3.2.1. Hyperparameter Tuning.

As discussed in the LDA theory section, the parameters are to tune the distributions in the first level of figure 3. The first hyperparameter, α, corresponds to the Dirichlet distribution which is used for sampling θ. The second hyperparameter, β is used to tune the distribution from which w_{dn} is sampled from given z_{dn}. The third and final hyperparameter is simply the expected amount of topics, similar to that shown in the LSA model implementation.

Similar to the implementation of the LSA model, we conducted an exhaustive search which iterated through various combinations of α, β, and the number of topics. The possibilities of α and β ranged from various increments between 0.05 and 1. As mentioned, following the diagram of a sample Dirichlet distribution shown in figure 4, sampling from the simplex corners is more likely with these values set to below 1; thus values were chosen according to that principle, as we intended to find documents which represented these topics. Similar to the LSA implementation, the range of number topics tests ranged from 1 to 5. This exhaustive search was done separately on each of the text files during the implementation of the model.

3.2.2. Result Interpretations.

To keep evaluation consistent, the same evaluation criteria was used for the LDA model as it was for the LSA model. The metric of topic coherence was used as well as our own subjective judgement of which topic looked best. The results of the LDA model will be discussed in terms of these evaluations.

Table 4: LDA model results for "mental_health.txt" for 2 topics.

Topic	Words
Topic #1	mental health approach disorder treatment
Topic #2	disorder health system mental anxiety

Table 5: LDA model results for "sleep.txt" for 3 topics.

Topic	Words
Topic #1	sleep people time would one
Topic #2	sleep people time eye really
Topic #3	sleep get early sleeping per

Table 6: LDA model results for "concussion.txt" for 4 topics.

Topic	Words
Topic #1	brain player might symptom first
Topic #2	brain headache study day teammate
Topic #3	concussion brain player cte athlete
Topic #4	year game say played id

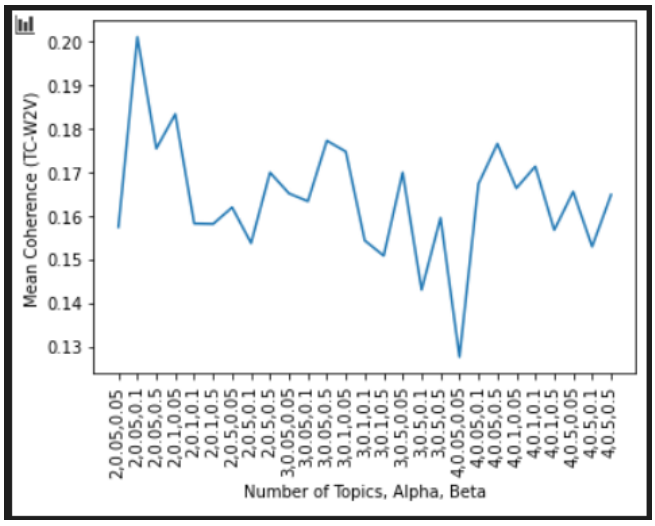

Figure 8: Coherence scores for LDA model on various number of topic parameters, shown is "mental_health.txt" example. The hyperparameter (number of topics=2, α=0.5, β=0.1) with the highest coherence score (TC-W2V=0.2010) was selected for further investigation of the model.

Topic Coherence.

For sake of consistency of evaluation, the LDA model was evaluated using the same coherence metric as the LSA model. Please refer to the LSA topic coherence section for more information regarding this evaluation method.

The output for the LDA model for the three analyzed files are shown in table 4, 5, and 6 respectively for the chosen hyperparameters. Figure 8 shows a visual representation of the coherence scores given hyperparameters for the "mental_health.txt" file in order to provide a demonstration of how hyperparameters were tested for the LDA model.

As shown in the tables, LDA had the best performance evaluation for "mental_health.txt" for 2 topics. Following the same metric, "sleep.txt" was set to 3 topics, and "concussion.txt" set to 4 topics.

Subjective Measures.

In order to keep the evaluation consistent, the same subjective measures were used on the outputs of the LDA models as they were for the output of the LSA models. Please refer to the LSA subjective measures section for more details relating to this subjective measure.

3.2.3. Model Limitations.

Though not as well defined as the limitations of the LSA model implementation, LDA does have some potential limitations present which may provide issues during execution, especially for real world situations. This limitation is the handling of topics both as input and output.

In terms of input, the number of expected topics must be provided as input to the model. This is a flaw within the model as its purpose is to automatically summarize the input documents, but investigators must be aware of the amount of topics within a document in order for the model to run accordingly. There are tuning methods such as those conducted which involve exhaustively testing multiple number of topics, but this amount is very hard to generalize; even with just ranging from the three different topics that were analyzed throughout this assignment. Users of a model implementation such as this may be even more limited by this hyperparameter as they may be processing thousands of documents in a limited amount of time.

As for the output of the model, a multinomial distribution relating to the words which the model has processed to relate to a single topic is somewhat limiting to researchers, as it requires them to deduce the topic meanings themselves. Unfortunately, machines are not able to process text data the same way as humans, and coming up with meanings and definitions to these topics is actually an extremely difficult task [9]. Relating to the input limitation, these topic deductions take a lot of time and energy to come up with, especially in time dependent real world situations. Due to these factors, LDA may not be the most efficient choice for situations such as those where text summarization is required.

As discussed prior, the LDA model implementation also took in text data as input in the form of a bag of words. Due to the nature of this input, the LDA model disregards any context or position of text within the original documents. Due to the intention of implementation of the LDA model, a lack of predefined sentence structure and context may be limiting, especially for documents containing various topics, which one topic may simply have more sentences throughout the text.

3.3. Model Comparison

Table 7: Coherence metric (TC-W2V) comparison between LSA and LDA for analyzed .txt files.

.txt File	LSA Score	LDA Score
mental_health.txt	0.1782	0.2010
sleep.txt	0.1595	0.1699
concussion.txt	0.1531	0.1652

The purpose of this assignment was to ultimately compare two methods of text summarization, now that background of data summarization has been discussed, LDA and LSA models have been implemented, and evaluated; a comparison between the models can finally be made.

As discussed in the evaluation sections, topic summarization such as those shown with LSA and LDA have a crucial human component of deducing which topics belonged to which output sentences and words. As both models have outputted words which are somewhat similar, as shown in tables 1 through 6, it is a difficult task of showing which model performed better. With that being said, there are a few key observations which can be made for the argument of the LDA model, at least slightly, outperforming the LSA model.

First, as shown in Table 7, the average coherence score is higher with the LDA model compared to the LSA model. Although this does not fully represent the models performance, it can act as a supplement to show the quality of the topics, thus summarizations, that the models were processing.

Second, as shown in tables 1 through 6, LDA is able to summarize the documents in less topics compared to the LSA model. This may be beneficial in terms of the objective of text summarization, which is to reduce the time taken to get the main ideas within a document. Due to the more concise output, the LDA model may be on the verge of making the text summarization process more efficient for the user, and thus may be a more robust model of choice compared to LSA.

Lastly, the limitations stated in the LDA model may be less detrimental to the success of the model, compared to the limitations of the LSA model. In reference to the limitation section of the LSA model, if a model is not outputting enough information for an effective text summarization, than the robustness of that summarization model may be in question.

Overall, given the nature of the current literature of text summarization, it is an extremely subjective process to compare each model. This comparison ultimately depends on the parameters and text chosen to be processed, as well as the limitations of the current implementations of the models. According to this subjective process in terms of the theory, implementations, and evaluation metric, LDA was deemed as the more robust model.

4. Conclusion

Text summarization is becoming more and more important with the ever growing nature of the Internet. Throughout this assignment, we discussed various text summarization techniques, before choosing two to implement. Latent semantic analysis and latent Dirichlet allocation were discussed, implemented, and evaluated, as both models have the capability to detect the significance of underlying themes and topics of input documents. The processed documents were related to mental health, sleep, and sports concussions. Through means of a topic coherence metric and subjective analysis, LDA was deemed the more robust model due to it producing better results according to evaluation as well as having less detrimental limitations.

In terms of the assignment objective, given the choice, we would recommend the latent Dirichlet allocation model over the latent semantic analysis model.

References

[1] https://aeon.co/

[2] Allahyari M., Pouriyeh S., Assefi M., Safaei S., Trippe E. D., Gutierrez J. B., & Kochut K. (2017). *Text summarization techniques: a brief summary.* arXiv:1707.02268.

[3] Alpaslan F. N., Cicekli I. (2011). *Text summarization using latent semantic analysis.* Journal of Information Science. 37(4):405-417. DOI: 10.1177/0165551511408848.

[4] Blei D. M., Ng A. Y., & Jordan M. I. (2003). *Latent dirichlet allocation.* Journal of Machine Learning Research. 3: 993-1022.

[5] Dunning T. (1994). *Accurate methods for the statistics of surprise and coincidence.* Computational Linguistics. 19(1):61-74.

[6] Fang A., MacDonald C., Ounis I., & Habel P. (2016). *Using word embedding to evaluate the coherence of topics from twitter data*. SIGIR 2016. https://doi.org/10.1145/2911451.2914729.

[7] Gong Y., & Liu X. (2001). *Generic text summarization using relevance measure and latent semantic analysis.* SIGIR 2001. https://doi.org/10.1145/383952.383955.

[8] Griffiths T. & Steyvers M. (2007). *Probabilistic topic models.* Handbook of Latent Semantic Analysis. 427(7):424-440.

[9] Luhn H. P. (1958) *The automatic creation of literature abstracts.* IBM Journal of Research and Development. 2(2): 159-165.

[10] Nenkova A. & McKeown K. (2012). *A survey of text summarization techniques.* Mining Text Data.

[11] Salton G., Buckley C. (1988) *Term-weighting approaches in automatic text retrieval.* Information Processing & Management. 24(5):513-523.

[12] Singhal A. & Google Inc. (2001). *Modern information retrieval: a brief overview.* IEEE Big Data Eng. Bull. 24(4):35-43. doi=10.1.1.505.313.