Utrecht University

Master thesis Artificial Intelligence

A Socratic Principle Based Method For Fighting Fake News

Author:
D.H.W. Lam *(4298772)*

Supervisor:
Prof. dr. Y. Velegrakis

Second examinator:
Dr. A.A.A. Qahtan

December 19, 2020

44 ECTS

Universiteit Utrecht

Abstract

Misinformation and fake news are a typical problem for media that can lead to negative consequences in society. This work represents research into a solution that indirectly fights misinformation by introducing different perspectives through opinions. The goal is to crumble information bubbles and narrowed-minded views and create a neutral information environment. We developed a system that searches and returns different opinionated documents given a reference document. The system extracts so-called sentimented topics that hold information about the sentiment score for a certain topic by utilizing topic modeling and sentiment analysis. Additionally, we introduce a search algorithm to find the best combination of k sentimented topics that creates a neutral informative setting based on distance values.

To analyze the system, we used a collection of text documents related to the subject COVID-19, which we extracted from the web. We evaluated the performance of the proposed system and the quality of the output of the system. Results showed that the components responsible for preprocessing and selection of sentimented topics are computationally expensive. The proposed system does return text documents that are both related and bear different sentimented topics compared to the reference document. Although the system has some flaws, we believe this system could serve as the basis on which to create a neutral informative setting.

Contents

1 Introduction

Fake news and misinformation are quite problematic in society and can lead to some severe consequences. One of the recent topics that was the target of fake news is COVID-19. False information about how to cure this pandemic and false sensational stories were spread on social media [1]. These kinds of misinformation can stir up unnecessary panic and anxiety within crowds. One of the effects is hoarding food or sanitary supplies. There were scammers who exploited hoarding by selling face masks or hand sanitizer gels for extortionate prices as a result of misinformation [2].

"Fake news" is defined as fabricated information without credible sources that mimic news media content according to the work of Lazer et al. [3]. False news stories can have a negative impact, such as undermining the credibility of (traditional) news outlets and damaging a person or (official) organization. Due to the accessibility of the internet, it is easier to create and publish content for people by people compared to the past.

The work of Vosoughi et al. [4] showed how true and false news was spread on Twitter. They analyzed fact-checked rumor cascades. A cascade was defined as an assertion about a topic by the number of retweets by other users. Rumor cascades were checked for depth, size, maximum breadth, and virality/trend over time. The results showed that false information was diffused significantly faster and farther than the verified information. They expected one of the possible contributing factors for the spread to be the work of social bots or great network connections. However, this was not the case. Instead, they discovered that it was human behavior that contributed most to the spread of false information. They showed that people may often share articles without reading the content, just the headlines.

One way to counter misinformation is by evaluating fake news through fact checking. Fact checking is a process that determines the accuracy and veracity of nonfictional texts (e.g. news articles, social media posts). The main steps consist of identifying the claims, finding supporting or refuting evidence, analyzing the evidence, and delivering a verdict about the claim in question. News outlets are expected to fact check their stories before they are published. This task is also performed after relatively important representatives, e.g. politicians, made a statement or gave a speech. For example, FactCheck.org is a non-profit "consumer's advocate" for voters in the U.S that fact checks information in politics, such as debates or speeches by analyzing, writing and editing articles regarding selected claims. This is done manually by journalists, editors and writers.

Another organization is Full fact [1]. Full fact is an independent fact checking organization that checks news media. They are currently building robust and scalable fact checking tools to assist journalists. This is necessary because of the sheer volume of information and articles being generated. Their main tools are called "Live" and "Trends". "Live" automatically spots claims in TV subtitles and fetches the most recent articles or information that matches with the claims. This tool fact checks on the spot using reliable data available at that moment. The tool "Trends" records the repetition of inaccurate claims and their sources to keep track of where the misinformation originated from [5]. Machine learning and artificial intelligence contribute to improving the process of fact checking, but research regarding automated fact checking is ongoing.

Although fact checking helps to interfere with the stream of fake news, cognitive biases might hinder the effects of the process. Lazer et al. [3] described that people prefer information that aligns with their perspectives, beliefs, or attitude. These are characterized by selective exposure, confirmation bias, and desirability bias. Repeating false information also gains their perception of truthfulness. The effect of repeating false information was shown in the work of Ciampaglia [6]. Due to algorithmic bias by big data filtering and ranking algorithms, people fall into information bubbles based on their preferences. These information bubbles create an echo chamber where one's prior belief is reinforced. This lowers the individual's guard for misinformation due to repetition. Under these circumstances, that leads to tunnel vision, so fact-checking becomes counterproductive. As people are more reluctant to accept the facts given by fact-checking systems as evidence and will still believe the incorrect information even after the verification. People with a strong prior belief or view will reject correct evaluations from fact-checking systems or news organizations.

[1] https://fullfact.org/

A narrow-minded mindset contributes to prejudice and is prone to regard other perspectives as flawed and wrong. This poses a big problem.

Therefore, in this research, an approach is proposed that indirectly fights information bubbles, without directly verifying the information as true or false.

1.1 Motivation

The goal is to crumble information bubbles and narrowed-minded views and create an information environment by showing opinion pieces or opinionated articles. An opinion-based article reflects the author's opinion on a subject or topic. The purpose of such an article is to show how one or multiple topics are viewed by a particular group and what the reasoning is behind the view. We want to introduce different opinions or perspectives of a topic to a person who might not discover the viewpoints at all.

Figure 1 shows a chart of different American media that are categorized on the political spectrum based on the perspective presented in their articles. Due to the differences in political beliefs, articles between left and right media outlets are bound to be different.

Figure 1: The chart shows the media bias on political spectrum for different news media agencies. The image originates from https://www.allsides.com/media-bias/media-bias-chart.

Showing different articles from each media agency with a certain viewpoint on a certain topic could create a neutral informative setting, which helps to see an issue or problem from different perspectives. This gives insight and helps the person to be more knowledgeable on a certain topic. It mitigates biases as the person has more information on a certain topic to adjust or correct his or her opinion.

1.2 Our solution

This work introduces a system and presents an algorithm that will choose articles with different opinions. The system differentiates documents with different opinions and selects a number of documents based on a reference document. The objective is to introduce the user to other opinions from different viewpoints and present an informative situation. Presenting a document with a similar opinion might enhance their prior beliefs, while a document that contrasts greatly might be unlikely to be accepted. The process of the system and selection procedure will be explained and evaluated.

The difference between this solution and fact checking is that our algorithm does not evaluate whether a claim or reasoning in question is accurate or false. It will only provide documents, each with different levels of opinion on a topic and does not give a verdict on whether an opinion or statement is factually correct. Showing articles with different opinions is information that is less likely to be rejected by the user compared to telling them that their opinions or prior beliefs are incorrect due to misinformation. The scope of the project will be the development and execution of such an algorithm. The effects of using it, especially on a social level, will not be covered in this thesis.

1.3 Relation to Artificial Intelligence

This research project is related to artificial intelligence, as we give a computer the intelligence to understand opinion-based articles, to help us with the task of information discovery. The program needs to discover other documents that contain different user opinions given a reference document. Our system has to understand which opinions are present in the documents and what sentiment an opinion holds. To achieve this, our system utilizes techniques from Natural Language Processing, which is a subfield of AI. These techniques include text processing, sentiment analysis, and topic modeling. Sentiment analysis ensures for our machine to understand the sentiment of the text or opinion, and topic modeling shows which topics are covered. They are crucial to our research as they form the fundamentals of the system.

1.4 Thesis organisation

The remainder of this thesis is structured as follows; Section 2 provides a motivating example, Section 3 summarizes the related work, Section 4 describes the problem statement, Section 5 describes the proposed solution, Section 6 describes the implementation of the proposed solution, Section 7 describes the experimental evaluations and Section 8 presents discussions and conclusion.

1.5 Motivating Example

John is a user who is quite worried about the COVID-19 virus that is spreading through his hometown. He decided to look for more information about this virus. He queries the subject COVID-19 on Google search system and sees multiple web pages leading to more information about COVID-19. During the search process, he has read multiple online articles regarding the situation of wearing a face mask and health concerns of the virus and has formed his own opinions and prior beliefs regarding those topics. John thinks that face masks does not protect him from COVID-19. He comes across other web articles that share the same opinion as John, such as the following example:

> *Mask wearers frequently report symptoms of difficulty breathing, shortness of breath, headache, lightheadedness, dizziness, anxiety, brain fog, difficulty concentrating, and other subjective symptoms while wearing medical masks. As a surgeon, I have worn masks for prolonged periods of time in thousands of surgeries and can assure you that these symptoms do occur when surgical masks are worn for extended periods of time.*

Since his opinion is shared by multiple articles, John believes that his opinion is correct. However, this only shows one aspect of the topic and does not represent the whole truth about face masks. John likely received only web pages sharing opinions similar to his opinion because the search system returned results that are related to his search history. Hence, John is in an information bubble. He could try to find and read different perspectives about face masks, but this takes time and effort.

We want to show John other documents that take a different position on the matter of face masks. One user opinion is that a face mask creates a false sense of safety, thus people forget to keep a distance of 1.5m. Another opinion is that a face mask is not to protect yourself from getting COVID-19, but to prevent the spread of COVID-19 if you have the virus unknowingly. The problem is how to identify these documents with different opinions and which documents are going to be shown to John. The text in the previous box is used as a reference document that reflects or represents John's initial opinion as a starting point.

We need to extract that the topic of the text example is about face masks and their effects. In addition, we also need to extract that the sentiment of the text example is negative. Therefore, if the topic of the document is 50% about face masks, 50% about side effects and it is negative about the face masks, then it would be nice to find documents that consist of different percentages of topics e.g. 30% about face masks, 30% side-effects and 30% effectiveness with a neutral opinion. For example, the following text documents give other perspectives regarding face masks.

> *People who wear face masks do not suddenly have a false sense of security, prompting them to ignore social distancing. It is not masks, but crowds that make people not keep their distance from each other.*

> *Surgical masks may protect others by reducing exposure to the saliva and respiratory secretions of the mask wearer. Yeah, that might be good, but there is limited data on whether face masks are actually effective.*

> *Wearing masks may be difficult for some people with sensory, cognitive, or behavioral issues. If they are unable to wear a mask properly or cannot tolerate a mask, they should not wear one, and adaptations and alternatives should be considered.*

It must also be taken into consideration which documents with a specific opinion are shown as information to John. John already has a negative view of face masks. If he receives a document that has extremely positive opinions about face masks, he will likely reject the information from that document. The contrast between his opinion and the user opinion of the presented document would be too great. This effect also applies in another situation where John receives 2 documents in which face masks are viewed negatively, but 10

documents that are positive to face masks. This will give an impression to John that the information setting is biased to be positive about face masks, which results that John will not accept the skewed information results.

Furthermore, documents that share the same opinion should not be shown to John as it enhances his prior beliefs. Showing the same information or opinion does not contribute to providing a global view on a topic. Besides the topic face masks, there are other topics that are associated with face masks or COVID-19. It is helpful that other topics such as handgels or travel also assist in giving information regarding face masks in COVID-19. One user opinion tells that when you are traveling in public places, that you should always put on a face mask. Another user opinion says that handgels are doing more harm than good, encouraging people to use face masks. Documents containing these user opinions are useful to show to John.

The question is thus how these documents with different opinions can be found and selected to create a neutral and informative setting. Humans can comprehend text and identify emotions and sentiments. They are able to judge which articles are related and if they bear the same opinion or not. Needless to say, this task poses a challenge for a machine as it does not have the ability to comprehend text. The ability to understand a text, differentiate user opinions, and select the right number of documents needs to be translated by a machine to solve the challenge.

2 Related work

In order to create a system that indirectly fights the information bubbles that are susceptible to misinformation, we need to understand which techniques are involved in fact-checking and how it directly fights misinformation and fake news. This gives an insight into which techniques are crucial for information retrieval and how the system differs from the process of fact checking.

2.1 Fact checking

Traditionally, fact checking is a task in journalism to check the correctness of facts in news articles before they are published. This process refers to an analysis of a claim after it has been published in online articles, tweets, or digital media. Due to the accessibility of the web, it is easier for users to create and publish content through social media. However, this also leads to content containing misinformation or intentionally directed harmful content that can be seen by others. According to the work of [7] and [8], the process of fact-checking consists of the tasks; claim extraction, searching facts related to the claim, claim verification, and providing perspective to claims if needed. Our proposed solution is quite similar to this process. There are some differences, which will be elaborated later.

2.1.1 Claim extraction

Claim extraction involves finding claims in text that are check-worthy. There are several methods in which this task can be performed.

Hassan et al. [9] introduced a supervised approach for detecting check-worthy factual claims in transcripts of presidential debates. Each sentence can be classified in one of the three potential labels;

- Non-factual sentence (NFS) e.g. subjective, opinionated or question sentences.
- Unimportant factual sentence (UFS) e.g. factual claims in which the general public is not interested whether this claim is true or false.
- Check-worthy factual sentence (CFS) e.g. factual claims in which the general public is interested whether this claim is true or false.

Their goal is to automatically detect the CFS apart from NFS and UFS through multi-label classifiers. They acquired their dataset from presidential debate transcripts with a total of 1571 labeled sentences (882 NFSs, 252 UFSs, 437 CFSs). A total amount of 6201 features regarding sentiment score, word count, words, entity type and Part-of-Speech (POS) tags were retrieved, which was reduced to 30 important features with the help of random forest and GINI index for each classification tree. They compared the performance between Multinomial Naive Bayes classifier (NBC), Support Vector Machines (SVM) and Random Forest classifier (RFC) on different combinations of features. The RFC classifier, trained on only words as features achieved a precision of 71% and 62% F-measure, while the SVM classifier trained on the combination of words and POS tags achieved a 70% precision and F-measure.

A subsequent research paper by Hassan et al. in [10] explained the claim detection component of the system ClaimBuster. The methodology is relatively similar to the research in the previous paragraph with an addition that sentences will also be ranked on how important they are based on the probability equation $score(x) = P(class = CFS|x)$ with SVM. The dataset consisted of 8231 sentences from transcripts of political debates. Overall, the SVM was the best performing classifier. The ClaimBuster system can be found online [2].

Lippi and Torroni [11] created a context-independent claim detector by creating an SVM classifier for the similarities between parse trees through Tree Kernels (TK). Their methodology is based on the idea that a sentence containing a claim is characterized by rhetorical structures. By using the kernel over parse trees and feature vectors, they train the SVM classifier with a set of labeled examples. On the test set of 200

[2] https://idir.uta.edu/claimbuster/

sentences, the classifier achieved 9.8% precision, 58.7% recall and 16.8% F1. However, the dataset they used for testing and training were context-dependent.

Hansen et al. [12] presented an end-to-end trainable neural network for ranking sentences on check-worthiness. The sentences are represented by word embeddings and syntactic dependencies to capture the semantic in context. Their main idea was that common top-weighted words may be distinguishable by their syntactic role. They used the public API of ClaimBuster to weakly label the input sentences with a degree of check-worthiness continuous score between 0 and 1. Three datasets were used for training domain-specific embeddings, comparing the performance against the baseline without weak supervision and with weak supervision. The neural network with weakly labeled dataset achieved 30% precision, while the performance without weak supervision achieved 27.8% precision.

Another research by Patwari et al. [13] introduced TATHYA which is a multi-class SVM system. The dataset consisted of sentences from primary and presidential debate transcripts. The training data is divided into k groups, where each group is used to train a classifier. The classifier with the highest confidence for each data group will then be used for the prediction task. The output of the classifier with the highest confidence will be used to predict if a statement is check-worthy or not. They used the features: entity history, POS tuples, Bag-of-words, topic agreement and normalized text. The authors also experimented that $k = 3$ groups resulted in a consistent performance which is implemented in TATHYA-MULT. TATHYA-SVM achieved 20.9% F1, while TATHYA-MULT achieved 21.4% F1.

2.1.2 Finding related source information

To verify statements or claims, evidence has to be found for this task. This task consists of finding relevant documents or information. Fact-checking systems can make use of different (external) sources to check the veracity of claims.

The claim checker component of ClaimBuster in [10] utilizes knowledge bases such as Wolfram Alpha and Google search for collecting evidence. The factual claim is sent as a general search query in Google. The component parses the search result and downloads the web page of the top results. Sentences within the web page that matches with the claim and answers of the Wolfram Alpha will be collected and grouped as context to determine the accuracy of the claim. Another method to find information is by searching if the claim has already been checked in the past by using repositories of several fact-checking websites.

Wang et al. [14] provided an end-to-end system to automatically discover documents that are relevant to fact-checking articles in question. According to their research, fact-checking articles from fact-checking organizations can be identified by the schema.org ClaimReview markup which provides a summarizing overview in the key fields claim, claimant (the person or organization making the claim) and verdict. The system consists of 3 components for the task candidate generation, relevance classification and stance classification. The goal of the candidate generation component is to identify as many documents as possible. This part collects outgoing links and source articles from the fact-checking article and utilizes Google search to fetch more relevant documents with the following type of queries:

- Title text and claim of ClaimReview markup
- Entity annotated title and claim text
- Click graph queries associated with the fact-checking article

For a query, the top 100 results are collected in which duplicate documents will be removed. The relevance classifier predicts whether the fact-checking article and documents are relevant in order to filter irrelevant documents. Using information such as title, headlines, selected sentences, paragraphs and annotated entities with confidence score, the component extracts similarity scores regarding:

- Entity
- Core text
- Claim-to-sentence

- Claim-to-paragraph
- Sentence
- Content

The similarity scores are calculated using the cosine similarity between text embeddings. These scores and differences in publication days were used as features for the classifier. Their last component stance classification determines whether a relevant document support or contradicts a claim. This component divides the documents into contradicting or supporting documents. For the experiment, they used a manual corpus with 450 fact-checking articles and 4000 pairs of a fact-checking article and relevant document. The candidate generation component achieved an 80.0% recall. The relevance classifier achieved 81.7% accuracy and the stance classifier achieved a 91.6% accuracy.

Karadzhov et al. [15] also made use of a search engine (Google and Bing) to retrieve web pages or snippets of text. They generated a relatively short query by ranking the words with term frequency-inverse document frequency (TF-IDF) and considered only the nouns, verbs and adjectives within the claim and named entities. After they collected web pages and snippets, they calculated 3 similarities; cosine with TF-IDF, cosine over word embeddings and containment similarity, between the claim and snippets or claim and web pages. The best-scoring snippet, best-scoring sentence triplet and embeddings of the claim were used as features for the neural network and/or classifier.

In other works, [16] [17] [18] the authors utilized Knowledge Graphs (KG) using databases such as DBpedia, PubMedDB or Yago to verify claims. The KGs contain factual statements that are represented by semantic triples and predicates. With this approach, a fact-checking system only needs to mine an existing path or shortest path between entity nodes to retrieve relevant information or evidence. However, this approach would make it rather difficult to create an informative setting for a user by using only predicates and triples without context.

2.1.3 Claim verification

Claim verification is a crucial step in fact-checking whether the veracity and/or accuracy of a claim is justified. After evidence or relevant source material has been collected, a fact-checking system can check a claim in several different methods.

ClaimBuster [10] uses two methods to verify a claim. One method is to make use of the fact-checking repositories containing fact-checked claims and find a matching result to the claim in question. The other method is using the collected results of Wolfram Alpha and Google search. If discrepancies exist between the context and a claim, a verdict may be derived and presented to the user.

Karadzhov et al. [15] used neural networks and SVM to classify the veracity of a claim by using word embeddings and similarities from the claim in question and relevant information sources. The methodology has been explained in the previous paragraph.

Fact-checking systems that utilize a KG, check whether there exists a path in the graph between the entities that are contained in a claim. Shi and Weninger [16] view fact-checking as a type of supervised link prediction problem. The validation of a fact is determined if this is implied in the data within the KG. In the work of Ciampaglia et al. [17], they describe that a fact is true if there exists an edge or short path linking to its subject and object within the knowledge graph. Semantic proximity is used to calculate the shortest path to determine the truth value of a statement of fact. In the work of Fionda et al. [18], an evidence graph is built which contains supporting or disproving information for the given claim. An evidence score is calculated using a similarity function based on how often node S and O are linked and how many other entities are connected to S and O.

While claim verification is important in the process of fact-checking, the focus of our system is not to evaluate a claim or statement whether it's true or false, but to provide an informative environment. The user will be presented with multiple articles containing relevant topics that shed another perspective towards a subject. By showing multiple perspectives or different articles, the user adjusts or deviate their opinion and question the veracity of claims that are made in e.g. online news articles. Per definition, this project is not performing

fact-checking, but similarities between the tasks of fact-checking except claim verification and this research project can be found.

2.2 Latent Dirichlet Allocation (LDA)

The system must be able to identify the content of a text document. Finding words that are deemed important in a document would give too little information to summarize the content. We need to find sets of words that indicate information for a general subject. Therefore, we take a look at existing work of the technique of LDA as it detects underlying topics using unsupervised learning. Categorizing documents using common topics gives a better overview of which documents share a similar topic or are different from others.

LDA is a topic modeling technique introduced by Blei et al. for text modeling [19]. They described that LDA is a generative unsupervised topic detection technique which can discover a pre-defined number of k topics from a corpus. The fundamental idea is that LDA assumes that a document, which is a collection of words, contains a mixture of k underlying topics in which the topics are represented by word probabilities in which the probabilities are sampled from the Dirichlet distribution.

Blei et al. measured the performance of their LDA topic model by using the perplexity, which indicates how well the model can predict a sample.

Jelodar et al. [20] surveyed the applications and models of LDA and topic modeling. LDA is one of the popular methods in topic modeling, which has been applied to multiple area disciplines. In political science, LDA has been applied e.g. to extract the topics from speeches of politicians which can provide information for political priorities and to discover topics for contrastive opinion mining. In bio- and medical science, LDA has been applied to obtain latent topics from biological terminology or to detect hidden patterns of internal treatment in Clinical processes. Other methods that have been used to estimate the LDA parameters, inference and training are e.g. Gibbs sampling, Expectation-Maximalization and Variational Bayes Inference.

Mei et al. proposed a probabilistic mixture model named Topic-Sentiment Mixture (TSM) to not only analyze latent topics but also find the associated sentiment from a collection of Weblogs [21]. They assume that each document contains a number of k major topics and that a weblog contains one positive and negative sentiment polarity. They generated topics and sentiments using a mixture of multinomial distributions.

- θ_B named the background topic model to capture common English words such as "the" or "a".
- θ which is the set of k topic models.
- θ_P is the positive sentiment model for finding the positive opinions
- θ_N is the negative sentiment model for finding the negative opinions

The words that are related to a topic will be further categorized whether they're representations of neutral, positive or negative opinions. The topic and sentiment are presented as a relative coverage in a form of a probability distribution which satisfies (specify deltas) $\delta_{i,d,F} + \delta_{i,d,P} + \delta_{i,d,F} = 1$, named as sentiment coverage. The variable $\delta_{i,d,N}$ represents the coverage of neutral opinion, $\delta_{i,d,P}$ for positive opinions and $\delta_{i,d,N}$ for negative opinions. Furthermore, they introduced two additional concepts to see the changes of intensity in the distribution of a topic and sentiment over time. For estimating the sentiment model priors, they used Opinmind, a blog opinion search engine to retrieve positive and negative sentences for an arbitrary topic. The results of the TSM were used for summarizing search results, predicting human behaviors and monitoring public opinions.

Besides the perplexity metric to evaluate the performance of an LDA topic model, Röder et al. introduced a unifying framework that combines topic coherence measures to evaluate a topic model on human interpretability of topics [22]. Their framework consists of 4 parts:

- Segmentation of word subsets

- Probability estimation; the word probability is determined by the number of documents that the word has appeared divided by the total amount of documents. Other methods to estimate probabilities are based on the number of sentences, paragraphs and window sliding for the number of words in a sentence. The component uses a reference corpus to estimate the probabilities.
- Confirmation measure; this component measures the semantic support between pairs of words.
- Aggregation; this part aggregates the values of the confirmation measure into a single coherent value.

Their framework covers the following coherence measures:

- C_{UCI} is based on the pointwise mutual information (PMI) in which the probabilities are estimated on word co-occurrence counts.
- C_{UMass} is based on the word co-occurrences counts (w_i, w_j) divided by the occurrences of w_j.
- C_{NPMI} is an enhanced version of the C_{UCI}, but uses the normalized PMI.
- C_V is a measure of the combination of indirect cosine similarity measure with the NPMI and sliding window over a text document.

The LDA technique will help to uncover the underlying topics covered by multiple documents. It will help to identify which words are relatively important within a topic group by their probabilities and which words are coherent to form an arbitrary underlying topic. The performance of a topic model will be measured using the coherence measures of Röeder et al. [22]. The measures will assist in choosing the topic model with the best hyperparameters to find the topics.

2.3 Sentiment analysis

An opinion consists of a sentiment whether it's negative, neutral or positive to a certain subject or topic. To indicate the sentiment of an opinion, we look at the technique of sentiment analysis. Sentiment analysis helps to quantify and classify the sentiment, which helps to identify the overall sentiment of an opinion from a text document needed for our system.

In the work of Mohammad [23], sentiment analysis is a technique to identify and determine one's attitude towards a topic or target from a text. This task can be applied to measure the general opinion in elections, developing a dialogue system to handle queries or complaints, etc. Some tasks in sentiment analysis concern detecting sentiment from speech or textual chunks such as chat messages or reviews. Wang et al. [14] used in their work stance detection, which is related to sentiment analysis, to check if a relevant document supports or contradicts a claim by observing the titles and headlines of articles. They've built and trained a classifier model with a lexicon of unigram words that detects contradicting documents. A relevant document can have the following stances: {supporting, contradicting}. Stance detection consists of determining the sentiment towards a target which may not always be explicitly given. This task has been applied e.g. on Tweets or online debates. Mohammad et al. [24] used a linear-kernel SVM model trained on several features such as n-grams, sentiment lexicon features and target features to predict stances of Tweets. A tweet can be classified with stance categories {Favor, Against, Neither} given the target entity.

Yuan et al. [25] focused on answer stance detection to determine the stance or attitude towards a target or entity within a question. They used a recurrent conditional attention (RCA) model which is trained on Q&A data and word embeddings to determine which words in questions or answers have a greater influence towards a stance category {favor, against, neutral} instead of hand-crafted features.

In the work of Bar-Haim et al. [26], they provided a benchmark dataset and proposed a claim stance model predicting whether a claim is against or in favor of an open-domain target. Claim target identification is done by checking candidate targets nouns in claim sentences:

- through syntactic and position in the claim sentence
- whether target noun is a title of a Wikipedia article
- whether the sentiment of claim sentence can be connected to target noun

- how much semantic similarity is between topic and target noun

The model made use of terms in the sentiment lexicon, sentiment shifters and sentiment term weights and scores for claim sentiment calculation. The stance relation between a claim and topic is calculated by the following formula:

$$Stance(c,t) = s_c \times R(x_c, x_t) \times s_t$$

The claim sentiment s_c , which tells the sentiment of the claim towards its target, is a score between {-1,1}, where -1 denotes negative sentiment and 1 denotes positive sentiment. The contrast relation between topic target and claim target denoted by $R(x_c, x_t) \in \{-1, 1\}$, where -1 denotes contrastive topics and 1 to consistent topics. Topic sentiment s_t consisted of a topic and sentiment $\{Pro, Con\}$. The stance formula would return binary output {-1,1} for con and pro respectively. However, they also build a continuous model in which the values of the parameters ranged between [-1,1].

For creating a neutral informative environment, it is important to know the position of a document, whether it is positive or negative towards a target topic in order to find and select the appropriate documents.

2.4 Document clustering

Since the system needs to work with a collection of text documents, it is important to consider which documents should be considered to be returned as results. The system needs to categorize the documents based on sentiment and topics. We look at existing work for the technique document clustering to get an understanding how this can be achieved.

According to the works of Saxena et al. [27] and Xu and Tian [28], clustering is am unsupervised method in which unlabeled data is grouped through similarity into clusters. The aim is to find natural groupings of data based on similar patterns. The process contains extracting and selecting features, applying the clustering algorithm, evaluate the clusters and optionally explain the cluster results. Clustering can be categorized into hierarchical or partitional clustering. In hierarchical clustering, clusters are built in a hierarchical approach. It can be built top-down (divisive) by starting with one cluster and iteratively divide it into smaller clusters or bottom-up (agglomerative) by starting with a single object and merge atomic clusters into larger clusters. In partitional clustering, data are assigned through criterion functions into k-clusters in which the center of the data points serves as the center of the cluster. A common criterion function is a Euclidean distance in which a data point is assigned to a cluster with the minimum distance between point and cluster.

Naik et al. [29] conducted a survey on semantic document clustering. This process consists of dividing a collection of text documents into category clusters in which documents within a cluster are semantically similar, but documents from distinct clusters are dissimilar. Traditional document clustering is done by using Bag-of-words and term frequency weighted model for document representation. However, the drawbacks included that it ignores semantic relation between words and dissimilar clusters cannot be identified. The semantic approach of clustering was described that it made use of concept weighting with e.g. ontologies, semantic graphs or WordNet. In their work, they summarized 17 research paper regarding the process of semantic document clustering and analyzed the advantages and disadvantages for each tool and algorithm has been used. In the end, they proposed an architecture for semantic document clustering based on their survey. Their proposed architecture consisted of the following items:

- Document pre-processing: Necessary steps involves tokenization, stemming and stop words removal
- WordNet ontology mapping: Find the frequency of keywords, use TF-IDF equation to calculate the weights and map the weights to Wordnet concepts.
- Clustering algorithm: Bisecting K-means, hierarchical agglomerative clustering (HAC) algorithm and Self Organized Map (SOM) neural network
- Similarity measure: Cosine similarity

Bafna et al. [30] utilized TF-IDF to extract relevant labels or words from a set of documents. After preprocessing documents by removing stopwords and using stemming list e.g., they applied TF-IDF to

retrieve the term-document frequency and calculate the cosine distance matrix. Furthermore, they used HAC and fuzzy k-means to form clusters and compared the performance by measuring the Entropy and F-measure. Using the datasets; News 20, Reuters, Research papers and E-mails, HAC performed well on the documents on datasets News 20 and Reuters, while fuzzy k-means performed well on others.

Another way to give structure to unstructured data is by applying the method facet extraction from faceted search. According to Zheng et al. [31], faceted search is an interactive and heuristic refinement search paradigm in which search results can be narrowed down by facets terms. Facet extraction is one of the key components of a faceted search system. This is usually done by using the statistics of the term, linguistic features or external knowledge bases. After the facet terms have been found, a hierarchy needs to be constructed to discover the "is-a" or "is-part-of" relations between facet terms. For this step, clustering or pattern-based methods are usually used. Clustering-based methods exploit semantic similarity or semantic distance between concepts to form a cluster. Each cluster can be seen as a facet term. Pattern-based methods use e.g. co-occurrence facet terms or existing hierarchical relationships in a semantic database to form hierarchical structures. Furthermore, If there are too many facets, facet ranking is applied to show only the important terms. This technique is useful to organize articles in a systematic manner.

Dakka et al. [32] presented an unsupervised technique to extract useful facets from free-text. The idea behind their approach is that high-level facet terms rarely appear in documents, thus externals resources such as Wikipedia are needed to extract context terms. Their algorithm first identifies the important terms within the document that characterizes the content of the document by using Named entities, Yahoo term extraction and self-developed Wikipedia term tool by checking its wiki pages and hyperlinks. The identified terms were queried into external resources to retrieve related documents to derive more context terms based on co-occurring or frequent terms. At last, a comparative term frequency analysis is performed to extract facet terms based on the intuition that facet terms are infrequent in the original document, but frequent in the expanded setting. They checked the difference in frequency and whether the difference was significant.

Li et al. [33] proposed Facetpedia which is a faceted search system over Wikipedia. The system made use of the user-generated collaborative vocabulary such as hyperlinks and category system to extract facets. Usually, the facets are associated with a hierarchy of categories that can reach target articles through associated articles by hyperlinks. After a keyword query search returned result articles, Facetpedia constructed a facets hierarchy using an algorithm that only considered safe-reaching and relatively low navigation cost facets. A user navigation model is implemented to make the exploration of target articles for the user more convenient. However, in the discussion, they argued that hyperlinks are not always good features in the sense that authors can refer to hyperlinked articles to explain an entity and that a hierarchy structure based on Is-A relationships might result in more meaningful facets than the Wikipedia category system.

3 Problem Statement

In this work, we assume that there are infinite words. A document is a sequence of words. A collection of documents is denoted as D.

Definition - Topic A topic is a set of pairs of consisting of word and probabilities $\{(w_1, p_1), ..., (w_n, p_n)\}$.

We assume that a document has at least one topic of what the text is about. A topic is a cluster of words with probabilities of how likely the words belong in the clustering.

Definition - Sentiment Score The sentiment score S is a value between [-1, 1].

A sentiment score of -1 means that the attitude is very negative for a subject or object, while a score of 1 is very positive. The sentiment score is attached towards a topic t is denoted as S_t.

Definition - Sentimented topic A sentimented topic is a pair (T_t, S_t) consisting of the topic T_t and attached sentiment score S_t.

A sentimented topic represents the degree of sentiment towards a topic T_t. A document contains one or more topics with a degree of sentiment for each topic, in other words, a document contains at least one sentimented topic and a max of n sentimented topics. Intuitively, the set of sentimented topics forms a user opinion. The sentimented topics summarize the content and opinion of a text document through their topic and sentiment score.

The goal is to find documents that contain different sentimented topics and form a neutral information setting. A user sees a document, which we call the reference document d_r, that contains a certain sentimented topic. We want to indicate other documents that are related to the reference document d_r. For this reason, we consider the sentimented topic of d_r as a query.

Definition - Query A query q is a sentimented topic.

The query acts as a reference to find related sentimented topics that indicate which documents are to be returned. It is not desired to select sentimented topics that are too similar to the query or too far from the query.

To find the related documents, we use the query to find k sentimented topics that create an informative setting for the user. Therefore, the set of sentimented topics must be in balance with each other. When these sentimented topics are found, the documents containing one of those sentimented topics will be returned to the user.

The search for the k sentimented topics is presented as a search problem in an n-dimensional space. Every sentimented topic and the query are transformed into a point in this n-dimensional space. Every dimension corresponds to a word from the set of topics and an extra dimension with the value range of [-1,1] corresponds to the sentiment score. We want to find k points surrounding the query point in which they are relatively close to the query point, yet are as far as possible across the k points.

Figure 2 is a visualisation of a search space with 3 dimensions. The x-axis represents the topic word "Covid-19" and the y-axis the word "face mask", while the z-axis represents the sentiment score. The red point is the query and the black points represent sentimented topics. Each sentimented topic is assigned to one or n documents from the collection D. For the case of the figure, each black point indicates a unique document. Tags for the black points close to the query doc point were left out to increase readability.

The position of a point is determined by its sentimented topic by using the word probabilities associated with an arbitrary topic t and sentiment score s. For example, the position of the black point with the label "Doc 39" is (0.8, 0.8, -0.10). This means that the sentimented topic originating from document 39 has a topic that indicates that the content is likely about Covid-19 with 80% and 80% on face masks with a negative sentiment score of -0.10.

The problem that needs to be solved is to find the best combination of k points for the query q. Given a query q, a set of points P and a number k, find the combination of k points that have a moderate distance with the query q, while the combination of k points are as far as possible between them.

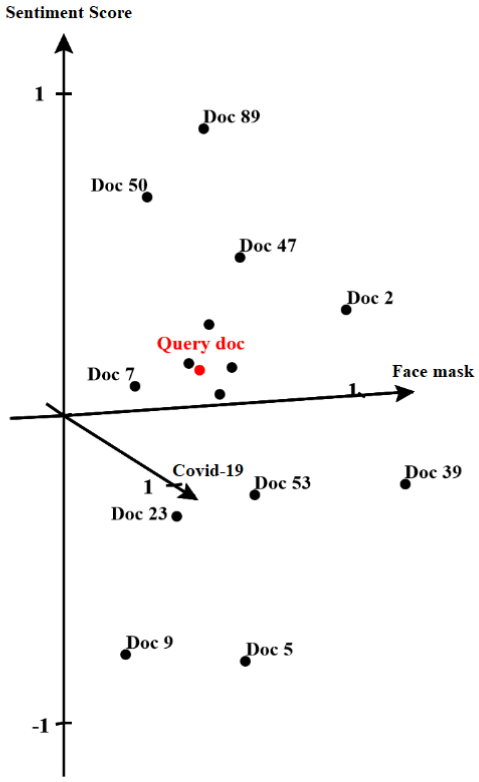


Figure 2: A visualisation of the search space with the query as the red point and sentimented topics as black points.

Example - Problem statement Continuing with figure 2, we assume that we have a document that talks about face masks and COVID-19 as topics with a sentiment of 0.2. This document is represented as the red point with the label query doc. We want to identify a set of documents that creates a neutral informative setting. With $k = 3$, we want to find a set of 3 points that are not too close nor far away from the query point and are as far apart between them. The black points surrounding the query doc are excluded as they are too close to the query doc. This also applies to black points with the labels "Doc 5", "Doc 9", "Doc 39" and "Doc89" as the distance between these points and the query point is too great. The ideal solution is to have a set of points with the labels "Doc 50", "Doc 2" and "Doc 23". These points have a moderate distance from the query and are as far as possible from each other.

4 Proposed Solution

4.1 General workflow

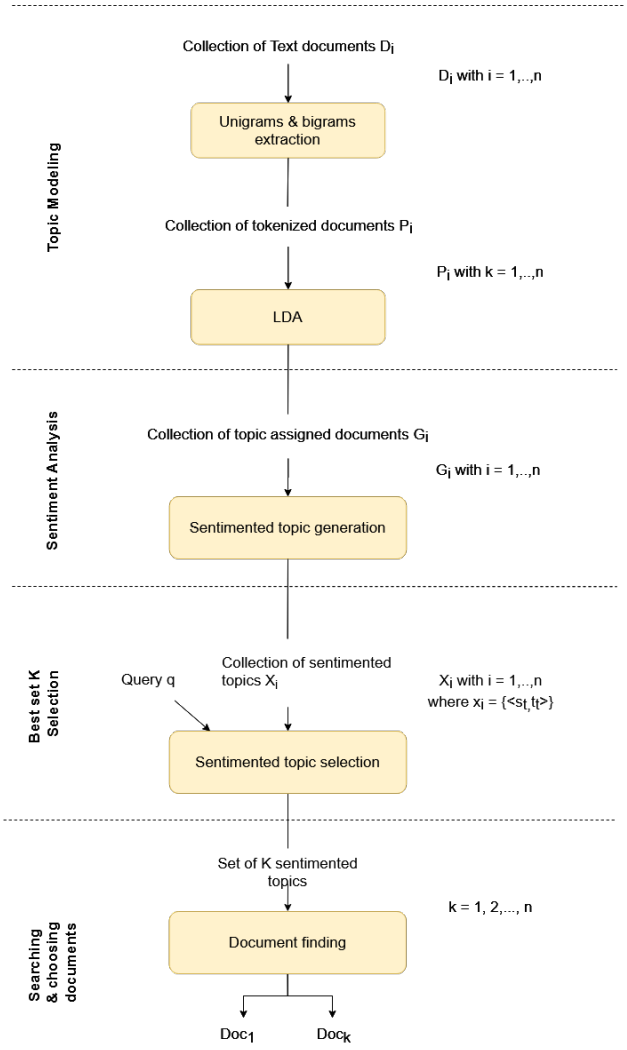


Figure 3: Workflow of the process of finding the best k sentimented topics and associated documents.

This subsection briefly explains the general workflow of the process. The workflow consists of five tasks in which each process can work individually given the designated input. Figure 3 shows the general workflow of the system with the name of the process and the name of the task on the sidelines.

Topic modeling consists of finding the topics based on the collection of text documents and assigning the topic(s) to the document. In the process of unigram & bigram extraction, we take a collection of text documents D_i and clean the documents first by removing punctuation and stopwords. After this step, each document is tokenized into a list of words. For the list of words, lemmatization is applied to convert words to their base form. Subsequently, bigrams are created by combining 2 adjacent words with the condition that the bigram occurs a minimum of 5 times. This results in a collection of tokenized documents that contain unigrams and bigrams of words.

The process LDA creates a topic model with N topics from the collection of unigrams and bigrams. This process creates a dictionary with all unique unigrams and bigrams and a corpus containing documents with

the word ids and their frequency. The dictionary, corpus and an integer n are passed to train a topic model using LDA, which results in a trained topic model that contains n topics. Each topic consists of a fixed number of 30 pairs of a word and a probability. Given the topic model, the model generates a probability score for all n topics based on the unigrams and bigrams of a document. If the probability of a topic T exceeds a certain probability threshold, then that topic is assigned to the document. This results in a collection of topic assigned documents.

The sentimented topic generation creates a set of sentimented topics for the text documents and uses sentiment analysis. The process receives the topic assigned documents as input and splits the text into a list of sentences. For each sentence, the subjectivity score is calculated using a subjective lexicon. This lexicon holds predetermined sentiment and subjectivity values for subjective words. If the subjectivity score exceeds a certain threshold, then the sentence is marked as subjective. Thereafter, the sentiment scores for all extracted subjective sentences within the document are calculated. After this step, the process matches the subjective sentences to one of the corresponding topics that were assigned to the document. This step occurs when a document contains at least 2 topics, otherwise, all subjective sentences are assigned to a single topic. At last, the sentiment scores assigned to a topic are aggregated and averaged. This results in an averaged sentiment score for a topic, in other words, a sentimented topic. Each document thus contains a set of sentimented topics. The output is then passed to the next process.

The process sentimented topic selection is where the selection procedure occurs. It receives the collection of sentimented topics and converts them into n-dimensional points. This method is described in algorithm 2. It also receives the query of the reference document d_r to find the set of k sentimented topics that suits best, which will be explained in the next section. As output, this process delivers a set of k sentimented topics.

The task document finding searches the documents that correspond to the set of k sentimented topics. The process uses the sentimented topics to find the documents that contain one of those sentimented topics. It returns these text documents to the user with optional information on what topic(s) it holds, including topic words and what the sentiment value is of the document.

4.2 Selection procedure

For sentimented topic selection, the process chooses the best combination of k sentimented topics based on the query. The sentimented topics that are not relevant, meaning these sentimented topics that have a great distance or very small distance to the query, should not be considered as candidates for selection. After finding the potential candidate points for the query point, the best combination of k points needs to be found. The best combination of k points needs to be as far as possible between them in distance. After we find these points, we can then select the text documents with the related sentimented topics that correspond to those points.

To intuitively grasp the idea, we illustrate the process in figures 4 and 5. Note that the figures are shown in 2D. In reality, this space is n-dimensional and the circles are delimited areas within n-dimensional space. We have a set of sentimented topics that are points in the n-dimensional space with the query point as our central point. Figure 4 shows 2 circles that divide the search space in 3 regions. The blue circle represents the minimum distance boundary. Intuitively, points that reside in the region within the blue circle are too close to the query point. The red circle represents the maximum distance boundary. The points that reside in the region between the red and blue circles are considered to be candidates for the selection procedure.

Depending on the integer k, the search algorithm needs to find the best combination of points from the set of candidate points based on distance values. The left figure in 5 shows an example solution for $k = 3$. The encircled points chosen by the search algorithm are considered the best combination of points. As can be seen in the right figure, the chosen points in the solution for $k = 4$ are completely different compared to the solution for $k = 3$. This has to do with the constraint that the points should be as far as possible between them. Thus, the best combination of points alters for different values of k. After the best combination is found, we fetch the documents that are linked to those points.

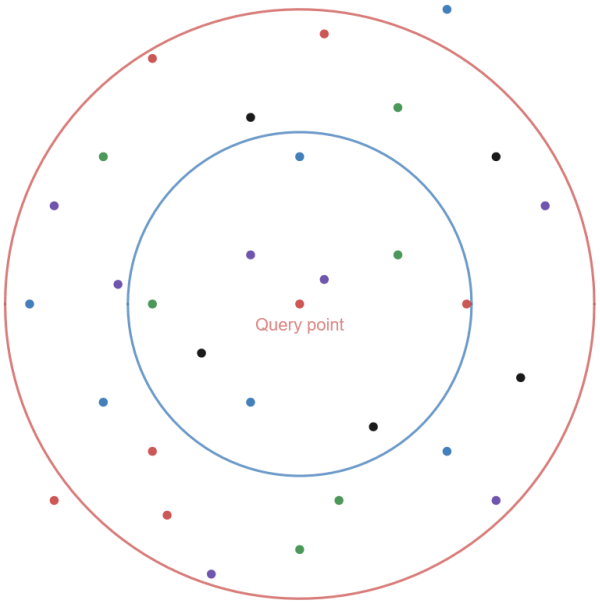


Figure 4: A visualisation of finding candidates for query point.

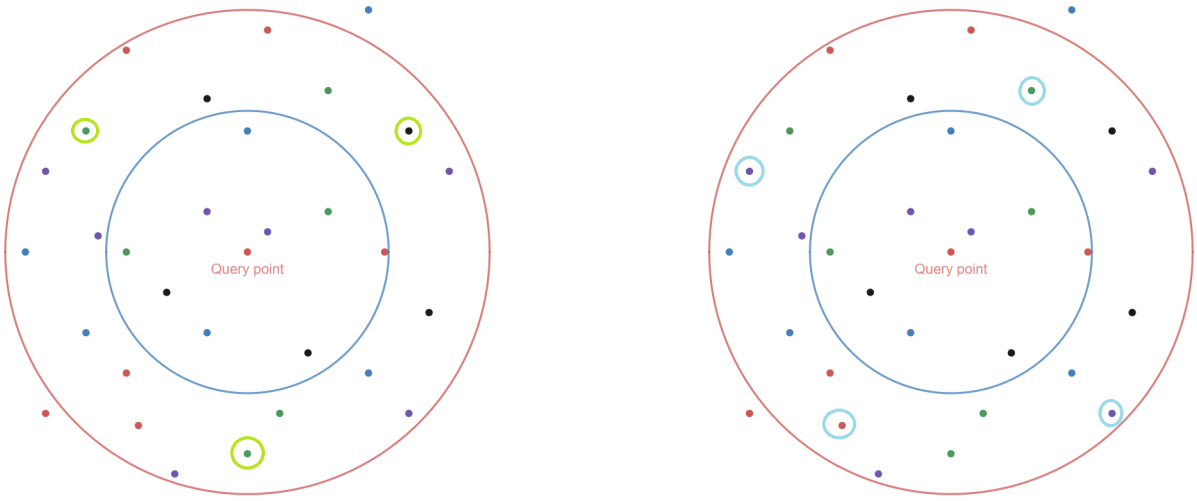


Figure 5: Left: Example solution for k=3. Right: Example solution for k=4

The process starts by extracting the unique sentimented topics and converting them into n-dimensional points. As can be seen in figure 3, the selection component receives a collection of sentimented topics D_i and a query which is a sentimented topic of the reference document d_i. The selection procedure first extracts unique sentimented topics and converts those sentimented topics into n-dimensional points.

Algorithm 1 Sentimented topic extraction

Input: D, a set of sentimented topics
Output: U, a set of unique sentimented topics

```
1: function ExtractUniqueSentimentedTopics(D)
2:     U ← {}
3:     for all d ∈ D do
4:         if d ∉ U then
5:             U.insert(d)
6:     return U
```

Function 1 ensures to return a list with unique sentimented topics. If set U does not contain a sentimented topic d, then it gets added to set U. This will reduce the collection of sentimented topics in case if two or multiple documents share the same sentimented topic, meaning that they share an identical n-dimensional point. The set U of unique items will be converted into n-dimensional points.

The conversion of the sentimented topics to points is done by using the topic words and the sentiment score. Each topic word in the set W of topic words and the sentiment score are an axis in the n-dimensional space. The size of the n-dimensional space is $dim(V) = |W| + 1$.

Algorithm 2 Sentimented topics to point conversion

Input: U, a set of sentimented topics
Output: P, a set of points

```
1: function ConvertSentTop(U)
2:     W ← {}
3:     P ← {}
4:     K ← emptylist
5:     X ← emptylist
6:     for all u ∈ U do
7:         T ← get the topic of u
8:         for all w ∈ T do
9:             if w ∉ W then
10:                W.insert(w)

11:    num ← 0
12:    for all w ∈ W do
13:        K[w] ← num
14:        num ← num + 1

15:    for all u ∈ U do
16:        for all i ∈ [1...|W| + 1] do
17:            X[i] ← 0
18:        for all (w,p) ∈ topic T of u do
19:            index ← K[w]
20:            X[index] ← p
21:        X[|W| + 1] ← sentiment score of u
22:        P[u] ← X
23:    return P
```

The function takes a set U of sentimented topics. A sentimented topic consists of a pair (T, S), a set of topic

words T and a sentiment score S, as stated in Section 4. The words in the topic T are used to determine the dimensions and coordinate placement. The set of unique words from all topics is extracted in which each word receives a unique dimension number. The dimension number will help to assign the probability value to the correct dimension in the coordinates of a point. The total amount of words and the sentiment score defines the number of dimensions where the points are located.

The output of this process is a dictionary K in which each word w represents a dimension. For each sentimented topic u, it creates an empty valued point X of n dimensions based on the total amount of words and sentiment score, hence the size of X is $|W| + 1$.

The method iterates for every word and probability pair (w, p) to fill the coordinates of a point. Using the words and dictionary K, the associated dimension number, denoted as $index$, is fetched of word w. This number will be used to assign the probability value p to $X[index]$. The sentiment score is inserted in the last dimension of the point. Afterward, the point X is added to the set P of points. The function terminates when all sentimented topics are converted into n-dimensional points.

Before the selection procedure is conducted, the number of n-dimensional points will be filtered based on the distance boundaries between the query point and every other point. This method tries to filter the number of n-dimensional points before the selection function performs its task. Reducing the number of points will help to reduce the search time for finding the best combination of points. Algorithm 3 shows the pseudocode for the procedure.

Algorithm 3 Candidate selection

Input: Query point q
P, a set of points
Integer k
MinBoundary minB
MaxBoundary maxB
Output: C, a set of candidate points

```
7:  function GetCandidatePoints(q, P, k, minB, maxB)
8:      Y ← emptylist
9:      C ← {}

10:     for all p ∈ P do
11:         R ← √((p[1] − q[1])² + (p[2] − q[2])² + ... + (p[N] − q[N])²)
12:         Y[p] ← R

13:     if minB == empty and maxB == empty then
14:         avgR ← μ(Y)
15:         stdR ← σ(Y)

16:         while size(C) ≤ k do
17:             minBound ← avgR − 2 ∗ stdR
18:             maxBound ← avgR − stdR

19:             for all p ∈ P do
20:                 if Y[p] > minBound and Y[p] < maxBound then
21:                     C.insert(p)
22:             maxBound = maxBound + stdR
23:     else
24:         minBound ← minB
25:         maxBound ← maxB
26:         for all p ∈ P do
27:             if Y[p] > minBound and Y[p] < maxBound then
28:                 C.insert(p)
29:     return C
```

To reduce the number of points, a function is created to get the set of points that are relevant to the query point. Reducing the number of points also results in reducing the search time to find the best combination of k points. The candidate selection function filters the points based on the distance from the query point.

The function takes a query point q, a set of points P, an integer k and optional boundary values $minB$ and $maxB$. The distance between query point q and every point $p \in P$ is calculated by using the Euclidean distance for n-dimensions denoted as R. For each point p, the distance value R is saved in the dictionary Y.

The minimal distance boundary and maximum distance boundary determine which points are considered candidates. The $minB$ and $maxB$ arguments can be given to the function to find the candidate points. The function iterates through every point p to check whether the distance $Y[p]$ is greater than the minimum boundary, but less than the maximum boundary. If point p passes this condition, it will be added to set C of candidate points to the query point q.

However, it is also possible to leave the arguments minB and maxB empty. For this case, the procedure determines the minimum and maximum boundary values based on the mean and standard deviation distance value between query point q and every point in set P. This is displayed as $avgR$ and $stdR$. Initially, the

boundary values will be set between $avgR - 2 * stdR$ and $avgR - stdR$. If the number of points in the set P is less than the number k, the maximum boundary value will be increased with the standard deviation value until there are at least k points in set C. After this procedure, we end up with a set C of candidate points that can be passed to the selection algorithm.

After the establishment of the candidate points for the query, the selection needs to find the best combination of k points which are as far as possible between them. Given a set of n candidate points, the selection algorithm needs to find the combination of k points in which the points are farthest apart from each other. This means that the algorithm needs to find k points in such a way that the points are approximately evenly far apart from each other.

To determine whether one combination of k points is better than another combination, the notion of force is introduced. Take two points in the N dimensional space, for example. The two points in the N dimensional space are posing a force between them, in which that force depends on their proximity. The larger their distance, the smaller the force posed to each other is. The force posed by a point o to a point p is denoted as F_p^o. A force is a vector itself.

Let two points $o = (x_1^o, x_2^o, ..., x_N^o)$ and $p = (x_1^p, x_2^p, ..., x_N^p)$ in the N-dimensional space. The interpoint vector from o to p, denoted as $IPV^{o \to p}$, is the vector $(x_1^p - x_1^o, x_2^p - x_2^o, ..., x_N^p - x_N^o)$. Intuitively, the interpoint vector is the vector representation of the point p, considering point o as a reference point. The length of the interpoint vector $IPV^{o \to p}$, denoted as $|IPV^{o \to p}|$, is $|IPV^{o \to p}| = \sqrt{(x_1^p - x_1^o)^2 + (x_2^p - x_2^o)^2 + ... + (x_N^p - x_N^o)^2}$.

We would like to have a vector that is parallel to the $IPV^{o \to p}$, but scaled down to have a length of $\frac{1}{R^2}$ to get the force, where R is the distance between the points o and p. In other words, $R = |IPV^{o \to p}|$. To achieve this vector, the interpoint vector is divided by R to get a normalized interpoint vector, subsequently divided by R^2. Therefore, it is enough to divide every dimension of $IPV^{o \to p}$ by a factor of $|IPV^{o \to p}|^3$. The resulted vector is referred to as the *force vector* from o to p.

Definition - Force vector The *force vector* from a point $o = (x_1^o, x_2^o, ..., x_N^o)$ to a point $p = (x_1^p, x_2^p, ..., x_N^p)$, denoted as F_p^o, is the vector

$$F_p^o = (\frac{x_1^p - x_1^o}{R^3}, \frac{x_2^p - x_2^o}{R^3}, ..., \frac{x_N^p - x_N^o}{R^3})$$

where

$$R = |IPV^{o \to p}| = \sqrt{(x_1^p - x_1^o)^2 + (x_2^p - x_2^o)^2 + ... + (x_N^p - x_N^o)^2}$$

The force vectors help to determine the overall cumulated force of a set of k points. It is stated that we want to find the combination of k points that are as far as possible between them. When the k points are as far as possible between them, then the cumulated force value is low. Therefore, the selection module needs to find the set of k points in which the cumulated force from these k points is minimal.

Algorithm 4 Interpoint force vectors calculation

Input: C, a set of N-dimensional candidate points
Output: IPF, a collection of interpoint force vectors

function CalculateIPFs(C)
  initialise IPF as empty collection
  for all $m \in [1..|C|]$ **do**
    for all $n \in [1..|C|]$ **do**
      if $m = n$ **then**
        continue
      $o \leftarrow C[m]$
      $p \leftarrow C[n]$
      $R \leftarrow \sqrt{(p[1]-o[1])^2 + (p[2]-o[2])^2 + ... + (p[N]-o[N])^2}$
      $FV \leftarrow (\frac{p[1]-o[1]}{R^3}, \frac{p[2]-o[2]}{R^3}, ..., \frac{p[N]-o[N]}{R^3})$
      $IPF[m][n] \leftarrow FV$
  return IPF

The selection algorithm consists of 3 functions in order to find the best combination of k points. The function CalculateIPFs in algorithm 4 creates a dictionary IPFs that contains the force vector for every possible combination of 2 points. The method iterates over every point in the set C by using m and n as indexes for C. If index m and index n are identical, then the IPF calculation will be skipped. The force vector of a point p to itself is always 0, therefore it is negligible. The points at indices m and n are locally stored as point p and o. The distance R is calculated by using the Euclidean distance in n-dimensional space.

After the distance is calculated, the force vector FV is obtained by subtracting the point p from o in every dimension, then dividing each dimension by R^3. Thereafter, the vector is saved in a nested dictionary IPF using the indices m and n. In the end, the nested dictionary IPF will be returned by the function.

Algorithm 5 Cumulated force value calculation

Input: KP, a set of N-dimensional candidate points
  IPF, a collection of interpoint force vectors
Output: $cumulF$, a cumulated force value

function CalculateCumulForce(KP, InterpointForces IPF)
  $cumulF \leftarrow 0$
  for all $p \in KP$ **do**
    for all $i \in [1..N]$ **do**
      $D[i] \leftarrow 0$
    for all $o \in KP$ **do**
      if $p = o$ **then**
        continue
      for all $i \in [1..N]$ **do**
        $D[i] \leftarrow D[i] + IPFS[o][p][i]$
    $pCumulF \leftarrow 0$
    for all $i \in [1..N]$ **do**
      $pCumulF \leftarrow pCumulF + D[i]^2$
    $pCumulF \leftarrow \sqrt{pCumulF}$
    $cumulF \leftarrow cumulF + pCumulF$
  return $cumulF$

The function CalculateCumulForce in algorithm 5 receives the collection of interpoint forces IPF and a set KP of points to calculate the force value *cumulF*. The method initializes a variable *cumulF* will contain the total cumulated force value for KP. The function iterates for every point p in KP and initializes a vector D of size N, identical to the size of point p. Next, for every point p, it iterates every point o in set KP. If the points p and o are identical, then o will be skipped to the next element in KP. Vector D receives the force vector of point p and o according to the nested dictionary IPF. Each dimension i in D is filled in according to the value in dimension i of the force vector.

When this step is completed, a variable *pCumulF* is created to temporarily hold the cumulated force between two points p and o. The temporary cumulated force *pCumulF* is acquired by taking the root sum squared value for every dimension i in D. This value is added to *cumulF*. The process repeats until it passes every point in the set KP. The cumulated force *cumulF* is then returned by the function, which tells what the overall force value is for the set KP of points.

Algorithm 6 Finding the best combination of k points

Input: C, a set of N-dimensional candidate points
Integer k
IPF, a collection of interpoint force vectors
Output: *selectedKP*, a set of k candidate points

```
1: function CumulForceSelection(C, k, IPF)
2:     selectedKP ← empty;
3:     globkF = maxINteger;
4:     kP ← generate all possible combinations of k points from C

5:     for all kp ∈ KP do
6:         kF = CalculateCumulForce(KP, IPF);
7:         if kF <globkF then
8:             selectedKP = kp
9:             globkF = kF
10:    return selectedKP
```

Choosing the best combination happens in the function CumulForceSelection in algorithm 6. The function receives a set C of candidate points, the dictionary of interpoint forces IPF and an integer k. First, the function generates every possible combination of k points and initializes the variable *globkF* at the maximum integer value. If the force value of a combination *kp* is smaller than the *globkF* value, then we have found a better combination of k points. This step is repeated until the function traversed every combination in KP. The method uses brute force to solve this problem.

One of the reasons for using brute force is that the algorithm must be generally applicable in n-dimensions. It should be able to find the best combination of k points regardless of the number of dimensions. Thus, it is likely that multiple different combinations of k points result in a similar cumulative force value. Certain subsets of points may yield a better cumulative force value than other combinations of points based on their coordinates. Thus, the selection algorithm has to check every possible combination to see if the best combination of k points does yield the best cumulative force value.

Algorithm 7 Finding the documents associated with chosen points

Input: KP, a set of points KP
Docs, a set of Documents
Output: R, a set of chosen documents

```
30: procedure FindDocuments(KP, Docs)
31:     R ← {}
32:     for all p ∈ KP do
33:         for all doc ∈ Docs do
34:             if p ∈ doc and doc ∉ R then
35:                 R.insert(doc)
36:     return R
```

After the best combination of k has been found, the last step of this process will be to find the text documents that are associated with the k points. The procedure FindDocuments uses the result of the selection algorithm to find the text documents. Each document in *Docs* is marked with at least one and a max of n points. The selection algorithm passes the result to the function FindDocuments. The function examines for every point p in the best combination KP if a document *doc* contains that point p. Document *doc* will be added to the set R of chosen documents if the document contains the point p and if set R hasn't contained document *doc* yet. The set R of result documents is then returned to the user.

5 Implementation

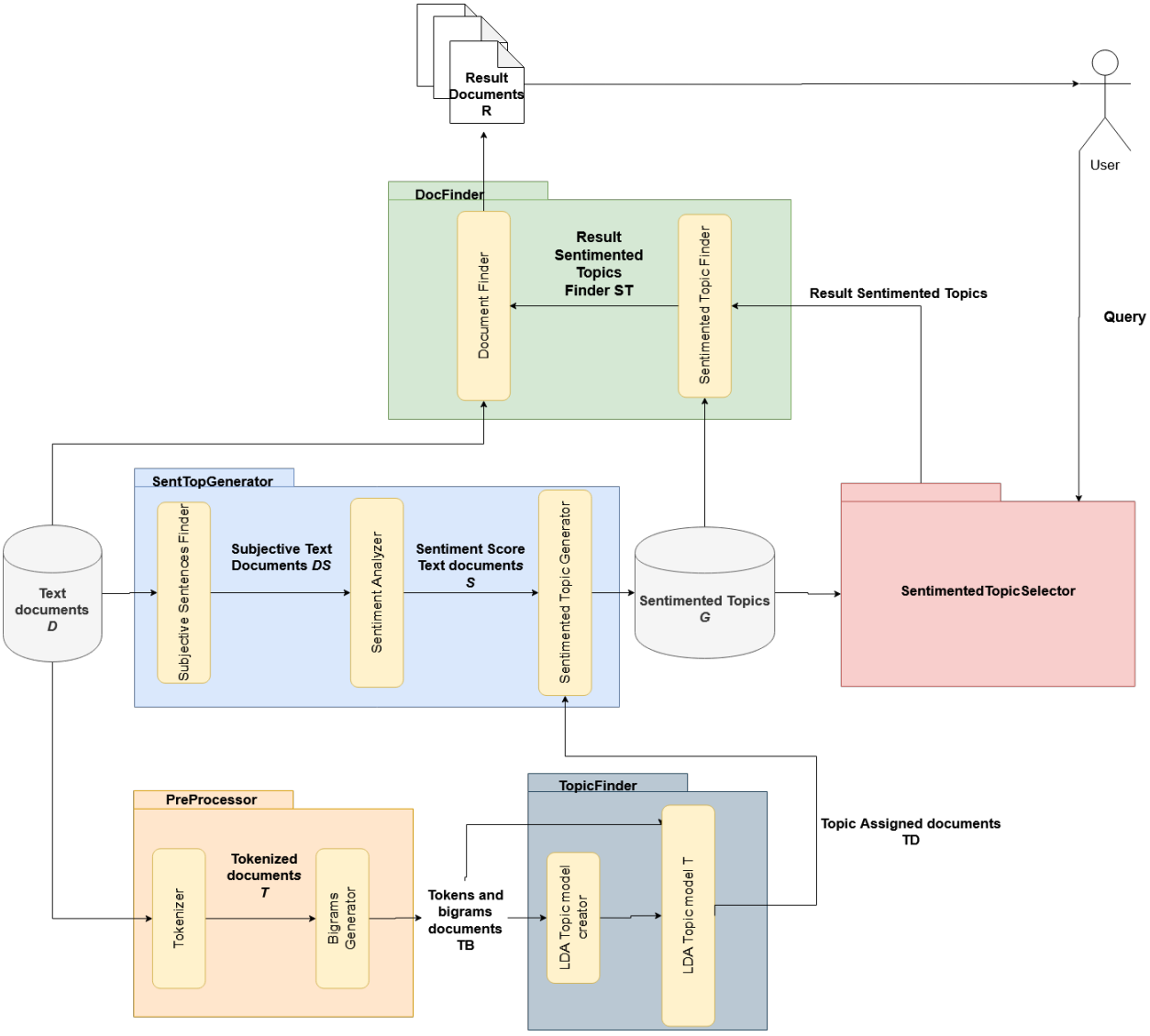


Figure 6: Specified architecture with internal details of each component and input and output. The text document database is the output of through data crawling.

Figure 6 shows the architecture of the system regarding the modules, input, and output. The modules are interconnected to send the required input to them. The text documents D are retrieved using Google and beautifulsoup4 library.

The system is implemented in Python v3.7.7 with the use of several libraries. The collection of text documents D is used as input for 3 modules; PreProcessor, SentTopGenerator, and DocFinder. In the PreProcessor, the text documents are converted by a tokenizer using the functions of the NLTK v3.5 and Spacy v2.3 libraries. This tokenizer performs the tasks that are described in the previous paragraphs. The tokenized documents are then passed to a bigrams generator from the library Gensim v3.8.2 to create a list of unigrams and bigrams of words for each document. The Gensim Phrases model was used to achieve this. Bigrams are considered into the list of unigrams and bigrams if they appear at least a total of 5 times in all documents and if the bigram score is greater than 10. These were the default values of Gensim. The bigram score is

calculated by using the following formula.

$$score(word_A, word_B) = \frac{(bigramCount - minCount) * |vocab|}{(word_ACount * word_BCount)}$$

The tokens and bigrams are then passed to the TopicFinder module, where the topic models are created.

The TopicFinder model creator receives the unigrams and bigrams to create a topic model of N topics. Using the tokenized documents of unigrams and bigrams, we use the LDA technique from the Gensim v3.8.2. library. It generates a topic model that contains N topics where each topic consists of 30 pairs of a word and a probability. The topic model uses the unigram and bigrams documents to derive the topic(s) based on the tokens and bigrams a document contains. The model assigns a probability for each topic inside the model of how likely the document contains that topic. If a topic exceeds a probability of 50%, then the topic is assigned to that document. However, if none of the topics exceed the threshold, then the model will pick the topic with the highest probability. This results in a collection of documents with assigned topics.

In the SentTopGenerator module, the subjective sentences of the documents are extracted by a subjective sentence finder using the functions of the library Textblob v0.15.3. TextBlob is built upon the NLTK and Pattern library. Another option that was considered for the sentiment analysis was VADER. VADER (Valence Aware Dictionary and sEntiment Reasoner) is also a knowledge-based analysis tool as it makes use of a lexicon with predetermined sentiment values. The reason that TextBlob was chosen over VADER is that VADER was built to analyze sentiment from Social Media texts. Tweets or other social media texts are relatively more expressive due to the usage of emoji's/emoticons, writing style, and word use. The collection of text documents consists of opinion articles or blog articles that are more nuanced in their writing style. Therefore, VADER might be less suited to use for determining sentiment scores based on the type of text supplied.

Sentences that exceed the subjective threshold of 0.5 are considered subjective. After this step, the sentiment score for each subjective sentence is then calculated through the sentiment analyzer. The results of the sentiment analyzer and topic assigned documents from the TopicFinder module are passed to the Sentimented Topic generator. This generator matches the sentence with topics by checking if the sentence contains one of the words in the topic. Otherwise, the sentence will be assigned to all topics. The sentiment score for each topic is summed and divided by the number of sentences assigned to the topic. Finally, the generator creates a collection of documents that contain one or more sentimented topics.

The SentimentedTopicSelector module receives the sentimented topics from the SentTopGenerator to convert them into n-dimensional points and find the best set of k points. The procedure and functions are explained in Section 5. The user sends a query to this module to find the best combination of k sentimented topics to the query.

At last, the DocFinder module uses the result of the SentTopGenerator, SentimentedTopicSelector and the collection of text documents to find the documents that are related to create an informative setting based on the reference document, in other words, the query from the user. The component sentimented topic finder searches for the document names that correspond to the sentimented topics. After the names of the documents are found, the result is passed to the Document finder to find the documents in the text document collection D to show it to the user.

6 Evaluation

To evaluate the proposed system and the selection algorithm, we evaluate the data, the performance of the system, and the quality of the results given a reference document. We use *document*0 as our reference document and its sentimented topic as our query.

6.1 Data

For the evaluation, a corpus of 7752 documents originating from unique URL links was fetched. COVID-19 related web pages in English that were published in the last 24 hours were fetched between the dates of 27Th of March and the end of May through Google search. 200 URL links were collected each day. The corpus has been collected through the Google search system. Note that the collection of documents contained duplicates of text documents, however, these documents were extracted from different URL links. Only 7387 documents were unique.

Documents that were equal or smaller than 1kB were removed and documents that contained the words "live blog", "timeline", "live updates" and "updates" in the title were removed from this process as well. Relatively small-sized documents contained 1 or 2 sentences, which are relatively negligible as it conveys too little information. Live-blogs and other related web pages are usually objective in terms of content and writing style. Since the system relies on subjective text pieces, live blogs or timeline articles are less desired. Therefore, these types of articles were removed. Furthermore, the content of large-sized documents of 100kb or larger were usually paired with live blog or timeline articles, hence large-sized documents were also removed. At last, non-English documents were sometimes extracted during the process of collecting. Since we have stated that we use English text only, non-English articles have been discarded as well.

Statistics	**Total number of words**	**Total number of sentences**	**Total number of unique words**	**Average number of words per sentence**	**Long sentence by words**	**Short sentence by words**
mean	1210	55	493	25	90	8
std	1546	70	394	21	154	20
min	81	1	53	7	12	1
25%	501	22	273	19	51	3
50%	816	37	401	22	66	5
75%	1226	57	555	26	91	8
max	16352	817	3614	512	3671	540

Table 1: Statistics of the document dataset of 7387 unique documents. The statistics shows the word and sentence information per document.

Table 1 shows the statistics of the text collection. As can be seen, a number of documents vary in word length. The shortest document contains 1 sentence and 45 unique words, while the longest document contains 817 sentences and 3613 unique words. According to the table, the median value for the total number of words and the number of sentences is 816 words and 37 sentences. Since the mean values for the number of words and sentences are higher, this means that the collection of text documents contains more larger sized text documents than small-sized text documents. The largest document contains 16352 words and 817 sentences, which is more than ten-fold of the mean values. It seems likely that some text documents are a type of live blog regarding the pandemic that have slipped through the removal process.

URL name	Count	Percentage
www.nytimes.com	496	6.71
www.theguardian.com	412	5.58
www.bbc.com	330	4.47
www.nbcnews.com	155	2.10
www.aljazeera.com	142	1.92
thehill.com	108	1.46
www.foxnews.com	102	1.38
www.cnbc.com	101	1.37
www.washingtonpost.com	80	1.08
www.marketwatch.com	75	1.02
www.nationalreview.com	63	0.85
www.businessinsider.com	61	0.83
www.inquirer.com	56	0.76
www.abc.net.au	56	0.76
www.dw.com	54	0.73
foreignpolicy.com	51	0.69
other	5045	68.30

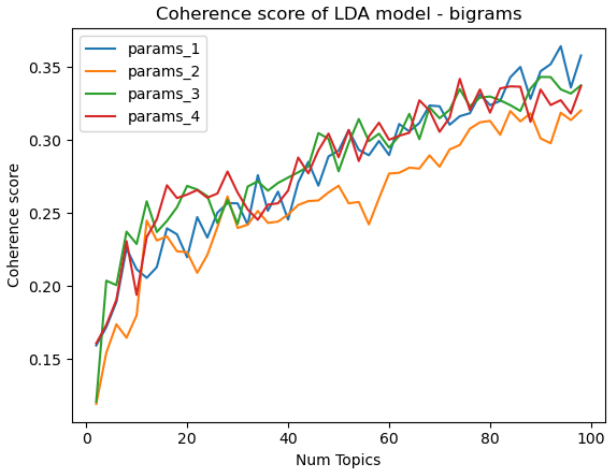
Table 2: The count of URL occurrences based of the collection of documents.

The documents in the repository were fetched from 1633 unique websites. Figure 2 shows the distribution where the documents originated from. If the occurrence of an arbitrary URL link occurs 50 times or less, then this occurrence value is added to the category "Other web URLs." As can be seen, a fair percentage of URL links originates from US-based media outlets.

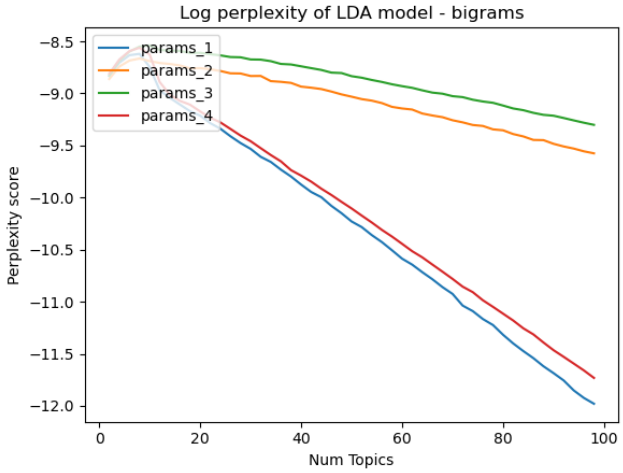
For the creation of the topic model, to choose the best parameters to train a topic model, the metrics perplexity and coherence score C_V measure were used to determine the parameters. We create a multitude of topic models with the number of topics starting from k=2 to k=100 with different parameter settings. A collection of 6500 documents were used to run the different parameter settings.

The following parameters were chosen:

- Parameter setting 1: epochs = 1, iterations = 50, batch size = 1000
- Parameter setting 2: epochs = 1, iterations = 50, batch size = 2000
- Parameter setting 3: epochs = 3, iterations = 100, batch size = 2000
- Parameter setting 4: epochs = 3, iterations = 200, batch size = 1000



(a) The graph shows the C_V coherence scores of 4 topic models over the course of 2 to 100 topics. Each type of topic model has a different parameters setting.



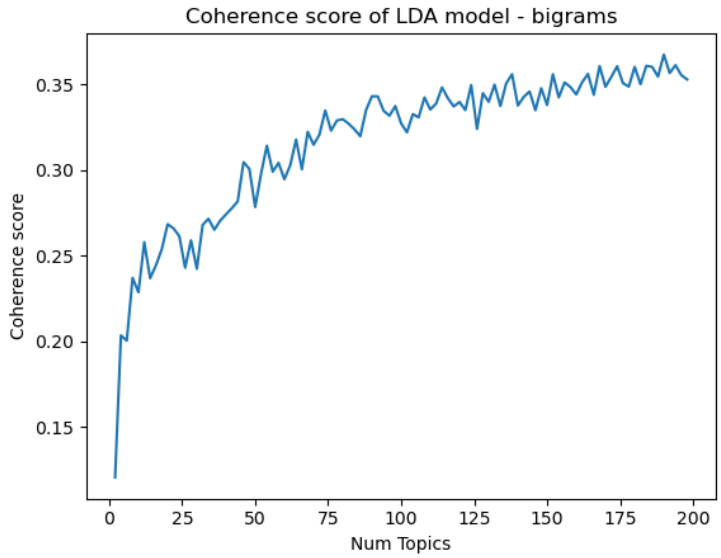
(b) The graph shows the log perplexity scores for 4 topic models over the course of 2 to 100 topics. The perplexity score indicates how well a model performs.

Figure 7: Evaluation measures graph for different settings for topic models by LDA

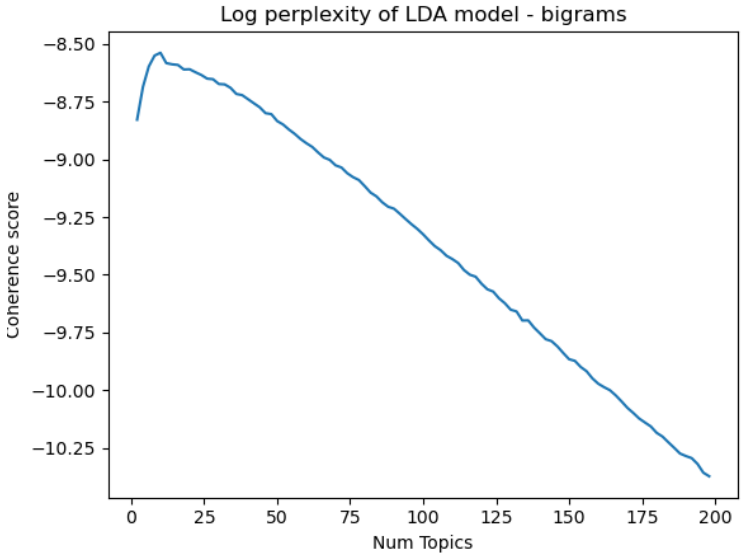
In figure 7, 2 graphs are shown with the evaluation metrics of the different LDA topic models. Figure 7a consists of the coherence scores for the topic models with different parameter settings between 2 to 100 topics. The coherence scores indicate the quality of the topic groups. A higher coherence score means that the words in a topic group are expected to be semantically coherent. The coherence scores for parameter settings 1 and 3 seem to be the highest compared to the other settings.

Figure 7b shows the log perplexity score for the topic models between 2 to 100 topics. The perplexity score indicates how well a topic model predicts for a sample. The model with the lowest perplexity is considered the best. However, as is the case with log perplexity, a high log perplexity indicates better performance for generally predicting a sample. The log perplexity score for parameter setting 1 and 4 descends faster when the number of topics increases. For parameter setting 1, this is likely due to the batch size when compared to the parameter settings 2. It seems that parameter settings 4 also suffers from the same cause. Based on

these results, we use parameter setting 3 for the system to train a topic model.



(a) The graph shows the C_V coherence scores of the topic model with parameter setting 3 over the course of 2 to 200 topics.



(b) The graph shows the perplexity scores of the topic model with parameter setting 3 over the course of 2 to 200 topics.

Figure 8: Evaluation graphs with different metrics of topic models with parameter settings 3.

Figure 8 shows the progress for topics 2 to 200 with this setting. As can be seen, the coherence score measure seems to fluctuate between 0.325 and 0.35 from topic 100 and onward. This shows that the quality of the topics does not improve when the number of topics increases after 100 topics. The perplexity score from topic 2 to 200 seems to descend further when the number of topics increases. As can be seen, the curve reaches its maximum between the interval of 20 and 25. However, the line graph shows that it descends further as the number of topics increases. During the process, 3 topic models were created to determine the

topics for each document from the dataset, resulting in sets of 20, 50, and 100 topics per document.

Topic number	Number topic words	Total number sent-tops	Positive sent-tops	Negative sent-tops	Neutral sent-tops
20	368	3819	2959	841	19
50	777	4653	3668	947	38
100	1560	5097	4045	985	67

Table 3: Table shows the number of unique sentimented topics and the number of positive, negative and neutral sentimented topics according to the topic number.

Table 3 shows the distribution of positive, neutral, and negative sentimented topics for the total number of topics. According to the table, a large number of sentimented topics are positive. A positive sentimented topic is defined such that the sentiment score is greater than 0. Due to the skewed ratio, this leads that the results of the selection procedure will likely return positive documents. Furthermore, the number of unique sentimented topics, therefore unique points, increases when the number of topics increases.

The values in the column "Number topic words" show how many unique topic words exist in the topic number. Each topic contains 30 pairs of a word and a probability. The number indicates the number of dimensions for the n-dimensional points as well.

- 20 topics: 369-dimensional points
- 50 topics: 778-dimensional points
- 100 topics: 1561-dimensional points

Sentimented topics from the 20 topics data result in 369-dimensional points; 50 topics results in 778-dimensional points and 100 topics result in 1561-dimensional points. This shows that if the number of topics increases, the number of dimensions of the points also increases.

For the evaluation of the selection procedure and how well the results match with the query of the reference document, the document *processed*0 is used as the reference document for the search process. The content of this text document can be found in appendix A. The sentimented topic used as the query was determined by the topic models to be $(6, 0.126)$.

Topics 20-model	Topics 50-model	Topics 100-model
('business', 0.015)	('business', 0.021)	('business', 0.028)
('president', 0.007)	('small', 0.009)	('small', 0.012)
('federal', 0.006)	('president', 0.009)	('president', 0.011)
('worker', 0.005)	('federal', 0.008)	('small_business', 0.01)
('small', 0.005)	('american', 0.008)	('chamber', 0.01)
('trump', 0.005)	('program', 0.008)	('program', 0.009)
('law', 0.005)	('small_business', 0.008)	('american', 0.008)
('company', 0.005)	('chamber', 0.007)	('company', 0.008)
('employee', 0.005)	('act', 0.007)	('act', 0.007)
('american', 0.005)	('congress', 0.006)	('congress', 0.007)

Table 4: Table shows the words and probability for topic 6 for each topic model. Note that the table shows the top 10 pairs, while the topics contain actually 30 pairs.

In table 4, each topic originating from the different models has the words "business", "small", "president" and "american". Documents that receive this topic are likely to contain information on small businesses in the U.S during the pandemic and perhaps the influence of the president on small business companies. The probabilities in the 50 and 100 topic models are higher compared to the 20 topics model, meaning that these words are more important within that clustering of words. It should be noted that the probabilities of the words are relatively small since the highest probability value does not exceed 0.021, which may affect the process in later steps.

Briefly explained, the content of the reference document is about relief financial support to small business companies hit by the COVID-19 pandemic. Words related to "small business" as well as "american" are present in the text document.

6.2 Performance of the system

The performance of each module was evaluated on the time it took to perform its task. We check the influence of the size and the type of input to see if it affects the modules. The performance of the modules was measured in Spyder 1.4.1 on an HP pavilion with an 8th Gen. Intel Core i5-8250U processor, 8GB of RAM, and running Windows 10.

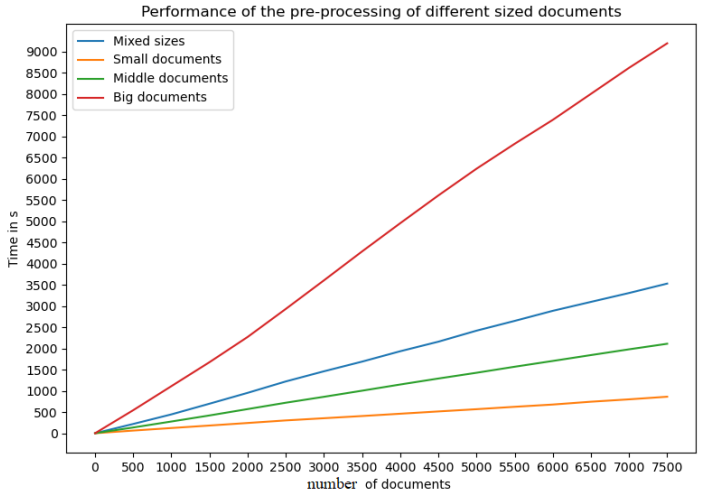


Figure 9: A graph showing the time the pre-processing module takes based on the number and type of input. Note that the process takes approximately linear time.

Size of documents	Time of pre-processing in s	Total time in s
mixed	3529.06	3561.1
small	799.49	861.32
middle	1979.18	2110.19
big	8617.40	9194.87

Table 5: Table shows the exact time in seconds of the performance of the pre-processing module for a total for 7500 text documents.

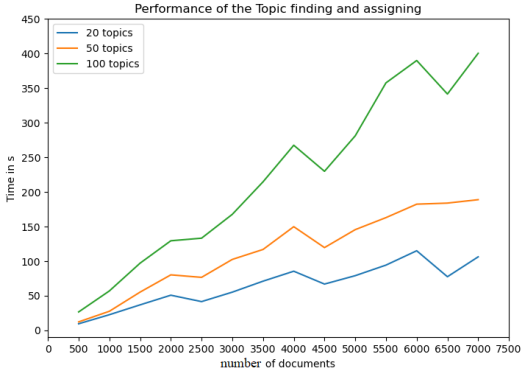
The PreProcessor was tested with 4 different types of text data to see its performance:

- Mixed sized documents: The collection of text documents that were used by the system for this research. This set consists of documents of varying word lengths.
- Small-sized documents: All documents that contain 500 words or less.
- Middle-sized documents: All documents that contain between 500 and 1500 words.
- Big sized documents: All documents that contain at least 1500 words.

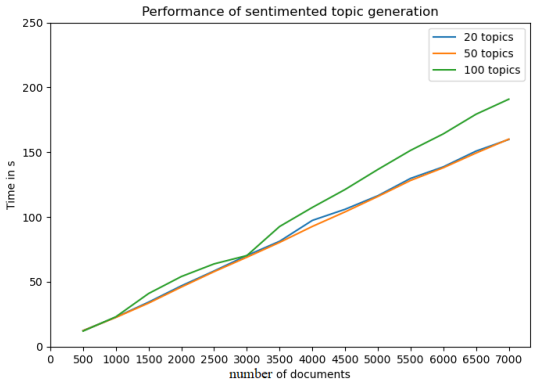
Figure 9 shows the plot of the time it takes in seconds to preprocess the documents compared to the number of documents to be preprocessed. The preprocessing in the figure consists of removing the stopwords, single letters, digits and creating bigrams. Table 5 shows the exact time of the procedure for a total of 7500

documents. The second column shows the time it took to clean the total number of 7500 documents, while the third column shows the total time the module took to preprocess the documents and create a set of unigrams and bigrams.

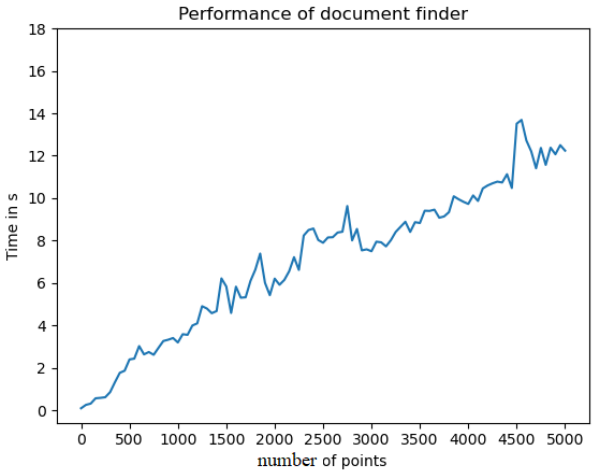
As can be seen in both the figure and the table, large-sized documents take more time to be processed by the module than the other different sized documents. Only processing the large-sized documents took 8617.40 seconds. Small-sized documents take the least amount of time for the process, which is logical as it contains fewer sentences and words to preprocess. Processing the mixed sized documents took a total of 3561.1 seconds. The size of the documents does have an influence on the time of processing. However, the PreProcessor module still takes roughly an hour to complete the task of processing mixed sized documents.



(a) Performance of the TopicFinder module with the number of documents to assign topics and finding the x topics.



(b) Performance of the SentTopGenerator with the number of documents to create sentimented topics.



(c) Performance of the DocFinder module by the number of points to find the associated documents.

Figure 10: The figures show the performance in time by the modules TopicFinder, SentTopGenerator and the DocFinder modules.

Figure 10 shows the time regarding the size of the data for the modules TopicFinder, SentTopGenerator, and DocFinder. The TopicFinder module consists of the task to find the topics of the collection of text documents and assigning the topics to each document. We evaluate the performance of this module by modifying the number of topics and the number of documents to assign topics for it. As can be seen in 10a,

finding 100 topics for a collection of documents took the most time compared to finding 20 or 50 topics. This was expected as the TopicFinder module needs to find more topics of words and probabilities and create a probability distribution for each topic in a single document. Overall, the process took a maximum of over 400 seconds to complete its task.

The SentTopGenerator module takes the text documents and the assigned topics of the documents as input. The module searches the number of subjective sentences that are equal or exceed the subjective threshold of 0.5. This process takes at most 200 seconds. Furthermore, it seems that the number of topics does not influence the time to create sentimented topics.

The DocumentFinder module takes the least amount of time to conduct its task compared to other modules. The number of points influences the time the module needs to perform, when the number of points to find the associated documents increases, the time to execute the task increases.

6.2.1 Selection Module

Since the algorithm uses a naive approach to find the solution, the simulations were run on a separate Linux server with 150GB RAM memory to conduct the experiments and run several simulations.

These parameters were used to evaluate the selection procedure:

1. the number of k
2. the number of candidate points
3. the minimum and maximum distance boundaries

The following results involve sentimented topics where the sentiment scores were not rounded to 3 decimals, meaning the values on the last dimension of the n-dimensional points were unrounded. The purpose was to get the total number of points as large as possible. These results show how the selection procedure performs based on the number of n-dimensional points. Tables 7, 8 and 9 show the number of k, the number of candidate points related to the query point of document *processed0*, the number of combinations of k points and the time to check every combination. The query point is the conversion of the sentimented topic $(6, 0.126)$. The following tables show the number of candidates when no distance boundaries are given.

n-dimensional points	**Unique points**	**Candidate points**
369-dim points	7229	61
778-dim points	6060	26
1561-dim points	7465	52

Table 6: Table shows the statistics for the number of n-dimensional points and candidate points to the query point when the sentiment scores are unrounded.

K	**Number of Candidates**	**Number of combinations**	**Amount of Time in s**
4	61	521855	194.48
5	61	5949147	3170.13
6	61	55525372	44103.74
7	61	436270780	450036.14
8	61	2944827765	X

Table 7: Table shows the statistics for the query point and candidate points based on sentimented topics with a total of 20 topics.

K	Number of Candidates	Number of combinations	Amount of Time in s
4	26	14950	20.22
5	26	65780	54.05
6	26	230230	216.89
7	26	657800	798.47
8	26	1562275	2521.44
9	26	3124550	6435.74
10	26	5311735	X

Table 8: Table shows the statistics for the query point and candidate points based on sentimented topics with a total of 50 topics.

K	Number of Candidates	Number of combinations	Amount of Time in s
4	52	270725	152.93
5	52	2598960	1807.31
6	52	20358520	20473.62
7	52	133784560	175000.35
8	52	752538150	1344448.12
9	52	3679075400	X

Table 9: Table shows the statistics for the query point and candidate points based on sentimented topics with a total of 100 topics.

The points were acquired by converting the sentimented topics that could contain one of the 20 topics. There was a total of 7229 unique points. When no minimum and maximum distance boundaries are given, the procedure finds 61 candidate points for the query point in table 7. The maximum number of k to find the best combination is $k = 7$. As the number of combinations increases, the time to find the best combination increases too. The run for $k = 8$ terminated as the number of combinations was too large for the amount of memory that was available.

For the number of 50 topics, there was a total of 6060 unique points and 26 candidate points found for the query point. The selection module found 26 candidates when no distance boundaries are given. The maximum number of k the selection procedure could handle was $k = 9$ with a total number of 3124550 combinations before it ran out of memory.

For the number of 100 topics, there was a total of 7465 unique points and 52 candidate points found for the query point. The maximum number of k for the dataset with 100 topics is $k = 8$ with a total of 752538150 combinations. The time to find the solution for $k = 8$ for 52 candidate points took roughly over 15 days to find a solution, which is the longest processing time of all modules of the implementation.

Based on the results, this indicates that the selection module can find $k = 9$ points at most when the number of candidates is 52 or less. In addition, the selection takes the most time to conduct its task when it has to find a combination depending on the number of candidates and the number k.

The following results were obtained when the sentiment scores were rounded to 3 decimals. When no boundary values are given, the procedure finds the following number of point candidates in table 10.

n-dimensional points	Unique points	Candidate points
369-dim points	3819	118
778-dim points	4653	48
1561-dim points	5097	82

Table 10: Table shows the statistics for the number of n-dimensional points and candidate points to the query point when the sentiment scores are rounded.

The number of unique points reduced for all the data types when the last dimension is rounded at 3 decimals compared to table 6. This means that multiple documents share an identical sentimented topic, which results that the system returns at least k or more documents. The maximum number of k in which a solution can be found for these number of candidates are $k = 5$, $k = 8$ and $k = 6$ in respective order. The tables show that there is a trade-off between the parameter k and the number of candidates to reach a solution. A low number of candidates makes it possible to search for a relatively high number of k, while a high number of candidates limits the k to a low value.

Boundaries (min & max)	Topics 20 data	Topics 50 data	Topics 100 data
0 - 0.1	1717	2059	2050
0.1 - 0.2	1262	1659	2051
0.2 - 0.3	569	646	690
0.3 - 0.4	166	182	196
0.4 - 0.5	64	65	68
0.5 - 0.6	19	19	20
0.6 - 0.7	11	13	13
0.7 - 0.8	3	3	3
0.8 - 0.9	2	2	2
0.9 - 1	1	1	1
>1	3	3	3

Table 11: Table shows the number of candidates when the minimum and maximum distance boundary values are given to the selection procedure. The number of candidates is shown for the query point.

In table 11 the number of candidate points for the query point given the boundary values is shown. As can be seen, a large number of n-dimensional points are found in the boundary value range between 0 and 0.3 for every topic model. This indicates that a large number of sentimented topics are relatively close to the query. This is caused by the probabilities of the words. The probabilities are low and do not exceed 0.021 shown in table 4. Therefore, it shows that quite many points lie in a distance value range between 0 and 0.3. According to the table, there are 3 n-dimensional points that have a distance value greater than 1 given the query point. Given the boundaries and the number of candidates, the maximum number of k is $k < 4$ if the maximum boundary is $max_{bound} < 0.5$ in order to reach a solution.

Boundaries (min & max)	**Topics 20 data**	**Topics 50 data**	**Topics 100 data**
0 - 0.1	297	738	1351
0.1 - 0.2	574	1171	1777
0.2 - 0.3	856	1336	1226
0.3 - 0.4	969	875	504
0.4 - 0.5	674	374	163
0.5 - 0.6	319	110	51
0.6 - 0.7	85	32	14
0.7 - 0.8	30	9	6
0.8 - 0.9	8	3	2
0.9 - 1	1	1	1
>1	0	0	0

Table 12: Table shows the number of candidates when the minimum and maximum distance boundary values are given to the selection procedure. The number of candidates is shown for a query point that is most common within the set of unique points.

In table 12, neutral sentimented topics with the most common topic in the data of n-dimensional points are used to count the number of candidate points. The following sentimented topics were used as query points, since the topics are the most common within that type of dataset:

- Topics 20 data: (12, 0)
- Topics 50 data: (30, 0)
- Topics 100 data: (30, 0)

The table shows how the candidate points are scattered over different ranges of boundary values. For each type of dataset, we see that the vast majority of the points lie in the distance range between 0 to 0.4. This means that when the user gives the distance boundaries between, for example, 0.1 and 0.4, the number of k should be $k < 4$, as no solution can be found due to the sheer amount of computation. Table 11 and table 12 show that most of the points lie within the maximum boundary value of 0.5.

6.3 Quality of the results

For the evaluation of the quality of the results, the system needs to find documents that are related to the content of the document of the query. We use the sentimented topic of the reference document. The content of this document appears in appendix A. The text consists of an $Q\&A$ interview with Chris Hurn regarding relief financial support to small business companies hit by the COVID-19 pandemic.

K	Topics 20	Topics 50	Topics 100	K	Topics 20	Topics 50	Topics 100
2	(6, 0.159), (6, 0.092)	(6, 0.167) , (6, 0.088)	(6, 0.081), (6, 0.167)	**4**	(6, 0.092), (12, 0.14), (6, 0.159), (12, 0.112)	(6, 0.167), (6, 0.088), (35, 0.122), (6, 0.126)	(6, 0.167), (41, 0.104), (41, 0.149), (6, 0.081)
3	(6, 0.159), (6, 0.093), (6, 0.126)	(6, 0.127), (6, 0.167), (6, 0.088)	(6, 0.081), (6, 0.167), (60, 0.13)	**5**	(6, 0.092), (7, 0.14), (16, 0.125), (7, 0.113), (6, 0.156)	(6, 0.143), (6, 0.115), (35, 0.118), (6, 0.167), (6, 0.088)	(6, 0.167), (41, 0.104), (30, 0.121), (41, 0.149), (6, 0.081)

Table 13: Table shows which sentimented topics are chosen as best combination based on k and the query (6, 0.126).

K	Topics 20 data	Topics 50 data	Topics 100 data
2	2	2	2
3	3	3	3
4	10	4	4
5	12	6	8

Table 14: Table shows how many documents are fetched based on k and the query (6, 0.126).

K	Topics 20	Topics 50	Topics 100	K	Topics 20	Topics 50	Topics 100
2	Doc1053 Doc6003	Doc1470 Doc4468	Doc1390 Doc1470	**4**	Doc6003 Doc1864 Doc6715 Doc1053 Doc1618 Doc2624 Doc6246 Doc7132 Doc7419	Doc1470 Doc4468 Doc6175 Doc0	Doc1470 Doc5056 Doc1410 Doc1390
3	Doc1053 Doc818 Doc0	Doc4701 Doc1470 Doc4468	Doc1390 Doc1470 Doc6205	**5**	-	Doc206 Doc5607 Doc3640 Doc4391 Doc1470 Doc4468	Doc1470 Doc5056 Doc2099 Doc3441 Doc620 Doc7 Doc1410 Doc1390

Table 15: Table shows which documents are fetched up until $k = 5$ for every type of data for the query point of sentimented topic (6, 0.126) with the exception of topics 20 data.

Tables 13, 14 and 15 show which sentimented topics and documents are found for the query. The document names for $k = 5$ in the 20 topics are not noted to keep the readability. In table 13, the best combination of k sentimented topics according to the system for the query (6, 0.126) are shown. Most of the sentimented topics contain the same topic as the query, namely, topic 6. This was expected as a sentimented topic that contains the same topic as the query, which is considered highly relevant to the query. Other topics such as topics 7, 12, and 16 in the 20 topics dataset, topic 35 in the 50 topics dataset, and topics 41 and 30 in the 100 topics dataset were also considered relevant to the query.

In appendices A and B, the content, topic and sentiment score of fetched documents can be found for the documents *Doc1390*, *Doc1470* and *Doc5056*. The topics originate from the topic model with 100 topics. *Doc1390* and *Doc1470* share a common topic with the reference document. *Doc1390*, however, is less positive about the topic than *Doc1470*. *Doc5056* does not share the same topic with other documents, yet it is related according to the reference document as it contains similar words to the topic and talks about economic relief for firms, just in the UK. This shows that the selection procedure correctly chooses different sentimented topics if they are related to the query and contribute to an informative setting.

Another noticeable feature is that all sentiment scores are positive and deviate slightly from the value 0.126.

This can be explained as a large number of sentimented topics are classified as positive as seen in table 3.

According to table 14, it is noteworthy that the resulting documents of $k = 2$ from 50 topics dataset and 100 topics dataset are subset of the resulting documents of $k = 3$ and $k = 4$. This can also be seen in table 13. This did not apply to the resulting documents for the 20 topics dataset. This shows that different values for k result in different solutions of the best combination of sentimented topics.

It should be noted that the resulting documents in $k = 3$ from Topics 20 data and $k = 4$ from the 50 topics dataset contain the query document, which is surprising as it was expected that identical documents to the reference document should not be returned. The cumulated force equation in algorithm 5 ensures that a sentimented topic or n-dimensional point that are equal to the query point yield a high cumulated force, meaning that highly similar sentimented topics are less likely to be returned.

The following tables show the Jaccard distance and TF-IDF similarity values between the result documents for $k = 2$, $k = 3$, and $k = 4$. The purpose is to show that the resulting documents should not be identical to each other and bear slight similarity with the reference document. Our proposed system should return different documents to provide different perspectives, however, these perspectives should be related to the query. The aim of this part of the evaluation is to show that the resulting documents are dissimilar between the output documents and the reference document, thus low similarity values are desired. The result documents are merged in the tables to keep an overview and see the similarity between the documents and the set of solutions.

Jaccard Similarity	*Reference*	**Doc1053**	**Doc6003**	**Doc818**	**Doc0**
Reference	x	0.098	0.109	0.081	1.0
Doc1053	0.098	x	0.085	0.068	0.098
Doc6003	0.109	0.085	x	0.052	0.109
Doc818	0.081	0.068	0.052	x	0.081
Doc0	1.0	0.098	0.109	0.081	x

(a) Table shows the Jaccard similarity of between documents from the results of k=2 and k=3 combined. Doc0 is the query document. The documents may contain one of the 20 topics.

Jaccard Similarity	*Reference*	**Doc4701**	**Doc1470**	**Doc4468**	**Doc6175**	**Doc0**
Reference	x	0.10	0.117	0.096	0.07	1.0
Doc4701	0.10	x	0.145	0.097	0.057	0.1
Doc1470	0.117	0.145	x	0.089	0.073	0.117
Doc4468	0.096	0.097	0.089	x	0.078	0.096
Doc6175	0.07	0.057	0.073	0.078	x	0.07
Doc0	1.0	0.10	0.117	0.096	0.07	x

(b) Table shows the Jaccard similarity of between documents from the results of k=3 and k=4 combined. Doc0 is the query document. The documents may contain one of the 50 topics.

Jaccard Similarity	*Reference*	**Doc1390**	**Doc1470**	**Doc6205**	**Doc5056**	**Doc1410**
Reference	x	0.106	0.117	0.074	0.104	0.085
Doc1390	0.106	x	0.088	0.062	0.094	0.073
Doc1470	0.117	0.088	x	0.083	0.132	0.078
Doc6205	0.074	0.062	0.083	x	0.106	0.060
Doc5056	0.104	0.094	0.132	0.106	x	0.120
Doc1410	0.085	0.073	0.078	0.060	0.120	x

(c) Table shows the Jaccard similarity of between documents from the results of k=3 and k=4 combined. Doc0 is the query document. The documents may contain one of the 100 topics.

Table 16: Three tables show the Jaccard similarity between the result documents given in table 14. The resulting documents were combined in a single table to keep overview of the similarity values.

For the resulting documents, the Jaccard index was used to see the similarity scores between the resulting documents. Note that the result documents are combined into a single table. For the 20-topics data, we combined the result documents of $k = 2$ and $k = 3$ to keep the results surveyable. We use the set of unique words from the documents to apply the Jaccard Index. Punctuation, single letters, stop words, and digits were removed from the documents. The *Reference* represents the reference document, while the bold documents are the result documents.

As can be seen in every table, the Jaccard similarity value between every document with the reference document is at most 0.117 for Topics 50, except for **Doc0** as it is the same text document as the reference document. Most of the values shows a Jaccard similarity of 0.15 or lower, meaning that the output documents are distinct. Most of the resulting documents within the set are also distinct from the *Reference* except for **Doc0**.

TF-IDF	*Reference*	**Doc1053**	**Doc6003**	**Doc818**	**Doc0**
Reference	x	0.076	0.186	0.084	1.0
Doc1053	0.076	x	0.057	0.052	0.076
Doc6003	0.186	0.057	x	0.059	0.186
Doc818	0.084	0.052	0.059	x	0.084
Doc0	1.0	0.076	0.186	0.084	x

(a) Table shows the TF-IDF similarity of between documents from the results of k=2 and k=3 combined. Doc0 is the query document. The documents may contain one of the 20 topics.

TF-IDF	*Reference*	**Doc4701**	**Doc1470**	**Doc4468**	**Doc6175**	**Doc0**
Reference	x	0.11	0.179	0.119	0.055	1.0
Doc4701	0.11	x	0.355	0.094	0.068	0.11
Doc1470	0.179	0.355	x	0.142	0.082	0.179
Doc4468	0.119	0.094	0.142	x	0.18	0.119
Doc6175	0.055	0.068	0.082	0.18	x	0.055
Doc0	1.0	0.11	0.179	0.119	0.055	x

(b) Table shows the TF-IDF similarity of between documents from the results of k=3 and k=4 combined. Doc0 is the query document. The documents may contain one of the 50 topics.

TF-IDF	*Reference*	**Doc1390**	**Doc1470**	**Doc6205**	**Doc5056**	**Doc1410**
Reference	x	0.228	0.165	0.069	0.091	0.169
Doc1390	0.228	x	0.149	0.037	0.089	0.134
Doc1470	0.165	0.149	x	0.057	0.181	0.133
Doc6205	0.069	0.037	0.057	x	0.104	0.07
Doc5056	0.091	0.089	0.181	0.104	x	0.166
Doc1410	0.169	0.134	0.133	0.07	0.166	x

(c) Table shows the TF-IDF similarity of between documents from the results of k=3 and k=4 combined. Doc0 is the query document. The documents may contain one of the 100 topics.

Table 17: Three tables show the TF-IDF similarity between the result documents given in table 14. The resulting documents were combined in a single table to keep overview of the similarity values.

The TF-IDF calculation is done by using the python library scikit-learn v0.23.2. Punctuation, digits, single letters, and stop words were removed from the text. After TF-IDF has been calculated, cosine similarity is used to calculate the similarity value between the documents. Most of the output documents contain low cosine similarity values, i.e. 0.2 or lower. As seen in the table of the 50 topics, **Doc4701** and **Doc1470** has a similarity score of 0.355, which is relatively high compared to the others, yet still low in general.

The value difference for **Doc1470** with *Reference* in table (b) with value 0.179 and (c) with value 0.165 can be explained as TF-IDF depends on the set of documents. The groups of documents, where **Doc1470** is in, are different in (b) and (c), therefore the IDF differs, resulting into slightly different TF-IDF values and also slightly different similarity values.

The tables show that most of the documents have low similarity values from other documents based on the Jaccard and cosine similarity values using TF-IDF. This means that most of the output documents are dissimilar from each other. **Doc6003**, **Doc1470**, and **Doc1390** have a slightly higher similarity score with *Reference*. However, looking at the contents of **Doc1470** and **Doc1390**, which can be found in Appendix A, the topics in both documents are related as it talks about the economic relief package for small companies. The content for the user opinion on the topic is different. This shows that the system does indeed return output documents that seem relevant to the reference document but still are dissimilar in text content.

7 Conclusion

7.1 Discussion

This section discusses the limitations and suggestions to future research based on the development of the research project. During the thesis project, some issues were encountered that has an impact on the dataset or project solution. The system relies on a document dataset that is varied enough in user opinions, perspectives or views for multiple topics. The main task is that it returns documents that provide a neutral informative setting by exposing the user to different viewpoints of other documents. For the dataset acquisition, the Google search system was used to collect text documents. However, the search system also personalizes search results based on search history, web history and location. Weblinks that were not expected to show up in the results still appeared.

Weblinks
https://www.hulldailymail.co.uk/news/uk-world-news/live-updates-coronavirus-uk-cases-4057703
https://www.rijksoverheid.nl/
https://www.demorgen.be/dossier/corona-virus
https://www.afasienet.com/nieuws-professionals/blog-van-sem-theuniszen-het-coronavirus/
https://www.kpbs.org/news/2020/apr/20/coronavirus-san-diego-live-updates-covid-19/

As can be seen, some web links originate from the Netherlands or Belgium. This means that the search results were biased from the beginning during the process of data crawling. Due to the personalized search results, it is highly likely that the dataset also contains bias. There's a possibility that there are more documents that align with personal opinion and view on topics, thus making the dataset of text documents with certain opinions skewed. It might be better to use a search system that does not make use of search history or personal recommendations to find web documents.

Additionally, the proposed system relies heavily on the collection of data that it should contain a fair ratio of positive and negative opinions on certain topics. It is possible that the subject "Covid-19" might be too general as several documents are talking about Covid-19 in combination with the economy, college or health. For future work, the collection of text documents should be focused on multiple subjects such as Covid-19 in economy or Covid-19 in health to see how the system performs with a more focused text dataset.

Another issue that has been found is during the Sentiment analysis part. The python library TextBlob for the sentiment analysis has unfortunately an incomplete subjective lexicon. Some sentences or words were e.g. assigned as relatively positive, while the overall consensus of the sentence is conceived as negative.

Sentences	**Sentiment score**
"Only Italy has reported more deaths."	0.25
"These front-line workers take great risk, yet we've let them down."	0.322
She also reminds residents the state has hate crime laws and an ethnic intimidation law to protect residents.	-0.8
My dad has two very serious illnesses, and there are many more like him.	0.189

Table 18: Examples of sentences classified by TextBlob which should have been assigned the opposite sentiment score.

It seems that there are sentences whose sentiment scores were incorrectly calculated during sentiment analysis. This influenced the creation of the sentimented topics in the sense that some sentimented topics should have been negative towards a topic rather than positive. Due to the amount of positive sentimented topics that were listed in table 3, it's likely that some positive sentimented topics should have been considered neutral or negative. For future work, it would be recommended to use a lexicon that is more complete than the library TextBlob.

We have stated that a document may contain 1 to N topics using the technique of LDA. However, most of the documents contain at most 1 topic. This is explained due to the fact that LDA generates a probability distribution over all the topics. For example, if the LDA creates a topic model with 100 topics, the technique can assign 2% for each topic to a document, which does not help to identify the content in terms of the topics

of a text document. Additionally, another problem with the technique LDA is that the word probabilities are rather low. These values are small that it has little to no impact to the selection procedure. Looking at the results in the subsection of quality of the results, the set of resulting documents contains the reference document as well. It was expected that identical sentimented topics should not be included as the cumulated force value would be too great, which was not the case.

Another matter that needs to be noted is that the topics from the technique LDA are not always been coherent. For example, the *document*7691 is briefly about distant teaching during the pandemic and what the struggles are from the perspective of a professor.

Topics 20	**Topics 50**	**Topics 100**
Student	Animal	Page
School	Market	Team
Universiy	Wildlife	College
Travel	Bat	Account
Education	Risk	Event

Table 19: The top 5 words from the topic that is assigned to *document*7691 from topic model with 20, 50 and 100 topics.

The topic from Topics 20 matches the most with the content of *document*7691. From the model with 100 topics, the topic contains the word "college", but other words seem incoherent. The cluster of words from the model with Topics 50 is completely off from the content of the text. Based on the table 19, it's likely that the resulting documents do not match with the reference document as they have been assigned with "incorrect" topics. This could result that irrelevant documents are returned that have nothing to do with the content of the reference document. In future works, an improvement would be to see if there's a way to extract more meaningful topics by tweaking the hyperparameters or use a different method to find and extract topics with probability values.

The selection procedure uses brute force to find the solution to the problem. When the system works with a large quantity of data, the time for the process takes increases. Additionally, the module is limited due to the sheer amount of memory it needs to conduct the task. It shows that $k = 9$ with a maximum amount of 52 candidates is the limit to find a solution. An improvement would be to find a way to cut the number of combinations to find the best k points without losing the optimal solution. This poses a difficult task as the results in 13 shows that the set of sentimented topics differs based on the number of k.

At last, the quality of the results showed that most of the documents are dissimilar to the reference document based on the Jaccard and TF-IDF similarity. This indicates that the result documents that are fetched by the system are distinct from each other, but this could also mean that the content of those documents is irrelevant to the reference document. This can be explained based on the quality of the collection of text documents. Based on the topics from topic modeling, the collection contains text documents that are mostly about the crisis, politics and economy. There could have been fewer articles regarding economic relief for small businesses. Furthermore, the collection contains more large-sized documents with an average of 1210 words. It's likely that a small paragraph from a resulting document corresponds to the context of the reference document. Other approaches to evaluate the quality of documents such as experimental evaluations should be considered to measure the quality.

For future research, it might be better to conduct evaluation experiments with human participants. It would be nice to see if the output documents achieve the effect that they create a neutral informative environment given a reference document. This would give a better indication of whether the system delivers the effect as intended. Another method to test whether the reference document and output documents are relevant to each other is to annotate the collection of text documents. Using the annotations, one can evaluate the result of the system with the ground truth to see if our proposed system works as intended.

7.2 Reflection

Despite its limitations, there is also some merit to our research. Our system can be used to find topics from a collection of, in our case, COVID-19 related web articles and create a neutral information environment based on a topic and sentiment. It could also be utilized for other types of information. For example, it could create an informative setting based on product reviews to get an overall impression of the product. The system is built in such a way that the techniques in modules are interchangeable. It is possible to use methods other than topic modeling or sentiment analysis. Currently, sentiment analysis is conducted using a subjective lexicon, but this could be replaced by, for example, deep learning models. Furthermore, we could find no other research that seems to propose a solution similar to this work, especially concerning the use of opinions to create a neutral information setting. As such, this thesis could bring interest to a new research field.

7.3 Conclusion

Fake news and misinformation are problematic and can cause harmful consequences in society. In this work, a system has been developed to create a neutral and informative setting by searching for a combination of documents to indirectly fight filter bubbles susceptible to misinformation or fake news. Several techniques regarding processing text documents and user opinion are introduced to create sentimented topics. The sentimented topics hold information about the sentiment of the user to a specific topic.

The system searches for the best combination of k sentimented topics to fetch the text documents that create a neutral and informative setting given the sentimented topic of a reference document. The dataset, performance of each module and the quality of the results of the system were evaluated. The results show that the pre-processing and selection modules take the longest time to perform their task compared to other modules based on the amount of data. The system does find different sentimented topics with different sentiment score and topics eventually. Based on the number of k, the system finds different solutions of sentimented topics.

For future work, improvements should be made to the proposed system to reduce the search time as the selection algorithm uses brute force to find the solution. Other suggestions described in the section "Discussion" should be taken into consideration to enhance the overall performance of the system.

Appendices

A Text

A.1 Reference document

How SBA Can Help Your COVID-Hit Company?
As the COVID-19 pandemic unfolds, federal support is becoming available for small businesses and non-profit organizations. The centerpiece of that aid is $367 billion in total financing that the U.S. Small Business Administration will oversee under the $2 trillion economic relief package.
SBA will oversee the legislation's $350 billion Paycheck Protection Program, according to a statement on the website of the Senate Small Business Committee from Sen. Ben Cardin, a Maryland Democrat and the committee's ranking member. Highlights include:
Before Congress took up the economic rescue package, it authorized SBA in early March to oversee a separate loan initiative for small businesses. The agency is issuing low-interest economic injury disaster loans of up to $2 million for each filing pandemic-impacted business, with repayment terms of as much as 30 years. Privately held, mid-sized businesses can also benefit from SBA 7(a) relief support loans of up to $10 million, which can help with payroll and other major costs. READ ALSO: CPE Coronavirus Coverage
Chris Hurn, the CEO of Fountainhead Commercial Capital, which provides SBA 504, SBA 7(a), and low LTV loans to small and mid-sized businesses, advises business owners to act quickly. The number of loan filings has been rising steadily, with his company recording more than 2,500 applications in the past month.
How much do you estimate the national economy will be by the COVID-19 pandemic? Can you compare it to anything the economy has experienced in the past?
Hurn: The COVID-19 pandemic will have a devastating impact on our economy. The Great Recession and 9/11 were more one-off situations, (but were the situations) most comparable to this. With recovery efforts already underway and extraordinary precautions like social distancing and sheltering in place, we're helping to stop the spread of the virus, but this leads to businesses grinding to a halt, which is something we haven't seen before.
To combat this, as of right now, $300 billion is proposed in SBA loans, which would be over six times what SBA has ever done in any previous fiscal year— and all expected to be done in a shortened period of about six months or likely less. (Editor's note: this interview was conducted before lawmakers approved the final version of the federal relief package.)
SBA and its participating lenders have never faced a challenge of this magnitude. This is a significant action in percentage terms and in scale, and it is properly directed at small to mid-sized businesses that suffer the most at times like these, unlike during the Great Recession when most of the recovery efforts were directed at large corporations.
Which sectors are more exposed, and why?
Hurn: The sectors that were more exposed at first were travel and hospitality. At this point, everyone is feeling the effects. Virtually all businesses will be affected. Professional service businesses, manufacturers, distributors, physicians, daycare operators, hoteliers, restaurateurs, auto repair, assisted-living, and many, many more. These types of businesses that are privately held and for profit would be best suited for SBA 7(a) loans for relief support. These loans go up to $10 million, which can help relieve a large financial burden such as payroll and other expenses.
Read Also: Coronavirus Raises Commercial Real Estate Uncertainty
What measures can small business owners take to minimize the impact?
Hurn:The biggest challenge will be recognizing their immediate need for relief and acting swiftly. They may fear that they need this, but instead of being immobilized, take immediate action. Waiting to see if things get better and simply hoping they will could delay critical funding. Business owners should get organized now for their loan submissions. This means having business tax returns, personal tax returns, interim financial statements (balance sheet and profit-and-loss statements), personal financial statements, business debt schedules as well as a punch list of how they'll use the proceeds, all ready to be submitted.

Are you seeing any significant increase in loan demand at this point? What are your expectations going forward regarding demand from borrowers?
Hurn: Yes, we're seeing a demand on a level that I would never have imagined. Fountainhead has seen over 2,500 loan inquiries, which is unprecedented for our company. We are hearing that business owners, justifiably, are very concerned with how to make payroll, especially if they're required to pay for extended leave for their employees. Fountainhead is one of the 14 non-bank lenders in the country equipped to handle these loans.
Would you advise businesses to keep spending money to keep things going at this time or to implement restrictions?
Hurn: The prudent action right now is to restrict spending.
What is the advantage of an SBA loan in these circumstances compared to other lending alternatives? Hurn: We expect no fees and delegated underwriting, which means expedited processing with faster approval than other lending alternatives. These SBA 7(a) loans will primarily be for working capital purposes, so the loan proceeds will be used to stabilize the financial condition of businesses that are impacted by the economic fallout due to the virus. Working capital proceeds can be used to meet payroll, pay payables, buy inventory, make critical repairs, purchase equipment, make leasehold improvements and/or other renovations and so forth.
How fast can an emerging business be approved for an SBA recovery loan and how can companies like Fountainhead help speed up the process?
Hurn: As of right now, SBA personnel will attempt to process these requests directly, which will take time. This may delay the inevitable decision to enlist private sector experts, such as Fountainhead, to help with this matter. SBA lenders can expedite capital for affected small businesses to help with recovery, and time is of essence right now. We can't afford to wait weeks for these new rules to get into effect as too many businesses will lay people off and shut down for good. Everyone who can make a difference here has to hustle.
How fast (or slow) do you expect the recovery to be?
Hurn: I am cautiously optimistic for a snapback recovery once we contain the virus. The response to the outbreak is what is causing us to have the economic calamity that we have now. When the outbreak is contained, I would expect that the economy will recover and truly bring a new meaning to cabin fever.

A.2 Document 1390

Why some small businesses may not get coronavirus relief - Los Angeles Times
On Friday, banks across the country began accepting applications from struggling small businesses desperate to get a piece of the $2-trillion stimulus package authorized by the CARES Act. Unfortunately, many of those small businesses — and perhaps those that need the money most — may be left waiting at the back of the line as others exhaust the available funds.
The key provision of the CARES Act for small businesses is the Paycheck Protection Program, or PPP, which seeks to inject an immediate $350 billion into small businesses struggling with the COVID-19 pandemic. It's the small business version of the $1,200 in stimulus checks heading to individual taxpayers.
The program works by giving a small business a loan equal to 250% of its average monthly payroll, and the loan can be forgiven if 75% of the proceeds are spent on payroll in the first eight weeks of the loan. The idea is to throw a lifeline to these businesses and their employees. If you run a small business that has struggled because of COVID-19 — and what small business hasn't — you should be looking to apply for a PPP loan.
But the Paycheck Protection Program is a first-come, first-served deal. And so far, many of the smallest of small businesses — the coffee shop around the corner, or your favorite neighborhood food truck or nail salon — are finding themselves at the back of the line.
The problem is the rules for distributing the money. Congress chose to dispense the PPP $350 billion through a network of approximately 2,000 banks authorized to make Small Business Administration loans. The CARES Act instructs these banks to approve PPP applications quickly regardless of a business' credit risk. But the banks are still banks, and Department of Treasury rules going back to the 1970s require that they meet "know your customer" regulations meant to prevent money-laundering and terrorist financing.
In the rush to implement the CARES Act bailouts, the applications from customers the banks already "know" are the ones getting accepted and processed. Small businesses that have never been vetted for a loan, or that don't already have an account at an SBA-approved lender will face long delays as the bank studies their past cash needs, confirms their corporate structures, and verifies the identities of their owners. And cash-intensive businesses, like local cafes and salons, where the risk of money-laundering is greater, could get particular scrutiny. Many banks say they expect to accept PPP applications from new customers in the coming days and weeks, but will the funds still be there?
....
The Treasury Department should act now to ensure that small businesses do not lose out in the race for PPP funds simply because they don't already have a bank loan and a preexisting relationship with a Small Business Administration lender. Otherwise, by the time your neighborhood cafe makes its way to the front of the PPP line, the $350 billion may be depleted and the cafe, its owners and its employees may be out of luck. And once we're all free again to go out for breakfast, lunch and dinner, we may be too.

A.3 Document 1470

Senate dives into negotiations on trillion-dollar stimulus bill as economic calamity grows
WASHINGTON - The Senate dove into intense negotiations Friday over a trillion-dollar stimulus bill to save the economy from collapsing under the ravages of the coronavirus, aiming to reach bipartisan agreement by the end of the day so the bill can pass early next week.
Despite multiple disagreements over the structure of the bill released Thursday, Senate Majority Leader Mitch McConnell, R-Ky., said agreement by day's end was imperative on the legislation that would direct hundreds of billions of dollars into the hands of individual Americans, small businesses and industries like airlines clobbered by the crisis.
...
The piece of the bill that appeared to enjoy the most widespread and bipartisan support is the small-business section, drafted by Sen. Marco Rubio, R-Fla., which offers loans to small businesses with under 500 employees. The $300 billion for the loans would be made available through lenders certified by the Small Business Administration, such as banks and credit unions, with the maximum loan capped at $10 million. The portion of the loan used by the small businesses to cover their payrolls could be forgiven if firms retain their employees through the end of June 30. Loans given to firms with tipped employees, such as bars and restaurants, could be forgiven if they are used to provide additional wages to their employees.
As negotiations progressed Friday, Rubio told reporters they're looking at loosening the SBA definition of small business - currently set at 500 - to ensure the legislation doesn't leave out businesses that are somewhat larger than that size, but also don't qualify for a separate loan program aimed at major industries like airlines.
...

A.4 Document 5056

Coronavirus: More than 140,000 firms claim wage bill help - BBC News
More than 140,000 firms have applied for help to pay their wage bill through the government's job retention scheme, which went live on Monday.
The programme funds 80% of workers' wages, up to £2,500 a month, if they are put on leave.
Speaking at the Downing Street press briefing, Chancellor Rishi Sunak said the money would help pay the wages of more than a million people.
But many more than that are expected to be "furloughed" due to the lockdown.
The Treasury said the system can process up to 450,000 applications an hour. Employers should receive the money within six working days of making an application, it said.
The chancellor said the scheme would help people who could have lost their jobs if they had not been furloughed.
Employers had made 67,000 job claims within half an hour of the system going live at 08:00, HMRC chief executive Jim Harra told the BBC's Today programme.
...
According to new research by the Resolution Foundation, the take-up of the scheme has been higher than initially anticipated.
It estimates that eight million workers could be furloughed over the coming weeks.
It found that those working in low-paid sectors - such as hospitality or retail - are worst-affected, with almost half of the workforce expected to be put on paid leave.
Daniel Tomlinson, economist at the Resolution Foundation, said: "The government's welcome Job Retention Scheme is what stands between Britain experiencing high unemployment over the coming months, and catastrophic depression-era levels of long-term joblessness.
"It is proving particularly essential in big, low-paying sectors like hospitality and retail, where around half the workforce are no longer working."

B Document information table

Document	Topic ID	Topic	Sentiment Score
Reference	6	'0.028*"business" + 0.012*"small" + 0.011*"president" + 0.010*"small_business" + 0.010*"chamber" + 0.009*"program" + 0.008*"american" + 0.008*"company" + 0.007*"act" + 0.007*"congress"	0.126
Doc 1390	6	'0.028*"business" + 0.012*"small" + 0.011*"president" + 0.010*"small_business" + 0.010*"chamber" + 0.009*"program" + 0.008*"american" + 0.008*"company" + 0.007*"act" + 0.007*"congress"	0.081
Doc 1470	6	'0.028*"business" + 0.012*"small" + 0.011*"president" + 0.010*"small_business" + 0.010*"chamber" + 0.009*"program" + 0.008*"american" + 0.008*"company" + 0.007*"act" + 0.007*"congress"	0.167
Doc 5056	41	'0.025*"business" + 0.018*"support" + 0.008*"customer" + 0.008*"information" + 0.007*"advice" + 0.007*"pay" + 0.007*"provide" + 0.006*"service" + 0.006*"scheme" + 0.006*"grant"	0.104

References

[1] Reality Check team. *Coronavirus: What misinformation has spread in Africa?* Apr. 2020. URL: https://www.bbc.com/news/world-africa-51710617.

[2] Angela Giuffrida. *Italian minister tries to calm coronavirus panic and attacks profiteers.* Feb. 2020. URL: https://www.theguardian.com/world/2020/feb/27/italian-minister-tries-to-calm-coronavirus-panic-and-attacks-profiteers.

[3] David MJ Lazer et al. "The science of fake news". In: *Science* 359.6380 (2018), pp. 1094–1096.

[4] Soroush Vosoughi, Deb Roy, and Sinan Aral. "The spread of true and false news online". In: *Science* 359.6380 (2018), pp. 1146–1151.

[5] *Automated Fact Checking.* URL: https://fullfact.org/automated.

[6] Giovanni Luca Ciampaglia. "Fighting fake news: a role for computational social science in the fight against digital misinformation". In: *Journal of Computational Social Science* 1.1 (2018), pp. 147–153.

[7] Sylvie Cazalens et al. "A content management perspective on fact-checking". In: *Companion Proceedings of the The Web Conference 2018.* International World Wide Web Conferences Steering Committee. 2018, pp. 565–574.

[8] D Graves. "Understanding the promise and limits of automated fact-checking". In: (2018).

[9] Naeemul Hassan, Chengkai Li, and Mark Tremayne. "Detecting check-worthy factual claims in presidential debates". In: *Proceedings of the 24th acm international on conference on information and knowledge management.* 2015, pp. 1835–1838.

[10] Naeemul Hassan et al. "Toward automated fact-checking: Detecting check-worthy factual claims by ClaimBuster". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 2017, pp. 1803–1812.

[11] Marco Lippi and Paolo Torroni. "Context-independent claim detection for argument mining". In: *Twenty-Fourth International Joint Conference on Artificial Intelligence.* 2015.

[12] Casper Hansen et al. "Neural check-worthiness ranking with weak supervision: finding sentences for fact-checking". In: *Companion Proceedings of the 2019 World Wide Web Conference.* 2019, pp. 994–1000.

[13] Ayush Patwari, Dan Goldwasser, and Saurabh Bagchi. "TATHYA: A multi-classifier system for detecting check-worthy statements in political debates". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management.* 2017, pp. 2259–2262.

[14] Xuezhi Wang et al. "Relevant document discovery for fact-checking articles". In: *Companion Proceedings of the The Web Conference 2018.* 2018, pp. 525–533.

[15] Georgi Karadzhov et al. "Fully Automated Fact Checking Using External Sources". In: *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017.* Varna, Bulgaria: INCOMA Ltd., Sept. 2017, pp. 344–353. DOI: 10.26615/978-954-452-049-6_046. URL: https://doi.org/10.26615/978-954-452-049-6_046.

[16] Baoxu Shi and Tim Weninger. "Discriminative predicate path mining for fact checking in knowledge graphs". In: *Knowledge-based systems* 104 (2016), pp. 123–133.

[17] Giovanni Luca Ciampaglia et al. "Computational fact checking from knowledge networks". In: *PloS one* 10.6 (2015).

[18] Valeria Fionda and Giuseppe Pirrò. "Fact checking via evidence patterns". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence.* 2018, pp. 3755–3761.

[19] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[20] Hamed Jelodar et al. "Latent Dirichlet Allocation (LDA) and Topic modeling: models, applications, a survey". In: *Multimedia Tools and Applications* 78.11 (2019), pp. 15169–15211.

[21] Qiaozhu Mei et al. "Topic sentiment mixture: modeling facets and opinions in weblogs". In: *Proceedings of the 16th international conference on World Wide Web.* 2007, pp. 171–180.

[22] Michael Röder, Andreas Both, and Alexander Hinneburg. "Exploring the space of topic coherence measures". In: *Proceedings of the eighth ACM international conference on Web search and data mining.* 2015, pp. 399–408.

[23] Saif M Mohammad. "Sentiment analysis: Detecting valence, emotions, and other affectual states from text". In: *Emotion measurement.* Elsevier, 2016, pp. 201–237.

[24] Saif M Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. "Stance and sentiment in tweets". In: *ACM Transactions on Internet Technology (TOIT)* 17.3 (2017), pp. 1–23.

[25] Jianhua Yuan et al. "Exploring Answer Stance Detection with Recurrent Conditional Attention". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 7426–7433.

[26] Roy Bar-Haim et al. "Stance classification of context-dependent claims". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. 2017, pp. 251–261.

[27] Amit Saxena et al. "A review of clustering techniques and developments". In: *Neurocomputing* 267 (2017), pp. 664–681.

[28] Dongkuan Xu and Yingjie Tian. "A comprehensive survey of clustering algorithms". In: *Annals of Data Science* 2.2 (2015), pp. 165–193.

[29] Maitri P Naik, Harshadkumar B Prajapati, and Vipul K Dabhi. "A survey on semantic document clustering". In: *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. IEEE. 2015, pp. 1–10.

[30] Prafulla Bafna, Dhanya Pramod, and Anagha Vaidya. "Document clustering: TF-IDF approach". In: *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE. 2016, pp. 61–66.

[31] Bweijunl Zheng, Wei Zhang, and Xiaoyu Fu Boqin Feng. "A survey of faceted search". In: *Journal of Web engineering* 12.1&2 (2013), pp. 041–064.

[32] Wisam Dakka and Panagiotis G Ipeirotis. "Automatic extraction of useful facet hierarchies from text databases". In: *2008 IEEE 24th International Conference on Data Engineering*. IEEE. 2008, pp. 466–475.

[33] Chengkai Li et al. "Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia". In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 651–660.

List of Figures

List of Tables