Imperial College London

MEng Individual Project

Interim Report

Imperial College London

Department of Computing

# Semantic Analysis of News Corpora

*Author:*
Akanksha Sharma

*Supervisor:*
William Knottenbelt

*Second Marker:*
Paul Bilokon

June 8, 2022

**Abstract**

Your abstract goes here

**Acknowledgements**

I would like to thank my supervisor Professor William Knottenbelt for his genuine enthusiasm, constant guidance throughout this project and for giving me the creative freedom for the direction of this project. I would also like to thank the team at DeepSearch Labs, in particular, Miss Mariam Torshizi for her constant support and feedback.

Finally, I would like to thank my family for their love and encouragement, which made this project possible.

# Contents

# Chapter 1

# Introduction

In today's fast-growing economy, there is an incomprehensible amount of information from a host of different sources readily available at our disposal. This can be extremely overwhelming and difficult to navigate. There is a need for a concise and digestible medium of information that allows the consumer to gain insight into the major themes and events of the moment, how they relate to one another, and how they are likely to affect key industry players by means of data visualisation. Data visualisation can help make the data more accessible and available.

This project tries to solve this problem of information overload in the business and industry sectors (such as airline, energy and pharmaceutical industry) by providing a visualisation tool that dynamically depicts how the main contributors (companies, people, events) within an industry are affected by the news and what the general sentiment surrounding these contributors are.

Such a product has great value and interest for not only the average civilian who does not want to sift through reams of news to gain a core insight into the developments within an industry but also for the businesses that want to obtain an intelligent overview of workings within specific industries.

## 1.1 Motivation

## 1.2 Objectives

## 1.3 Contributions

1: Background: LDA

2: https://link.springer.com/content/pdf/10.1186/s13673-019-0192-7.pdf : FOR TF-IDF and BoW

3: Background: Remove visualisation

# Chapter 2

# Background

This section highlights the key concepts that are needed to develop this project. It covers concepts including Knowledge Graphs, Natural Language Processes such as Named Entity Extraction, Sentiment Analysis, Topic Modelling that are used to extract key information from the data, understanding Word Representation as well as ways key considerations to visualise data.

## 2.1 Natural Language Pre-Processing Techniques

First and foremost, in order to extract information (entities, relationships) from unstructured data (plain-text, e.g. news articles), the input data needs to undergo some pre-processing techniques. The combination and order of these techniques is often dependent on the use case.

Some of the most widely used pre-processing techniques in Natural Language processing include, but are not limited to, stop-word removal, tokenisation, part-of-speech (POS) tagging, normalisation (using lemmatisation and/or stemming), sentence splitting, chunking and dependency parsing [1] [2].

An example of the pipeline of showing different aspects of text preparation is as follows:

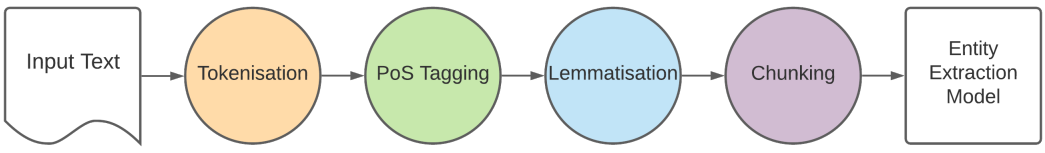


**Figure 2.1:** Text preparation pipeline

### 2.1.1 Tokenisation

Tokenisation (as seen in Figure 2.2 involves breaking down the sentence to retrieve fragments called 'tokens' which are pre-defined elements. These can include words, keywords, phrases, or symbols/ punctuation) depending on the application [1] [2]. Once the tokens are obtained, often some filtering methods, such as Stopword removal, are applied to prune any unnecessary tokens using a pre-determined set of words (often called stoplist). This list of words is not fixed but often contains words such as 'are', 'this', 'that' etc. which are generally not crucial for document classification approaches [1]. The questions of whether stopwords should be removed and/or which stopwords to remove are often dependent on the data mining problem.

### 2.1.2 Part of Speech (PoS) tagging

Part-of-speech (POS) tagging, also known as grammatical tagging, assigns 'parts-of-speech' to words in a text. It uses word and grammar structure taking into account the context around words to determine their characteristics, for instance, identifying a word as a noun, verb, preposition etc. [3] POS tagging often assumes some sort of tokenised text upon which it makes the grammatical tag classifications as seen in Figure 2.2. In the figure, the tagger identifies Emirates and Bitcoin as PROPN (proper nouns), airlines and payments as NOUN and as accept as VERB. POS tagging is a critical component of many NLP systems. It provides linguistic information about the text and enables information extraction from text corpora in order to identify key entities and relationships. The set of tags depends on the model used and can involve course and/or fine classification.

4: Add appendix for tag set

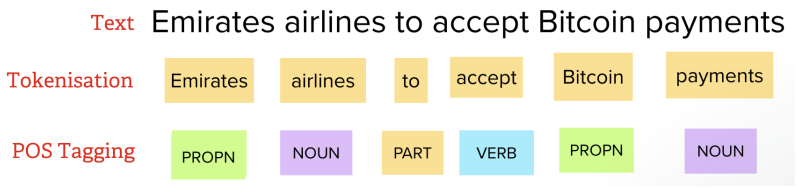


**Figure 2.2:** Tokenisation and POS tagging

### 2.1.3 Normalisation

Information retrieval focuses on getting or providing users with easy access to the information they need. It does not only look for the right information but represents it in a manner that is easily understandable to users, stores the information in an orderly manner and organises it in such a way that it can be easily retrieved at a later time

Normalising data is a key step in information extraction from the text. As the size of the data increases and it becomes especially important to represent data in a standard manner and reduce randomness [4]. There are two common methods of normalising data:

1. **Stemming** is the process of reducing words (or tokens) to their word stem or root form [4]. One of the most common algorithm for stemming english words is the Porter's algorithm. It makes use of a minimal length measure which is derived from the number of consonant-vowel-consonant strings that remains after a suffix is eliminated [5]. The main drawback of stemming is that it relies on a crude heuristic process to condense related words into a single stem, even if it is not a dictionary word. This can result in errors caused by under-stemming or over-stemming [6]. As an example of the latter, the words 'participation' and 'participate' may get reduced to 'participat' which is not an actual word.
2. **Lemmatisation** is a form of normalising the data by matching words with their canonical (dictionary) forms called lemmas by reducing inflictive variants to a 'root' word/token [4]. Example: walking, walks, walked will all reduce to walk. Lemmatisation makes use of POS tags and word/ sentence structure. This is an improvement over stemming as the base words are actual words and produces much better results for language modelling as described in [4].

### 2.1.4 Chunking

Chunking is a process which essentially makes use of POS tags by attaching additional information to the sentence breaking it down into its constituent phrases. There are usually 5 major categories of extarcting chunks from text: Noun Phrase (NP), Verb phrase (VP), Adjective phrase (ADJP), Adverb phrase (ADVP) and Prepositional phrase (PP). For example, the sentence "The large truck is going under the tunnel" will be broken down into a noun phrase: The large truck, verb phrase: is going, prepositional phrase: under the tunnel as in Figure 2.3.

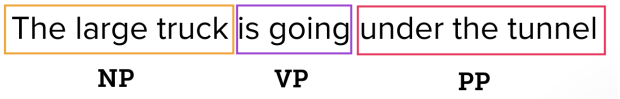


**Figure 2.3:** Chunking

## 2.2 Dependency Parsing

5: Write up dependency parsing

## 2.3 Coreference Resolution

Coreference resolution refers to the task of ascertaining which linguistic expressions (known as mentions) in a natural language refer to the same real-world entity (such as a person or thing) [7]. It is often an important step for other high-level tasks such as question answering, document summarisation and information extraction. The latter use case

is particularly relevant to this project. When two or more expressions, refer to the same entity, they share the same referent [8]. The goal of the coreference resolution algorithms is to find and cluster the "mentions" by their referent. For instance, in the sentence, "The U.K said they would relax travel bans", the mentions "U.K." and "they" refer to the same entity: U.K.

Coreference resolution is not a trivial task as it relies on both the semantic and syntactic meaning of the text. For example, in the sentence, "Alice said she would come", the mention "she" may or may not refer to Alice. This indicates the complexity of the coreference resolution task as it requires contextual information, real-world knowledge, a grammatical understanding of the language etc [7].

There are multiple different scenarios when exploring coreference. Some of the common ones include:

1. **Anaphora coreference:** Anaphoric references refer to the previous entities (the antecedent) for their meaning. For instance, in this instance "the aviation industry to find its way back in 2022", the anaphor 'its' follows the expressions it refers to, the antecedent: 'the aviation industry'.
2. **Cataphora coreference:** These references refer to entities/mentions later in the text. For example, in the sentence, "To aid their declining sales, Emirates is set to lower their fares", the cataphor "their" precedes the entity/expression it refers to, the postcedent: 'Emirates'.
3. **Split antecedents/ Multiple Antecedent Coreference:**

   These references/words have multiple antecedents that they refer to. For instance: in the sentence "British Airways and United Airlines set to resume their direct flights to Australia", the anaphor 'they' has a split antecedent: 'British Airways' and 'United Airlines'.

### 2.3.1 Span detection

The primary step of coreference resolution is mention detection. This involves finding the 'spans' i.e., the combination of words that constitute a mention. Generally, candidate spans cover NP (noun parts of the text), possessive pronouns and named entities [9] (discussed in detail in Section 2.5.1) . In order to detect spans, there are several different options for span ranking architecture. Most models are more liberal in detecting candidate spans at the initial stages and then apply some filtering and/or augmenting mechanisms to get the most relevant spans. This can involve pruning candidate spans based on a threshold ranking score, considering only a pre-determined K antecedents for each mention [10] or applying the span-ranking to the text in an iterative manner to update the span representations using prior antecedent distributions to allow later coreferences to be influenced by earlier ones [10].

6: Add images:maybe

### 2.3.2 Coreference Architecture

Once the candidate spans are extracted from the text, the next step is to resolve coreferences of mentions. For the purpose of this project, the most widely used architectures are focused on:

#### Mention-pair architecture

This is one of the simplest approaches and involves a binary classification task on a pair of mentions and/or entities. The classifier is given a pair of mentions, a candidate antecedent and a candidate anaphor and decides whether they are coreferring based on a score.

#### Mention ranking architecture

This type of architecture directly compares the candidate antecedents with each other, selecting the antecedent with the highest score for each anaphor. This approach is more complicated than the mention-pair model as for each anaphor, the best possible 'gold' antecedent is not known rather a cluster of 'gold' antecedents is known. In earlier models, the 'gold' antecedent was chosen to be the closest one.

Generally, the simplest approach to give credit to any 'legal' antecedent is by adding them together and using a loss function that optimises the likelihood of all correct antecedents in the 'gold' cluster of antecedents [9]. This is seen in Lee 2017 [11], a model on which a lot of recent state of the art models such as SpanBERT [12] are based on. It uses the mention ranking architecture to determine a conditional probability distribution (seen in equation 2.1) to give a configuration with the highest likelihood representing the correct clustering.

$$P(y_1, ..., y_n|D) = \prod_{i=1}^{N} P(y_i|D) \tag{2.1}$$

$$P(y_i) = \frac{\exp(s(x_i, y_i))}{\sum_{y' \in Y(i)} \exp(s(x_i, y'))} \tag{2.2}$$

$$\text{where} s(x_i, y') = s_m(x_i) + s_m(y_i) + s_c f(x_i, y_i)$$

In equation 2.2, goal of the task is to assign an antecedent to each span $x_i$ in document D. $s(i, j)$ is a pairwise score which depends on three factors: $s_m(i)$, how likely span x is to be a mention; $s_m(j)$, how likely span y is a mention and $s_c f(i, j)$, joint coreference probability of spans i and j in document D (assuming they are both mentions) referring to the same entity [11][10].

## 2.4 Word Representation

The general idea behind word representation is to convert the text into an understandable format for the computer. This is done by word embedding which learns a vector representation for words.

### 2.4.1 Types of embeddings

There are three main types of algorithms used for encoding words:

1. **Bag-of-Words (BOW):** The simplest approach for word encoding is the Bag-Of-Words approach where the general concept is to generate a dictionary of tokens from the text (say, news articles) by pre-processing the text using techniques such as Stop-Word removal, lemmatisation and Named Entity Detection (NER) and then represent the documents/ news articles as a vector of the occurrences of those tokens in the text. In other words, the $j_{th}$ element in the vector (say 5) represents the fact that the $j_{th}$ token in the dictionary (say 'London') appeared 5 times. This method is fairly primitive as it does not account for any grammatical rules and is an unordered representation. It falls short with increasingly high volumes of data as it is simply a syntactic representation and does not account for any semantic/ conceptual relations within the text corpus. [31]

2. **Word2Vec:** Word2Vec algorithms provide a much better way of representing words by using the concept of similarity of words. The core idea is that words that are syntactically and semantically close should have similar vector representations and thereby occupy similar spatial positions. This degree of similarity is calculated using the cosine similarity (cosine of the angle between two vectors). [33]. They also exploit the 'locality hypothesis' which centers around the idea that words that appear together (or close to) identical words will be spatially close. [31]

   For simplicity's sake, let's say the word 'knows' is represented by a 4-dimensional binary vector [0,0,1,0], the vectors for 'knew', 'known' to have a vector representation of [0,0,1,0] where the third dimensions groups these words. Similarly, there can be a feature (dimension) (e.g., $4_{th}$) representing a word type or category such as airlines and so, continuing the example 'Emirates', 'Etihad' and 'British Airways' should also similar vector representations with the $4_{th}$ dimension (feature) having a value of 1, thereby assigning them to the same group (cluster) [32]

   These Word2Vec embeddings are commonly in the Neural Network models: Continuous Skip-Gram and Continuous Bag-of-Words (CBoW). [33]

   The challenges with these word embeddings are that they provide a single context-independent representation for a word and struggle with "out-of-vocabulary (OOV) words" [31], i.e., words that were never encountered prior will often be represented as a random vector which is not ideal.

3. **Context2Vec:** An improvement over Word2Vec is Context2Vec. This accounts for

polysemy, the fact that in different contexts, words can have different meanings (or senses). This allows for a context-independent representation for a word, called ”contextual word vectors” which encapsulate the meaning for a word in a particular context. This is particularly useful as these representations can derive the meaning of a word in a specific context. For instance, in the context of the sentence ”I ate a banana split”, ’split’, is associated with food. [32]

This approach makes use a bi-directional (left-to-right and right-to-left) LSTM (long short-term memory) recurring neural network. This network makes use of N layers of LSTM, where the lower levels extract low-level features such as POS tagging and the upper levels learn the contextual meaning of words. [31] One of the most popular approaches for this is ELMo (Embeddings from Language Models) which is discussed in Section 2.6.2. It is also one the models used in AllenNLP which will be used for the purpose of this project.

### 2.4.2 Embeddings from Language Models (ELMo)

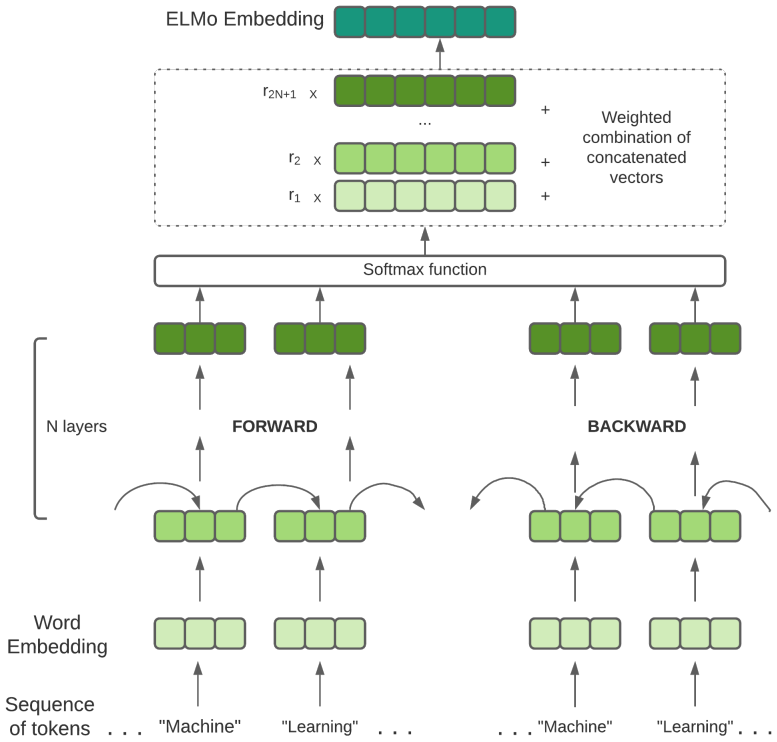


**Figure 2.4:** Embeddings from Language Models (ELMo) Context Representation

The outline of the ELMo model is as follows: [34]

1. The network consists of N bi-directional (left-to-right and right-to-left) LSTM layers and the input is a sequence of N words/ tokens.

2. The Forward language model gets a sequence of words/ tokens (of arbitrary length) from left to right and calculates the probability of the sequence of words $P(w_1, w_2, ...w_N)$ by making use of the history of words/tokens it has seen earlier. [31]

   $$P(w_1, w_2, ...w_N) = \prod_{i=1}^{N} P(w_i|w_1, w_2, w_{i-1})$$

   For a target word/token $w_i$, let the context-dependent vector from this (forward) language model (at layer l = 1..N) be represented as LRV(i, l), which contains information about this word given the context of words appearing before the target word.[31]

3. Similarly, the backward model computes the probability of the reverse sequence of words as follows:

   $$P(w_1, w_2, ...w_N) = \prod_{i=1}^{N} P(w_i|w_{i+1}, w_{i+2}, w_N)$$

   For a target word/token $w_i$, let the context-dependent vector from this (backward) language model (at layer l = 1..N) be represented as RLV(i, l), which contains information about this word given the context of words appearing after the target word. [31]

4. The outputs from these models are passed to the subsequent layer of their respective models. A non-linear activation function (for example, ReLU) can be applied for intermediate layers. Softmax is applied to the outputs from the last LSTM layers of the 2 models. [34]

5. The objective function is to jointly maximise the log-likelihood in both forward and backward directions.

6. For each word $w_i$, a N-layer biLM model will have a total of 2N+1 (including the input sequence of tokens layer) vector representations. The final vector (ELMo) which will be passed to the NLP training model, will combine these 2L+1 vectors by using a weighted sum of these vectors as seen in Figure 3. This is the ELMo embedding. [31]

## 2.5 Named Entity Extraction

Extracting named entities from texts and representing them as nodes in knowledge graphs generally, involves three main tasks: Named Entity Recognition (NER), Named Entity Disambiguation (NED) and Named Entity Linking (NEL).

### 2.5.1 Named Entity Recognition (NER)

**Named entity recognition (NER)** is a process which extracts and classifies named entities of a certain (sometimes pre-defined) types from an unstructured text. [9]

First, it identifies the 'names' of the entities from the tokens (pre-processed) which is commonly done by BIO (beginning-inside-outside) tagging and POS tags. It then performs sequence labelling (assigning labels/tags to each element of an input sequence) and classify entities into the predefined types such person, location, organisation etc. as seen in Figure 2. [9]

A popular sequence labelling model is **Conditional Random Fields** (CRF). These are "discriminative models" and generally outperform other Markov Models and such as Hidden Markov Models (HMM) as they are better able to cope with unseen tokens/words as well as Maximum Entropy Markov Models (MEMM) as they do not have labelling bias. [13]

They work by maximising the log-likelihood of the posterior distribution P(S—O) where S is the output label sequence and O is the observed input sequence. Let S' denote the correct label sequence:

$S' = \text{argmax}_S \, P(S|O)$

Therefore, it is fairly simple to determine output label sequence as it will be the one with the highest posterior probability. E.g. P([Per, Org, Org] — [Obama, UN, Congress]) will have a much higher probability than P([Loc, Loc, Loc] — [Obama, UN, Congress]). [13]

The NER models are usually trained on specific domains and thereby only extract certain types of entities (pre-defined types such as PER, LOC, ORG), thereby making them domain-dependent.

### 2.5.2 Named Entity Disambiguation (NED)

**Named Entity Disambiguation (NED)**: As seen in **Figure 2**, NER might not always be able to classify a named identity due to it having a different meaning in different contexts. This is why named entity disambiguation is crucial to determine which named entity a mention refers to; For instance, 'Trump' can refer to either a person, a corporation or a building. [9]

This is where context representation comes into play which uses the knowledge base to learn about the entities. In order to disambiguate words, models use as textual and structural representations. Examples of textual representations are bag-of-words (BoW) approach where for each ambiguous entity, it keeps track of all concatenated paragraphs the entity was mentioned in and compares them with a target article, selecting the best matching. Vector space model and cosine similarity can be used for text comparison and TF-ICF (term frequency-inverse corpus frequency, more efficient version of TF-IDF [12]) for weighting individual terms. [11]

In structural representation methods, the context can be represented as the entirely of the text. An example in [11] (p.7) shows that for the sentence: "Michael Bloomberg is the mayor of New York", their algorithm correctly identifies New York in USA instead of London as Michael Bloomberg co-occurs in the same paragraph New York city in the USA in the KB significantly more "(88 times) than with the New York in England (0 times)". Quantifying the impact of co-occurrences, can be done by using incidence matrix represents a

weighted graph where weights are the co-occurrence |P($e_i$,s,$e_j$,t)| i.e. count of paragraphs, where two different entities $e_i$ and $e_j$ were mentioned together in two different sentence forms (s $\neq$ t). [11]

## 2.6 Topic Modelling

Topic modelling is essentially finding patterns in collections of data (in our case, news articles). The motivation behind topic modelling for this project would be to condense the key themes/ topics per industry from vast collections of news articles (grouped by industry). [24]

Topic models are essentially based on Bayesian Networks and "assume that shared global multinomial word distributions (i.e., topic distributions) govern the corpus" [23](p.2). Each of these documents (news articles) will contribute to the frequencies of a word in a document which is derived from a mixed model of the topic distributions.

### 2.6.1 Main Approaches

There are 2 main methods for topic modelling:

1. **Latent Semantic Indexing (LSA):** often implemented as pLSA (probabilistic Latent Semantic Indexing) works by creating a semantic space from large collections of text. This space is then used can then be used to detect similarity among words, topics, or even entire documents. This semantic space is a large high-dimensional vector and is essentially a term-document matrix (with columns representing documents and rows representing unique words/topics). To reduce the high dimensionality of the vector, often dimensionality reduction methods such as Singular Value Decomposition (SVD) or Principle Component Analysis (PCA) are used. A drawback of LSA is that it makes the orthogonality assumption that each document is about one thing which intuitively is not true for most documents. Another limitation is that these methods may require a way to interpret the high-dimension vector as it has minimal legibility value for humans. [24]

2. **Latent Dirichlet Allocation (LDA):** The other much more popular model is LDA. It is a hierarchical Bayesian model, where each document is modelled as a mixture (multinomial distribution) of topics with each topic itself being a mixture of (multinomial distribution) of words. [25]. It is important to note that each topic can have multiple words, every word in the text corpus is tagged with a single topic. This essentially means that a word's presence can be attributed to one topic's distribution (even though in reality it could be present in the text corpus as a result of multiple topics).

   Additionally, LDA ensures sparsity in the underlying multinomial distributions by making use of the Dirichlet prior distribution. This aids in the interpretations of the extracted topics. [23]

In order to extract the key topics within a volume of data (for the scope of this project, news), some sort of metric is needed to extract the relevance of a topic. One such approach is highlighted in this paper. [23](p.3)

It defines $I_k$ (t), the news on a day t associated with a topic k as the "total numbers of words tagged with topic k on day t".

$$I_k(t) = \sum_{I(t)} \sum_w N(d, w, k)$$

where N(d,w,k) represents the frequency of word w (tagged with topic k) in the document d. I(t) is the represents the documents/ news articles obtained on day t. Equation cited from paper. [23](p.3) The topics for which $I_k(t)$ returns the largest values can be thought of as 'most relevant' for that day. Additionally, the relevance of a topic k over time can also be determined by computing this value for different days (changing t).

### 2.6.2 Considerations for Topic Modelling

An issue that needs to be considered when extracting topics for large volumes of news articles is that majority of them might have repeated and unwanted phrases such as "Top News" or "Read Similar Articles" that may be extracted as topics even though they have no intuitive relation to the news pertaining to a specific industry. Eliminating these phrases may be computationally heaving and require extensive parsing, as well as require an algorithm that accounts for all variations of such phrases. Therefore, a better approach to avoid this as discussed in [23] is to prune topics based on their distributions by focusing on top N (for instance, 6) words associated with a topic distribution and eliminate this topic if any of these N words are in a pre-defined dictionary of words derived from unwanted phrases.

Topics modelling can be used in conjunction with sentiment analysis, whereby a single joint model performs both i.e. extracts topics and sentiments. This idea stems from the notion that every opinion has an associated target (quadruple discussed in Section 2.4). Based on concepts mentioned earlier, topics modelling can be used to extract aspects which can be topics or sentiments. However, there will be no differentiation between the two. To combat this, some sort of an indicator variable may be used to make that distinction. [16]

## 2.7 Sentiment Analysis

Sentiment analysis is used to express whether a piece of text (can be about a 'target entity' in the text, topic in the text, a sentence or the whole document) implies a positive, negative or neutral sentiment. Sentiment analysis (or opinion mining) requires extracting the semantic orientation (polarity and strength) of the text. An example of this can be done by assigning a score in the range of [-1,1] where +1 can indicate an (extremely) positive sentiment and -1 can indicate an (extremely) negative sentiment with 0 being neutral. [16] [18]

It is important to understand the structure of the sentiment (or opinion). Using the definition in this book [16], an 'Opinion' represents a "*quadruple (g, s, h, t) where g is the sentiment target, s is the sentiment of the opinion about the target g, h is the opinion holder (the person or organization who holds the opinion), and t is the time when the opinion is expressed.*" Furthermore, g can be split into entity (e) and aspect of entity ($a_i$), where each entity can have multiple aspects and the sentiment is based on the aspects of entities and not the entities as a whole. This allows entities to have multi-faceted sentiments. [18]

The main steps of sentiment analysis involve:

1. Pre-processing the raw text (e.g. news articles) to using techniques such as lemmatisation, Parts-of-Speech Tagging (POS) and Stop Word Removal to break down each text document into its components (such as phrases, tokens etc.) [17]
2. Identifying sentiment bearing phrases and assigning it a sentiment score, which can then be combined to assign a multi-layered sentiment.

For this project, I will do sentiment analysis on each of the key entities in news (pertaining to specific industries.) Therefore, it may be useful to use the notion of subjectivity (strength) and polarity scores at global and entity levels as mentioned in this [paper. 22] where world scores use "total references" (denominator) to mean all the references of the entity from a history of news articles (global level) not just those references extracted from processing the news on a single day (entity level).

In the case of news articles, some of the automatic opinion mining systems can usually associate entities mentioned in the context of negative articles with a negative sentiment even if the entity may have acted positively. For Example, if an airline company was giving out free upgrades to those whose flight got cancelled without notice due to the pandemic, the airline might be associated with a negative sentiment due to the nature of the news (cancelled flights, pandemic) surrounding that airline even though they were being generous to their customers (which would warrant a positive sentiment). Therefore, often, when dealing with sentiment analysis, models consider windows of variable size surrounding these entities as discussed in this paper [20] to gauge a better context of the entity for determining the sentiment associated with it.

### 2.7.1 Sentiment Analysis Approaches

Generally, there are two approaches to sentiment analysis:

**1. Rule-based sentiment analysis** These techniques are rules and dictionary-based. A sentiment reference dictionary that contains keywords phrases labelled by sentiment is used to classify the sentiment of the sentence/text. These scores require rules to ignore or account for sentences (or parts of a sentence) containing negations, dependent clauses or even sarcasm. The advantage of these methods is that they incur less computation overhead as there is no training needed. However, their drawback is that they often lack context and do not consider semantic relationships thereby resulting in low accuracy.

**2. Machine Learning (ML) based sentiment analysis** These methods use a machine

learning model to classify the sentiment based on words and their lexical ordering by using a labelled-training set (supervised approach). These methods are sensitive to the training data and need to account for class imbalance. This class imbalance can be dealt with by using ensemble methods such as random forests as highlighted in this paper. [18]. The advantage with them is that they be customised to the domain.

### 2.7.2 Considerations in scoring sentiment data

The scoring metric can make of use of techniques like Adverb scoring axioms such as adverbs of degree (AoD) where for example, adverbs such as 'extremely', 'absolutely' and 'hardly' indicate the strength of the sentiment or Adverb-Adjective combinations (AAC) scoring axioms such as Variable scoring, Adjective priority scoring (APS) as discussed in this paper. [17]

Generally, it can be seen that phrases separated by 'and' have the same polarity and those separated by 'but' have reverse polarity.

As mentioned before, it is essential to account for negation and modifiers when assigning the sentiment score, this can be done in an approach described in this paper [22] where the polarity of a 'sentiment word' can be flipped if it is negated. For instance, if 'good' has a polarity of $+1$ , then 'not good' will have negative polarity of -1. Similarly, when accounting for modifier, the strength of the sentiment is altered, so for example, 'extremely good' can have a polarity strength of $+2$.

Additionally, pronoun resolution can be incorporated as well to get more entity sentiment additional co-occurrence relationships between entity and sentiment compared to the original news article. There is also a need to consider a way to obtain co-reference sets that allow the resolution/aggregation of aliases of entities. E.g. Donald Trump and Donald J. Trump should refer to the same entity. [22]

Another potential consideration is the use of duplicate news articles having an effect on the sentiment score. For this reason, the model should eliminate redundancy of articles by not considering those duplicated articles. [22]

## 2.8 Visualising data

### 2.8.1 Data Preparation

First and foremost, it is important to understand the goal of the visualisation, what precisely is needed to be visualised and think about what type of data is required for this. This data (news articles) can then be processed to transform and summarise it, extracting key pieces of information that is determined to be essential to the application. This transformed data/ information can then be stored in our desired format and utilised to output the visualisation.

### 2.8.2 Principals of Visualisation

1. **Simple** It is important to ensure that the visuals are simple and intuitive. The information that is most relevant to the application must be clearly and succinctly visible and organised in a consistent manner. Adding any unnecessary information can make the visualisation convoluted and difficult to understand. [29]
2. **Standard** Standardisation of data structure and elements is needed for a good visualisation. This requires handling any complexities and discrepancies in the data as well as eliminating redundancies. Examples of this include, but are not limited to, using common abbreviations, identical scaling, consistent layouts across your data visualizations. [29] Additionally, it is important to provide context for these visualisations as well by using standardised labelling and indexing. [30]
3. **Scalable** Scalability, in this context, refers to the ability of a visualisation to adjust with the increasing volumes of data seamlessly. This increase should have minimal impact on the speed as well as the performance of the program. Additionally, this also relates how to fit the virtualisation by dynamically scaling it to the virtual space as it grows. [29]
4. **Identify target audience** For the visualisation to be a good representation of the data, it is important to identify the target audience and how they will interact/ use the visual. As mentioned before, exploiting visual details like size, colour, position, font etc can allow for a more intuitive design whilst directing the focus of the users to key bits of information. [30]
5. **Making use of interactivity** It can be useful to leverage the interactivity in the visualisation, thereby allowing for a multi-faceted visualisation based on the context that the users choose. For example, if a user wants to the see key news related to the airline industry, the visualisation can zoom in to focus on the part of the visualisation specific to that sector. User interactions should be intuitive and simple so as to not confuse the user and discourage participation.

# Chapter 3

# Ethical Considerations

This project mainly focuses on mining news articles to visualise what can be interpreted as the key news and entities in an industry. From an intelligence standpoint, this could be extremely useful as it condenses large amounts of news to give a visual overview of an industry. Though this project might highlight recent developments and trends in a sector, it is extremely unlikely to influence and propagate some radicalised opinion.

There is no direct human interaction in this project, neither does it require any collection or direct use of personal data. It should be mentioned that data will be used from large open-source knowledge graphs where there may be data regarding a person of interest that may be deemed personal. E.g. Person: Donald Trump, Location: White house. However, any data that is in public domain will not be considered private or sensitive. Wikidata releases all its data under Creative Commons CC0 (public domain). [35]

One important ethical consideration, however, is to think about any bias introduced in the model/ visualisation due to the nature of the input data which inherently might contain some underlying bias. For example, if the data used for the energy industry primarily contains a majority of right-wing articles talking about climate change being a hoax and refusing to support clean energy alternatives, then that will be reflected in the visualisation through sentiment and topics extracted. This may result in a negative sentiment associated with clean energy. In order to combat this to the best of my degree, I will need to use a variety of different sources that balance the bias. However, it is important to note that most news in its nature can be polarising and biased, so therefore it can be hard to gauge what quantifies as unbiased.

Another key ethical consideration is that of using software and data that may have copyright licensing implications. A key aspect of this project is to use news articles for entity, sentiment and topic extraction. The news articles can be obtained from various online sources of major news companies such as BBC, The Guardian etc. Using the articles (which are an Intellectual Property (IP) of these companies) for ”non-commercial” data analysis is not a copyright infringement as the copyright law has been updated to provide an exception for ”Text and data mining technologies to help researchers process large amounts of data” [36]. All other libraries and software packages used such as Dash, AllenNLP are open-source and free to use for the purposes of this project (at least for academic use).

# Chapter 4

# Project Structure

## 4.1 Preparing Data: DataLoader

For the scope of this project, the data acquired was in the form of text-only English news articles, pertaining to the airline industry. This data was provided by DeepSearch Labs as a collection of news articles scraped from Bloomberg news, containing information such as the url of news article, the date it was published, the headline (title), the author, a pre-processed category and the article itself. The data (originally) a csv file, was loaded as a pandas dataframe for easy management and manipulation. An example of the format of the data is shown in 4.1

| | url | date | title | author | category | article |
|---|---|---|---|---|---|---|
| 0 | https://www.bloomberg.com/news/articles/2021-0... | 2021-07-19 00:00:00+00:00 | Anger at Heathrow as Johnson's French U-Turn A... | ['Laura Wright', 'Christopher Jasper'] | Politics | London's Heathrow airport was thronged with tr... |
| 1 | https://www.bloomberg.com/news/articles/2021-0... | 2021-04-24 00:00:00+00:00 | World Pledges Aid for India as Cases Surge: Vi... | [] | prognosis | Healthcare workers administer doses of the Joh... |
| 2 | https://www.bloomberg.com/news/articles/2021-0... | 2021-07-09 00:00:00+00:00 | Want to End Flying Shame? Meet Sustainable Jet... | ['Jack Wittels'] | QuickTake | Workers fill an Airbus A350 passenger plane wi... |
| 3 | https://www.bloomberg.com/news/articles/2021-0... | 2021-07-14 00:00:00+00:00 | Missouri County Sounds Alarm; Tokyo Cases Surg... | [] | prognosis | Health officials in southwestern Missouri aske... |
| 4 | https://www.bloomberg.com/news/articles/2021-0... | 2021-06-02 00:00:00+00:00 | Belarus Accused of Letting Illegal Migrants Cr... | ['Milda Seputyte'] | Politics | Alexander Lukashenko on May 28. Lithuania accu... |

**Figure 4.1:** Example data format

The objective was to analyse the semantic information in the news articles for each category within a specific time interval. This prompted the need to divide the data meaningfully and was done by grouping the articles by time intervals of a year using 'date published' column as well the existing categories that stayed constant to act as a control scaffold. Therefore, an input data group consisted of the articles in a specific category (e.g. Business) during a specific time interval (e.g. 2021). The motivation behind this was to see how news in certain categories changes over time in terms of topics extracted as well as information derived from semantic triples within these topics.

Once these '(Year, Category)' groups were obtained, they were filtered based on their size (i.e., the number of member articles) by omitting all those whose size was less than the $|mean - standard\ deviation|$ of the size of all groups. The reason for using the absolute difference between the mean and deviations of the value counts of the groups was because

variance in the value counts of these groups was too high. Therefore, the aim was to retain the larger year-category groups but omit the really small ones. This ensured that the input data was of a significant size in order to yield relevant results before any further processing was done.

## 4.2 System Design

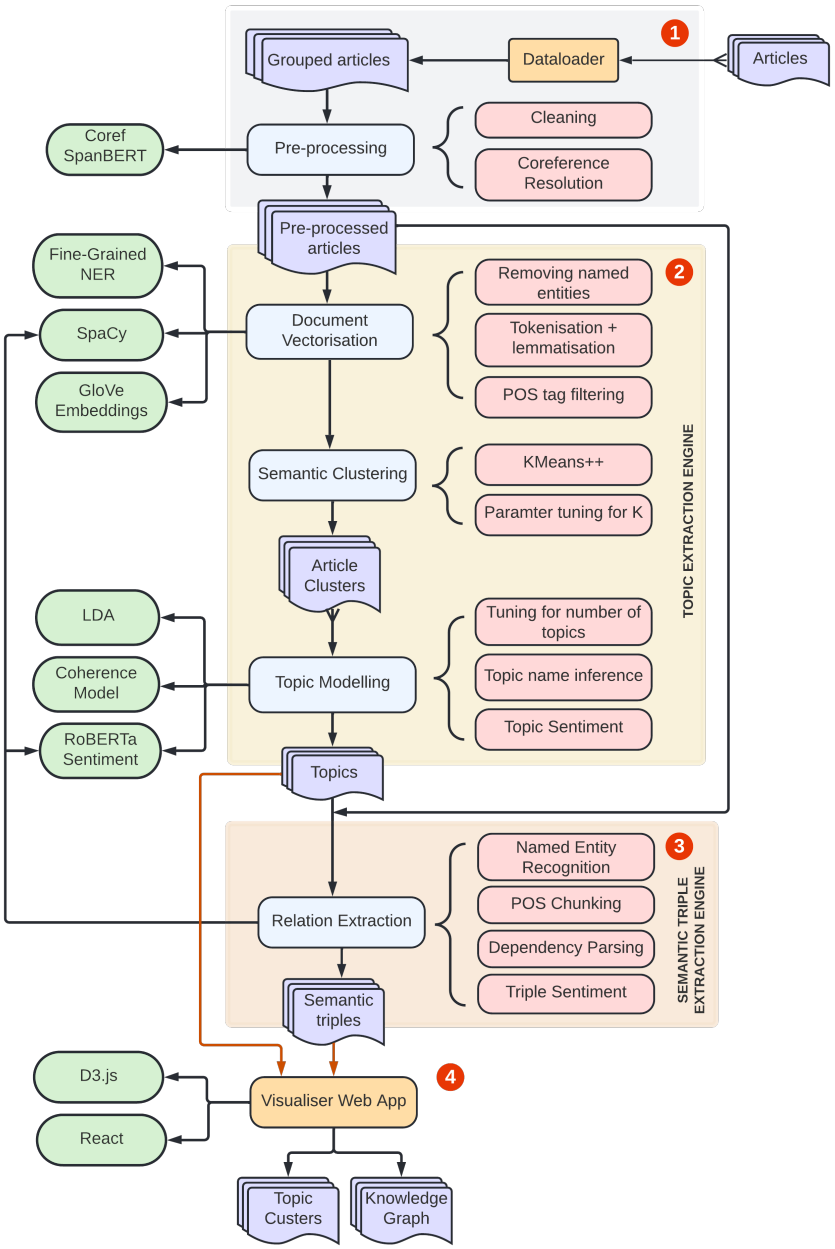


**Figure 4.2:** System Architecture Diagram

At a high level, the tool's main goal is to perform semantic analysis on the news corpora to extract information such as topics and semantic triples from the articles. Figure 4.2, shows a high level architecture diagram which follows the pipeline of the semantic analyser tool which follows 4 key stages:

1. **Data preparation:** The year-category data groups are generated from the dataset by the `dataloader` as discussed in 4.1. Each input data group then follows the pipeline of pre-processing → topic extraction → semantic triple extraction engine.
2. **Topic Extraction Engine:** This involves semantic clustering of the articles, from which latent topics are extracted. These topics undergo processing such as topic name inference, keyword extraction and sentiment analysis to obtain semantic information about these topics.
3. **Semantic Triple Extraction Engine:** For each of the semantic clusters, the semantic triple extraction engine extracts triples of type subject-predicate-object about entities in the article corpus.
4. **Visualisation App:** Finally, the semantic clustering and latent topic models are visualised in circle packing graph and the semantic triples are visualised using a force directed graph using D3.js.

## 4.3 Technologies used

### 4.3.1 Libraries

**SpaCy** is an open-source NLP library "designed specifically for production use" [13]. It uses state-of-the-art pre-trained models and in-built pipelines for data processing techniques such as tokenisation, dependency parsing and POS tagging etc. The motivation for using this over other alternatives such as NLTK, was that it was much faster than NLTK for word tokenisation and POS-tagging. Additionally, the NLTK API is quite primitive in comparison to the object-oriented spaCy and requires a lot of unnecessary string manipulation [14]. The spaCy pipeline is trained on several models. For this project, the pre-trained English en-core-web-lg model was used with a large word vector table with approximately 500,000 entries.

**Gensim** is an NLP open-source library that makes use of state-of-the-art models for word vectorisation, text similarity and topic modelling. For this project, this library was used for the pre-trained models it exposes, in particular, `Word2Vec` and `GloVe` and for topic modelling for which the Latent Dirichlet allocation (LDA).

**AllenNLP** is an open-source NLP library built on PyTorch and offers variety of well-engineered existing state-of-the-art model implementations [15]. Additionally, it is well integrated with components from the spaCy library such as the `Tokeniser` and `SentenceSplitter`, making it a fitting choice for this project.

**D3.js** is a JavaScript library which is commonly used for building data visualisations in the

web browser. This was used in conjunction with React to build the web application for the visualisation tool. The motivation for using this library over several other alternatives was its data-driven approach to DOM manipulation which enables building customisable interactive visualisation frameworks.

### 4.3.2 Models

Given the extensive prior research in literary analysis and natural language processing, the decision was made to use some of the state-of-art pre-trained models which have proven to be useful in other domains for this problem. The models used in different parts of the project are as follows:

**SpanBERT for Coreference Resolution** is a state-of-the-art model used for co-reference resolution developed by the Allen Institute of Technology. It uses the SpanBERT embeddings to get ”high-order coarse-to-fine” predictions of spans of text [12]. The motivation for using this over its predecessor BERT was that SpanBERT outperforms it for the coreference resolution task as it masks random contiguous spans of text unlike BERT which masks random tokens in a sequence, resulting in improved span predictions. The model scores these predictions to get coreference clusters which are applied to get the resolved text [12].

**RoBERTa Stanford Sentiment Treebank** is a binary classifier trained on RoBERTa large for Stanford Sentiment Treebank dataset, on which it achieved 95.11% accuracy [16]. RoBERTa is another variant of BERT and stands for a Robustly Optimised BERT approach. Where BERT is optimised for Masked Language Model task (MLM) and Next Sentence Prediction (NSP), RoBERTa forgoes the latter and only minimises the loss for MLM. Additionally, it uses dynamic masking (i.e., different parts of the sentence are masked for each epoch) instead of static masking (i.e., the same parts of the sentence are masked each epoch) [16].

**Fine-Grained NER** is a state of the art named entity recognition model from AllenNLP and is a re-implementation of the Lample (2016) [17] model. It makes use of bi-direction LTSM with a Conditional Random Field layer (CRF) and uses two types of embeddings: character embeddings and ELMo embeddings (refer to Sections 2.5.1 and 2.4.2). In Lample (2016), a comparison study for the performance of the model for English NER on CoNLL-2003 test set highlights that it outperforms several other models giving an F1 score of 90.94% [17]. The motivation for using this model was this high accuracy and that the model identifies a broad range of 16 semantic types.

### 4.3.3 Web deployment

A web interface was used for displaying the results of the semantic analysis tool as it allowed for more succinct and cohesive visualisations and enabled user testing. This was accomplished by using React and D3.js to build stateful visualisations. In the latter stages of the project, the website was hosted on Heroku [18] which was a scalable approach to accommodate user testing and aided in the evaluation discussed in Section 7.4.

# Chapter 5

# Topic Extraction Engine

The semantic analysis tool focused on 2 main text mining components in order to extract information from the news articles. This section focuses on the first of these components: the topic extraction engine, covering different data processing techniques, word and document approaches, semantic clustering and topic modelling using LDA.

## 5.1 Data Processing

The prepared data (from Section 4.1) grouped by (Year, Category) comprised of a list of articles that were published in that year and assigned that category. Processing all the text for every single article would add an intensive computational burden without a huge amount of added benefit, with a potential risk of cluttering the model. For the sake of brevity, the aim was to use the article data most representative of the article. To facilitate this, a decision was made to use the titles and introduction of the articles as they contain the core information in the articles. The introduction of the article ('intro') was extracted as the first 6 sentences of the article using SpaCy's `SentenceSplitter`. Throughout the course of this report, 'intro' and 'article' will be used inter-changeably to represent a single news article in the corpus.

Once the article introductions are extracted, they then undergo pre-processing before they can be vectorised for clustering. This section highlights the different pre-processing steps and key decisions made to convert the article intros into a list of tokens for the vectorisation step.

### 5.1.1 Coreference Resolution

The first step involves coreference resolution of the articles ('intros'). This involves using the pre-trained AllenNLP SpanBERT coreference resolution model for finding the set of mentions in the text that refer to the same 'entity' (i.e. coreference clusters). This step was performed on the complete article intro in order for the mentions to be resolved throughout the entirety of the intro and not just on a sentence level. For example, for the

text 'Sean Doyle is the CEO of British Airways. Prior to this, he was the CEO of Air Lingus', if the coreference resolution was done on the sentence level, the model would miss out on resolving 'he' in the second sentence to 'Sean Doyle'.

### 5.1.2 Cleaning Data

Post coreference resolution of the article intros, the data needed to be 'cleaned' in order to ensure that the news articles retrieved were informative and useful. This involved the following steps:

1. Removing any unnecessary characters e.g., trailing ellipsis, parenthesis etc.
2. Removing stopwords. The spaCy model's default stopwords list was used. This significantly reduced the amount of text per article intro by removing words such as this, 'that', 'there', 'where', 'are' etc. This alone, however, was not sufficient. Given the nature of online news articles, there were some prominent phrases that showed up in several articles, e.g., "Read here", "For more information", "Sourced by" etc. These were also removed as they do not provide any relevant information specific to the article, thereby reducing redundancy.

### 5.1.3 Removing Named Entities

As mentioned previously, the aim of these pre-processing steps was to obtain a `filtered_tokens` list for each article intro which would be vectorised to represent an article as a vector for clustering. One of the key decisions made was to remove all corresponding named entities extracted using the fine-grained Named Entity Recognition model from each article intro. This was done in order to remove any dependency of the topics on the named entities in the articles for clustering and topic modelling. Eliminating this dependency results in better silhouette scores and coherence scores for semantic clustering and topic extraction as detailed in Section 7.1.1 and 7.1.4 respectively. Additionally, this step was done before tokenisation as named entities can often be a collection of words such as "British Airways", "Paul Sweeney", "New Haven" etc. and removing these post-tokenisation would not be ideal as they would lose their semantic meaning as tokens. For instance, post tokenisation, the model would try to remove "New", 'Haven" from the article intro. Therefore, regex patterns were used to remove all the occurrences of the extracted entities in an article. This included removing all entity types which included but were not limited to, 'PER', 'LOC', 'DATE', 'ORG'. This also had a performance gain over the post-tokenisation approach as it avoided the lookup for each token against the unwanted entity tokens list.

### 5.1.4 Filtering Tokenised Data

Once the data was cleaned and void of the pre-determined entity types, the article intros were tokenised and normalised. This was done using the SpaCy library's `Tokeniser` and `Lemmatiser` pre-trained on the English language model: en_core_web_lg (refer to Section

4.3.1). Normalisation was done through lemmatisation (to obtain the base forms of the tokens). This was chosen instead of stemming as it guaranteed more semantically meaningful tokens which for reasons detailed in Section 2.1.3. The process of obtaining a list of tokens for each article intro was as follows:

1. Each sentence in the intro was turned into a list of tokens.
2. These tokens were filtered by their Part-Of-Speech (POS) tags against the allowed POS tags, which was restricted to nouns ('NOUN').
3. Tokens that were numeric and less than 2 characters were also removed.
4. The remaining tokens were lemmatised and added to the `filtered_tokens` list for the given article.

It is important to note that the allowed POS tags were originally set to include nouns (NOUN), adjectives (ADJ), verbs (VERB) and adverbs (ADV) . This resulted in a lot of unnecessary tokens such as 'because', 'did', 'very' etc. Different combinations of the allowed POS tags were experimented with, ultimately allowing only nouns to comprise the `filtered_tokens` for each article ('intro'). This decision was primarily based on the resulting clustering scores when the allowed POS tags were limited to 'NOUN' and when they included 'NOUN', 'ADJ', 'VERB' and 'ADV' as discussed in the evaluation in Section 7.1.2.

## 5.2 Generating Document Vectors

Once the corresponding `filtered_tokens` list for each article intro is obtained, the individual tokens are vectorised so they can be used to generate the document vectors.

### 5.2.1 Word embedding approaches

To vectorise the tokens, different pre-trained word embedding models were used. These approaches (in chronological order) with their strengths and shortcomings are outlined below.

TF-IDF

Term frequency-inverse document frequency is a statistical metric that examines the comparative relevance of a word/token with respect to an article intro in the collection of articles. Representing the tokens as their TF-IDF measure first involves calculating the term frequency (TF) using the bag of words (BoW) approach from Gensim `Doc2BoW`. To account for overlapping terms across all article intros, the term frequency was multiplied by the inverse document frequency to weigh down the terms whose high frequency is not unique to a document (Refer to Add in BCKG). A dense document-term using the TF-IDF values for each term is used to represent the article intro as a vector for clustering.

**Word2Vec** The next approach saw the use of Google's `Word2Vec` model (from Gensim) to obtain pre-trained word embeddings for each token in an article's corresponding `filtered_tokens` list. The issue with the previous approach was that it represented the statistical information about a document, in particular, the measure of the frequency of words in a document, rather than any semantic information about a document. `Word2Vec` embeddings, on the other hand, return a vector for each word that accounts for semantic similarity (based on cosine distance) between the words represented by the vectors [19].

**GloVe** The final approach was to use Stanford's `GloVe` (global vector) embeddings from Gensim. `GloVe` as been pre-trained on Wikipedia and Gigaword 5, comprising of a vocabulary of 400,000 words which are represented by 300 dimensional vectors [20]. The advantage of this approach over previous approaches is that it makes use of global statistics such as word co-occurrences to obtain word vectors. Using these embeddings also saw an improvement in the silhouette scores when deriving the semantic clustering of articles as detailed in Section 7.1.3.

### 5.2.2 Document embeddings

In Section 5.1.4, `filtered_tokens` are defined as the all *relevant* tokens for each article intro. As mentioned above, the filtered tokens were vectorised using the `GloVe` embeddings [20] resulting in a list of vectors corresponding to the `filtered_tokens`. In order to get the document vectors from these word vectors, two approaches were tested:

**Approach 1** The corresponding tokens in `filtered_tokens` list for the article intro are vectorised (using `GloVe` embeddings) and averaged to find the corresponding article vector. This was a simple approach that gave equal importance to each token associated with the article intro.

**Approach 2** Rather than simply averaging the token vectors for each article intro, the document vector was calculated using a weighted sum of corresponding word vectors (from `filtered_tokens`). This was done by using a TF-TDF model to get the term-frequency inverse document frequency weight for each token in the corresponding article. This meant that words that appeared in high frequency unique to the associated article contributed more to the document vector than those with a lower frequency or those that appeared frequently in other article intros.

7: Add the document graph

## 5.3 Semantic Clustering

Once the article intro vectors are calculated, they are processed so they can be used for semantic clustering.

### 5.3.1 Dimensionality Reduction

#### Normalising Data

The first step is to normalise the article vectors in order to eliminate redundancy in data and standardise the features (components of the vector) in case of high variance, which contributes to better clustering. Additionally, when it comes to performing principal component analysis for the article vector, we want the features (in vectors) to be independent of their standard deviation or variance. This is necessary to get a good covariance matrix among the features.

#### Principal Component Analysis

As established previously (refer to Section 5.2.1), `GloVe` document embeddings result in a 300-dimensional vector. Clustering the document vectors becomes inefficient and meaningless at these high dimensions as the concept of distance becomes less precise as the number of dimensions increases [21] as the volume of space increases exponentially making the available data spare - curse of dimensionality. For any given point in this high n-dimensional space, the difference in 'distance' (euclidean) to the closest point $d_min$ and the farthest point $d_max$ with respect to the $d_min$ becomes negligible [22].

$$\lim_{n\to\infty} \frac{d_{max} - d_{min}}{d_{min}} \to 0$$

The concept of clustering these articles relies on grouping similar articles together based on their attributes. The similarity is determined based on the vectorised `filtered_tokens`. However, given the high dimensional nature, some of these attributes will not be relevant in determining these clusters. The goal is to find the attributes (vector components) with the most variance across the documents vectors to represent the features. This is accomplished by performing principal component analysis (PCA) on the normalised high-dimensional input vector space to map to a low dimensional space, whilst minimising information loss [21].

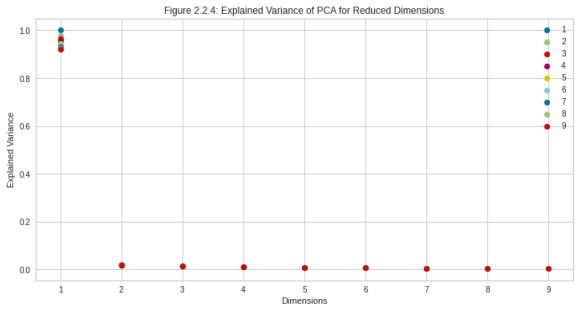


FIGURE 5.1 Explained Variances for PCA for dimensions=1..9

TODO: sil scores with different pca? or explained variances?

### 5.3.2 KMeans Clustering

Once the normalised data is mapped to the lower subspace, the documents (article intros) are clustered based on similarity. The aim is to cluster the article intro vectors based on their cosine similarity (Eq. 5.2) using cosine distance as the distance function (Eq. 5.3). The clustering technique used was KMeans clustering. This was done using the `sklearn` library's `kmeans` function with 'euclidean distance' (Eq. 5.4). This was feasible given the linear relationship between Euclidean distance and cosine distance [23] for normalised vectors as in shown in Eq. 5.6.

$$\textit{For normalised vectors, } x, y : \sum x_i^2 = 1, \sum y_i^2 = 1 \tag{5.1}$$

$$\begin{aligned} \textit{Cosine Similarity, } cos(x,y) &= \frac{\sum x_i y_i}{\sqrt{\sum x_i^2 y_i^2}} \\ &= \sum x_i y_i \end{aligned} \tag{5.2}$$

$$\textit{Cosine Distance} = 1 - cos(x,y) \tag{5.3}$$

$$\begin{aligned} \textit{Euclidean Distance, } ||x-y||_2^2 &= \sum (x_i - y_i)^2 \\ &= \sum (x_i^2 + y_i^2 - 2x_i y_i) \\ &= \sum x_i^2 + \sum y_i^2 - 2\sum x_i y_i \\ &= 1 + 1 - 2cos(x,y) \\ &= 2(1 - cos(x,y)) \end{aligned} \tag{5.4}$$

$$= 2(\textit{Cosine Distance}) \tag{5.5}$$

$$\tag{5.6}$$

Furthermore, to avoid falling into the trap of random centroids initialization, which is a major shortcoming of the KMeans method, the clustering was done with KMeans++. The reason for doing so was that KMeans++ offers a better initialisation approach for centroids, in which the first one is picked at random and the subsequent centroids are

chosen with a probability proportional to the squared distance from the closest chosen centroid.

## Finding the optimal number of clusters: k

Determining the number of clusters (k) in KMeans is a crucial factor to consider. This was done by computing the KMeans clustering algorithm for different values of k varying from 2 (minimum number of clusters) to m (maximum number of clusters), and picking the optimal k based on a comparing statistic. The two comparison statistics considered were: Elbow method with Wihtin Cluster Sum of Square distance (WCSS) and silhouette score.

### Method 1: Elbow method

The elbow method uses the distance (Euclidean) between the cluster centroid and its members, i.e., intra-cluster variance (distortion score or WCSS) to determine how many clusters are needed to encapsulate the variance of the data. In particular, it minimises the loss function: WCSS (a.k.a. distortion score) which is the sum of the squared distance between each point and the centroid in a cluster [24].

$$WCSS = \sum_{C_k}^{C_n} ( \sum_{d_i \in C_i}^{d_m} ||d_i - C_k||_2^2) \tag{5.7}$$

As seen in Figure 5.2, the elbow (bend) in the plot determines the optimal number of clusters, i.e., the k value = 5. After this value, distortion score gives diminishing returns.

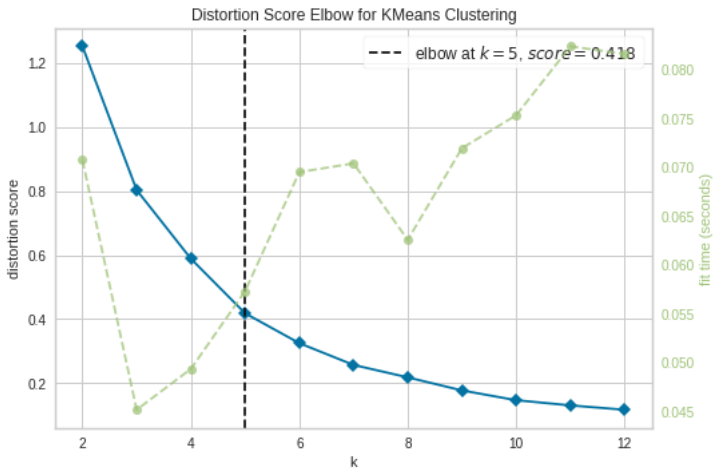


FIGURE 5.2 Elbow score: Elbow Method Plot for ('Business', 2020) for k=2 to 12

### Method 2: Silhouette score

The second approach involved using the silhouette score. Unlike, the elbow method silhouette score combines both separation and cohesion [24]. This means it means how close

a point is within its cluster (cohesion) and other clusters (separation) [25].

$$Silhouette\ Score = mean_i(\frac{S_i - C_i}{\max(S_i, C_i)})$$

where cohesion, $C_i$ = average distance between i and all points within its cluster. separation, $S_i$ = average distance between i and all points not in its cluster.

The silhouette score is the mean of Sil(i) for all points i in the data. It ranges from -1 to 1. A value closer to 1 indicates that clusters are well separated and distinguished. A value very close to 0 indicates there is significant overlapping between the clusters, and the clusters are not distinguished. A value near -1 indicates that clusters are assigned incorrectly.

Out of these two approaches, I decided to go with the silhouette score as it gives a better value of the goodness of a clustering. It factors both how compact a cluster is and how distinct it is. Additionally, it is better suited for large volumes of data and given that in the context of the problem, the aim is to cluster the news articles into distinct clusters with little overlap, as o each of these clusters, topic modelling will be performed to extract topics and ideally, there should very few common topics among the clusters.

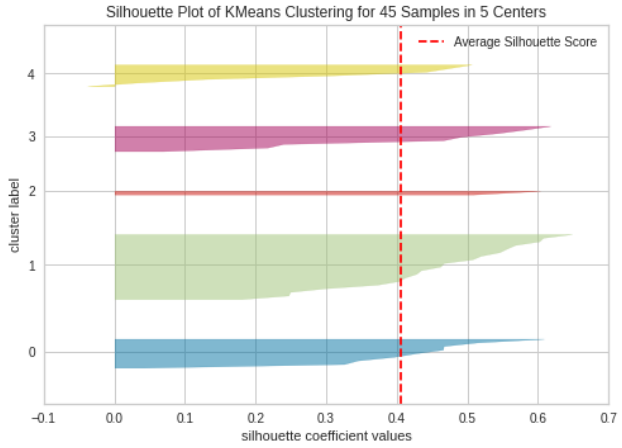


FIGURE 5.3 Silhouette score = 0.4059555 for Kmeans clustering k=5, for ('Business', 2020)

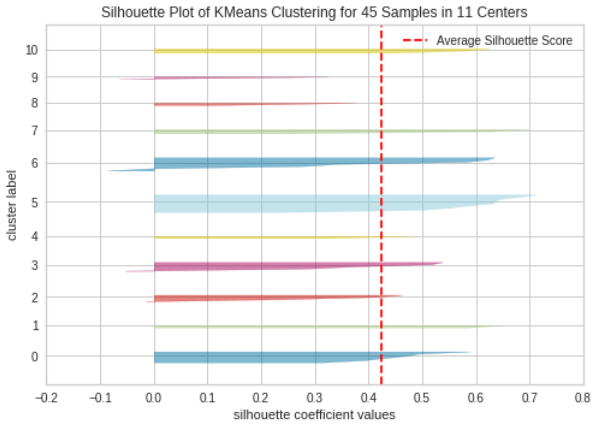


FIGURE 5.4 Silhouette score = 0.4229686 for Kmeans clustering k=11, for ('Business', 2020)

We can see in the plots generated by YellowBricks machine learning visualization library in Figure ???) that the optimal cluster number, k = 5 (from Figure ???) chosen by the Elbow Method is not the same as the one chosen by the silhouette method k = 11.

8: Add explanation

### 5.3.3 Find n most representative docs

Once the article intros are clustered, we want to obtain a cluster to article mapping. However, with volumes of data, it may not be as unnecessary to consider every single

article in the cluster particularly those that are not close to their respective cluster centroids. More importantly, since the next step is topic modelling on these clusters of article intros, we do not want to consider those clusters that have very few members as they will not spawn any good topics. In order to facilitate this, the idea of using the n most representative articles comes to play.

A pre-determined value n is selected and for each cluster, the euclidean distance of each member (article intro) in the cluster is compared with its centroid. The articles are sorted in decreasing order of distance and the top n articles are selected.

The question to consider then is how to decide this value of n. This value of n is calculated based on the clustering. First, the range of the size of clusters is calculated with the minimum number of data points (i.e, article intros) in a cluster subtracted from the maximum number of data points in a cluster. Then, the average number of data points in clusters is calculated. The minimum of the these two values determines n. This is done so that sparse clusters are omitted from the input data for topic modelling. This gives a final mapping of cluster to article intros (pruned).

9: update this!! Now uses std and mean

10: Maybe add formula for n most repr

11: talk about diff docuemnt vectors graoh

## 5.4 Topic Modelling

Once the clusters of articles are obtained, the aim is to extract the topics associated with each cluster. This was done through topic modelling, which follows an unsupervised approach of extracting the top topics from a corpus. In particular, the model used was the Latent Dirichlet allocation (LDA) from the Gensim library.

As stated in section **??? (refer to background section)**, each document (in this case, article intro) is made of up words (in this case, filtered tokens) and each topic has several (key)words associated with it. Given that the LDA model is a probabilistic model, it tries to estimate the topic distribution for each article (intro) and the word (data words) distribution for each topic. The motivation of using LDA, in this case, is to get the topic-document probability distribution which can then be used to get the topics associated with a document. For the scope of this problem, this was limited to one dominant topic.

12: Why use semantic clustering and topic modelling —¿ topic modelling LDA does not rely on semantic information and so we obtain semantic clusters first.

### 5.4.1 Key decisions

#### Noun-only corpus

From the pre-processing step, the article intros have already been cleaned, lemmatised, filtered (remove stopwords) and tokenised to get their corresponding 'filtered_tokens' (which were used to obtain the article vector used in clustering). In particular, the POS filtering step (section??), resulted in only 'NOUN' 'filtered_tokens'. This promoted the question of whether the corpus should be restricted to nouns. Based on previous studies [26] [27], and the results from Eval coh scores??? which saw improved coherence values of topics and fewer numbers of 'unpopular' topics (i.e. few associated articles), the corpus was limited to nouns. This aligned with the intuition that nouns are better indicators of a topic as they provide more semantic meaning about a given article. Since LDA treats all words in the corpus with equal importance, opening the corpus to other POS categories, e.g. ADJ, VERB etc. will result in sub-optimal topics as for a given news article, "fast" and "pandemic" will have equal weight.

13: eval table coherence scores

14: APPENDIX POS tages,ENT TYPES

#### Using N-grams: Bigrams

Inspired by the work: 'Beyond bag of words' [28], a decision made was to augment the corpus by using n-grams, in particular, bigrams, rather than solely using unigrams. This allowed the model to consider a combination (phrases) of two words occurring across documents such as "coronavirus_pandemic", "aviation_industry" etc. This was particularly relevant given the dataset of news articles which contain noun chunks as subjects which can be used to infer topics.

#### Removing Named Entities

Another crucial decision was to remove any names entities of type, including but not limited to, GPE, LOC, and ORG from the corpus. This was also satisfied by the 'filtered_tokens' (from pre-processing in ????) where the entities found using the Fine-Grained NER model were removed. As seen in Figure ???, removing these entities also improved the semantic coherence of the topics.

**BoW vs TF-IDF**

In LDA, the documents (intros) need to be transformed into numeric feature vectors which form the corpus. Since it does not rely on semantic information of the document and builds topic estimators based on words, documents and topics, it uses frequency vectors as obtained by Bag of Words and Term Frequency - Inverse Document Frequency. The decision was made to use TF-IDF over the more common Bag-of-Words [29], as since the input data group passed from the dataloader is categorised, there are bound to be words that have high Document Frequency (i.e., occur frequently in all documents). Since we want assign article intro to a dominant, we want minimal overlap of topics.

### 5.4.2 Implementation: LDA

1. **Bigrams:** Using Gensim's Phraser model which uses colocation detection, the bigrams are generated (a_b where a and b are nouns), which are appended to the 'filtered_tokens'.
2. **Corpus:** Based on the decisions highlighted in section ???, the corpus is defined as a collection of filtered, tokenised, lemmatised nouns and noun-chunks (bigrams) of article intros, barring entities and stopwords.
3. **Dictionary:** A dictionary mapping the words in the corpus to their integer ids is created using the Dictionary wrapper? in Gensim.
4. **BoW, TF-IDF:** The corpus is passed to the bag of words model from Gensim and then is transformed by the TF-IDF to return a weighted frequency of words in article intros. This is the input to the LDA model.

### 5.4.3 Parameter tuning

15: Talk min prob and other param???

A prime contributor in determining the quality of the topics extracted is the number of topics parameter, n_topics, passed into the LDA model. LDA outputs as many topics as are defined by K: a low K results in too few or very broad topics, whereas a high K results in uninterpretable topics or topics that ideally should have been merged. Choosing the right value of K is thus an important task in topic modeling algorithms, including LDA. [30] This is assessed by calculating the coherence of learned topics by using the CoherenceModel class from the Gensim library. The optimal number of topics (i.e., value resulting in the highest coherence score) will be dependant on on the input data. Therefore, for each of input groups (passed from the loader), we compute this optimal number 'optimal_n_topics' by running the LDA model with different values of k varying from min_topics(=2) to max_topics, where max_topics is the maximum number of topics per (clustered) set of article intros, and the LDA model with the highest resulting coherence is chosen. The max_topics value is set as the size of the cluster (i.e., number of article intros within cluster) integer divided by min_topics (defaulted to 2). The motivation for this was

to bound the max_topics so that each topics would have at least the minimum number of article intros (=2). Running the LDA this many times, also added a performance cost so pruning the number of LDA runs was necessary.

16: Either here or somewhere else explain coherence?

17: Make a coherence plot

### 5.4.4 Dominant Topic-Article Mapping

Once the optimal value of n_topics is determined, the corresponding LDA model is applied to the TF-IDF corpus to get the resulting article intro-topic distribution. This returns the most probabilistic topics for each article intro. This is of the form articleId: (topicId, Probability) where Probability refers to the likelihood that the arcticle (intro) belongs to the topic corresponding to the topicId. Of these, the most dominant topic (highest likelihood of belonging to the topic) is selected, resulting in an 1-1 topic-document mapping.

### 5.4.5 Topic Name Inference

The LDA model returns a set of keywords associated with each topic. An augmented feature of the Topic modelling engine was to infer the topic names from the keywords of the dominant topic. This involved using the top 5 keywords (including splitting the bigrams) associated with each topic, converting these to word vectors using the GloVe embeddings and calculating the **top N** most similar words vector to the list of keyword vectors. This is done by using the glove_model.most_similar() method from Gensim. It calculates the cosine similarity between an average of projection of the word vectors, and takes a list of positive words (=keywords) which positively contribute to the similarity and negative words which contribute negatively, returning a list of potential topic names. These contenders are checked for validity in terms of whether they are a noun and are not augmented stopwords. The first (most similar) candidate that meets the validity checks is then inferred as the topic name for the given topic.

18: Change TOP 5 based on implementation

### 5.4.6 Topic Sentiment

The last component of the the topic modelling engine involves sentiment analysis. This requires two pre-requisites. The cluster to article intros mapping obtained from Section 5.3.3, and the sentiment associated with each article intro. For the latter, the RoBERTa Sentiment Treebank model is used. This outputs a binary label where 0 indicates negative sentiment and 1 indicates positive sentiment. For each article (number) associated with a given topic, the sentiment of the coref-resolved article intro is calculated (not the associated tokens as sentiment model assigns sentiment based the semantic meaning and

filtering words will result in loss of meaning) . These are then averaged to get the average sentiment of the topic. Based on the range of this value it is assigned one of the following: 'Positive', 'Negative' and 'Neutral' as seen in Figure ????.

```
def get_sentiment(a):
  # label: 1 -> positive, 0->  negatice
  if a >= 0.4 and a <= 0.6:
    return "Neutral"
  return "Positive" if a > 0.6 else "Negative"
```

19: Change imp to use probs instead of labels

20: Fix highlighted.

# Chapter 6

# Relation Extraction Engine

This section focuses on the relation extraction engine, covering the journey to the current implementation, focusing on data preparation and the different approaches for extracting semantic triples.

## 6.1 Data Pre-Processing

Before the relations are extracted from the articles, the articles need to be processed. This involves the coreference resolution and data cleaning to remove stopwords, unnecessary punctuation etc. as discussed in previous sections 5.1.1 and 5.1.2 respectively. Pre-processing the articles with coreference resolution has the benefit that all the .....

21: add stuff

## 6.2 Motivation

The motivation behind this part of the project was to find a way to extract meaningful information from the article, in particular, information centred around key entities in the article corpus for a certain topic. This subsection covers the two different ideas explored for extracting information from articles.

### 6.2.1 Cooccurence Graph

One of the earlier approaches was to derive a co-occurrence graph from the article corpus (pre-processed with coreference resolution and data cleaning) for a selected topic. This was useful in showing what entities were present and how they connected to one another as shown in Figure 8.2. In order to derive the cooccurence graph, Named Entity Recognition (NER) was used. For each article, the intro sentences were extracted and on each sentence NER was performed to extract all relevant entities. The types of entities

extracted included but were not limited to, 'GPE', 'PER', 'LOC', 'ORG' etc. and excluded 'DATE', 'MONEY', 'TIME', 'CARDINAL', 'PERCENT' and 'QUANTITY' (See Appendix for reference). As seen in Figure 8.2 in Appendix, the the size of the nodes representing the entities were reflective of the number of co-occurrence links (edges) incident to and from other entities. This was done so that for any given topic the most relevant entities, i.e., mentioned the most in the group of articles representing a topic, was apparent. The **limitation** of this approach, was that while this was a good indicator of general entities talked about in a topic, the amount of information extracted from the topic article corpus was not sufficient and did not provide any explicit information about *'how'* the entities were related to one another, but rather, just simply, that they were.

### 6.2.2 Semantic Triples

In an attempt to extract more 'relevant' information indicative to the content discussed in articles of a given topic, the decision was made to extract semantic triples from article corpus instead of building a cooccurrence graph. This involved finding subject-relation-object triples form the sentences in the article corpora. ??????? better semantic information extraction

22: Talk about why the pure dependency parsing method did not go well

The process of extracting these relations involved inferring both syntactic and semantic dependencies between tokens in a sentence, making use tokenisation, dependency parsing, part-of-speech tagging and named entity recognition.

## 6.3 Key decisions

In order to capture better nodes and relations, phrases are used instead of words. Given an example of teh sentence in figure, are calling for gives better understanding than calling. Also talk about how the knowledge graph is the goal and in particular we want to see the subject as entities to find out all the information about them in articles in a topic.

### 6.3.1 Relation as verb phrases

### 6.3.2 Subject as entities

Given that the end goal is to build a knowledge graph, the nodes must have common entities so that the graph can display all the different triples about an entity and which entities exist in the article corpus of a given topic. Give an example. For example, having two triples one with subject British Airways and other will British Airways spokesperson is not ideal as both triples provide information about the entity British Airways, however, they will be displayed as discrete triples. Instead, splitting the subject phrase as

### 6.3.3 Object as noun phrases

## 6.4 Implementation

The semantic triples need to be of type subject-relation-object triples. The section highlights the how the subject, relation, object phrases were extracted from the articles in order to build a knowledge from these triples.

### 6.4.1 Extracting Verb Phrases

As mentioned in Section 6.3, the relation phrases would be extracted as verb phrases. First and foremost, the coreference resolved and cleaned articles are split into sentences and the verb phrases are extracted from each sentence. This is done by regex matching of part-of-speech tags as seen below.

```
verb_patterns = [[{'POS':'AUX', 'OP': '?'}, {"POS":"VERB"}, {"POS":"ADP"}],
          [{'POS': 'VERB', 'OP': '?'}, {'POS': 'ADV', 'OP': '*'},
           {'POS': 'VERB', 'OP': '+'}],
           [{'POS': 'VERB', 'OP': '+'}] ,
           [{'POS': 'AUX', 'OP': '?'}, {'POS': 'PART', 'OP': '?'},
           {'POS': 'VERB', 'OP': '+'}]]
```

23: Talk about why these particular regex -how extracted pos- give examples

Once the relevant verb phrases in a sentence are extracted, they are filtered to obtain the phrases where the root of the sentence is present. This is done by using dependency parsing through the SpaCy library. Generally, in dependency-based grammars, all words or tokens, in a sentence, barring one, are dependant on other words. This is the root of the sentence. Most commonly, this word a verb. Therefore, since the scope of relations extracted focuses on the predicate being a verb, we derive the root verb from the dependency tree and filter to get all the verb phrases that contain the root verb. Given the regex pattern matching criteria discussed above, we could have multiple root verb phrases, for example, one with just the root verb (e.g 'calling'), one with the auxiliary verb (e.g. 'are'), root verb (e.g. 'calling') and adposition (e.g. 'for') as shown in Figure 6.1. In order to extract the most informative coherent relations, the longest verb phrase is selected.

24: reference of root being a verb

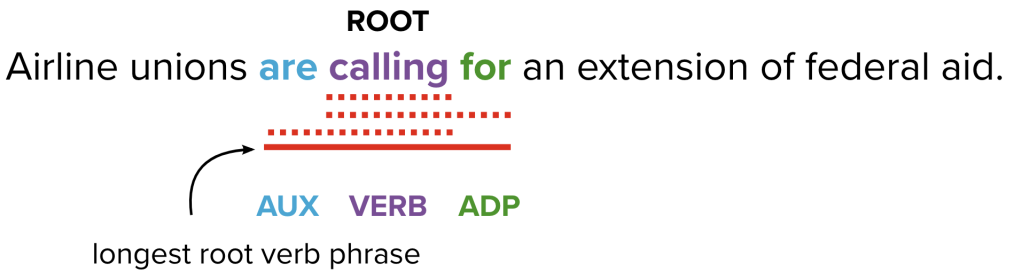


FIGURE 6.1 Verb phrases extracted from an example sentence from article corpora

### 6.4.2 Extracting Noun Phrases

After the extracting the verb phrases for relations, the same is done for noun phrases in the sentences. As discussed in key decisions in order to capture meaningful relation, the model focuses on extracting subjects and objects as nouns. The most common sentence structure follows the dependency of subject-relation-object in that order. Therefore, for the scope of this project, the focus is directed to these types of semantic relations. First and foremost, for each sentence, the noun phrases are extracted in that sentence which uses chunking and POS-tagging exposed by SpaCy. These return a list of `Spans`, which are essentially slices of the sentence and contain information such as start, end position are of the phrase (slice) in teh sentence. Additionally, for each sentence all the relevant named entities are also extracted using the Fine-Grained NER model. These entities allowed all types such as 'GPE,' 'PER', 'LOC', 'ORG', 'NORP' etc. except for 'DATE', TIME', 'CARDINAL', 'PERCENT' and 'QUANTITY' (See Appendix for reference). This reason for this was the for each article corpus associated with a topic, the aim was to find information about named entities and how they related to one another. In particular, the model enforces the subject node to be about an entity. Therefore, including entity types such as 'DATE' for instance, can result in unnecessary and irrelevant triples with ('two', 'CARDINAL') as the subject node. This is not to say that any information about 'DATE', 'QUANTITY' etc. is completely ignored. Phrases containing these types of entities often show up object nodes when they are reference to a 'valid' subject named entity.

**A base noun phrase, or "NP chunk", is a noun phrase that does not permit other NPs to be nested within it, so no NP-level coordination, no prepositional phrases, and no relative clauses. copied from spacy doc**

#### Subject phrases

25: Mad convoluted - rewrite

In order to obtain the subject phrases, the noun phrases (returned as a list of Spans) are filtered to obtain those that have a starting position less than the starting position of the verb phrase as seen in line 5 in 1. This essentially results in all noun phrases that are at the left of the root verb phrase. Additionally, as mentioned in Section 6.3.2 since the subject

node is enforced to be a valid entity, the noun phrases are filtered to obtain candidate subject phrases (`valid_ents`) that mention a valid entity from the ones extracted from the sentence. From these candidate subject phrases, we obtain the first occurring (based on position of span) qualifying phrase as the subject phrase (lines 9-10). As seen in 1 on line 11, once the first qualifying phrase (i.e. entity) is found, the subject noun phrase is split to obtain the entity (which is the subject) and the remnant phrase which gets prepended to the verb phrase to form the updated relation phrase. This way the information conveyed by the noun phrase is conserved but the subject nodes are maintained as an entity so a connected knowledge graph can be formed indicating the entity as the subject and showing relationships between these entities and relationships to other object nodes.

```
1  def get_subject_relation(verb_phrase, noun_phrases, ents):
2    subject = None
3    relation = None
4    for n in noun_phrases:
5      if n.start < verb_phrase.start:
6        valid_ents = [(i,n.text.find(i)) for i in ents if n.text.find(i) != -1]
7        if len(valid_ents) == 0:
8          continue
9        valid_ents.sort(key=lambda tup: tup[1])
10       subject = valid_ents[0][0]
11       relation = n.text.partition(subject)[2].strip() + ' ' + verb_phrase.text
12       break
13   return subject, relation
```

LISTING 1 Get subject and relation

### Get object phrase

The object phrase is also derived from the noun chunks in the sentence but instead of filtering the noun phrases occuring before the root verb phrase, the candidate object phrases obtained by finding all noun phrases (spans) where their starting position in the sentence is after the root verb phrase.

This particular method avoids the stringent dependency of the ”subject” and ”object” phrases on their actual dependency positions in the grammar, This is because given the nature of news articles, the sentence structure is relatively complex with several different subjects and objects. For the knowledge graph, ultimately, the relations extracted need to focus around different named entities. Relying on the dependency grammar structure to find the subject and object dos not hold as in a given sentence, the structure of the sentence heavily influences the information extracted. This means that more often than not, relevant information about entities is lost due to the entity not necessarily having one of `"nsubj"`, `"csubj"`, `"nsubjpass"`, `"csubjpass"` as the dependency.

26: Show types of relations extracted with dep parsing and otherwise

## 27: Talk about the object-relation-subject gotcha.

**Filtering triples**

Once the semantic triples are obtained from the article corpus for each topic, the triples are filtered. After ensuring the that all components of the triple are not null, the triples where the relations/predicates contains words like ”said” and ”told” in their different forms are filtered. This is done because, often in news articles, direct quotes are made by entities and the relation extraction engine retrieves some poor relations that do not add to the quality of the relations extracted for a particular entity as seen in Figure !!!!

## 28: Show some poor relations with said and told

Once the qualifying semantic triples for each topic are obtained, these are passed to the visualisation engine. An example of the semantic relations extracted for the topics for a given cluster in Business 2021 is shown in Figure ??

# Chapter 7

# Evaluation

In this section, the main body of work is evaluated based on objectives defined at the beginning of the project (Section 1.2) and are detailed as follows:

1. Investigate the effect of different pre-processing techniques on semantic and clustering and topic modelling.
2. Determine the best approach for generating word and document vectors.
3. Investigate the quality of semantic triples extracted based on decisions highlighted in Section 6.3.
4. Evaluate the legibility of the results based on user feedback.

29: Change this based on actual eval done

## 7.1 Topic Extraction Engine

### 7.1.1 Effect of different processing techniques on clustering

In order to cluster the news articles, different pre-processing techniques were applied to the article corpora in the topic extraction engine. The motivation behind this was to gauge the effect of these techniques on the silhouettes scores and incorporate the techniques that the result in the most optimal scores indicating improved clustering. As discussed in Section 5.1, the article corpora are pre-processed using techniques such as coreference resolution (CR), removing stopwords, removing named entities before obtaining article vectors for clustering. Figure 7.1 illustrates how omitting these different pre-processing techniques affects the silhouette score of the semantic clustering obtained for the Year-Category data group: Travel 2021. We can see that not doing any pre-processing ('No pre-processing') on the articles in Travel 2021 data group results in a poor silhouette score of 0.36 compared to 'All', which results in a silhouette score of 0.691 and involves performing coreference resolution, named entity removal and stopword removal. The silhouette score

of 1 indicates dense, nicely separated clusters, a score of 0 indicates that clusters are overlapping, or distance between them is insignificant and -1 indicates incorrect clustering. Figure 7.1 shows that omitting these pre-processing techniques deteriorates the quality of clustering, by resulting in poor silhouette scores which indicate overlapping clusters.

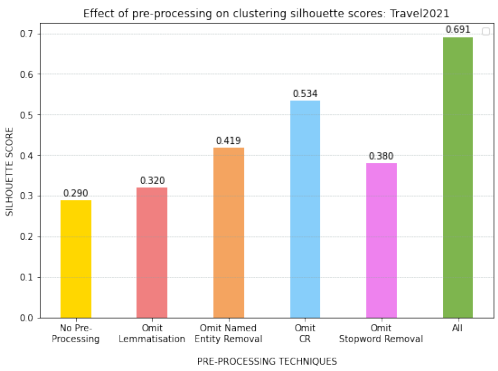


FIGURE 7.1 -

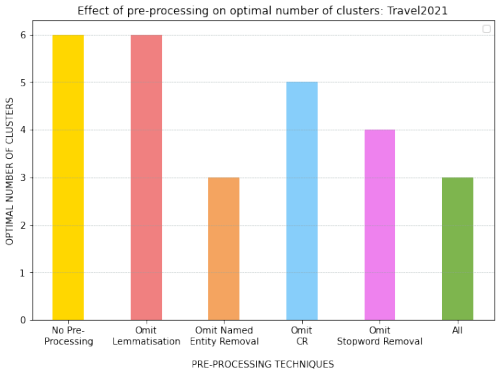


FIGURE 7.2 -

Another interesting takeaway is that post pre-processing considering the techniques in questions, usually results in lower number of clusters as seen in Figure 7.2. Generally, the aim is to get the best silhouette score with smallest number of clusters. This is because the topic extraction engine should not try to over-cluster the data by fixating on small differences within the corpus. The semantic clusters are derived for topics to be modelled within them. In this example shown in the figure, Travel 2021 was selected as it has a very few number of member articles: 17. Not performing any pre-processing results in 6 clusters for a group of 17 articles. This is excessive as it means each semantic cluster has about 2 or 3 articles. Therefore, the topics extracted from these will be modelled on an extremely small corpus of about 2 articles. For the same data group, performing all the techniques mentioned, coreference resolution, named entity and stopword removal, results in a much smaller number of clusters: 3. In conclusion, the trend established by Figures **??** and **??** is that performing the pre-processing techniques mentioned results in better silhouette scores with lower number of clusters, which indicates better clustering.

### 7.1.2 Pre-processing: Filtering on POS tags

Another key decision made to improve the clustering of the input data was to filter the tokens extracted from the articles by their Part-Of-Speech (POS) tags. The decision was made to use only nouns to the tokens that represent each article. Figures  show the optimal cluster number and silhouette score for the Year-Category data groups: Business2020, Business2020, Politics2021 and Pursuit2021 when allowed POS tags are just 'NOUN' and when they are 'NOUN', 'VERB', 'ADJ' and 'ADV'. These figures show the general trend is that having a noun only tokens list to represent the articles for generating article vectors for semantic clustering gives a higher silhouette score for a smaller number of clusters. For instance, in Figure , the silhouette score for noun only tokens list is 0.59 compared to 0.48 when tokens list contains nouns, adjectives, verbs and adverbs, resulting in a 23%

improvement in silhouette score and indicating better clustering for noun only tokens list. Same is true for Figures which show an 8.3% and 16.3% improvement respectively for clustering articles based on noun only article corpus.

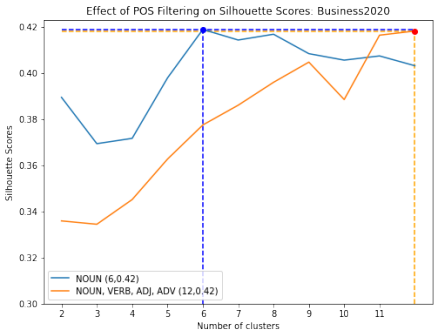


FIGURE 7.3 -

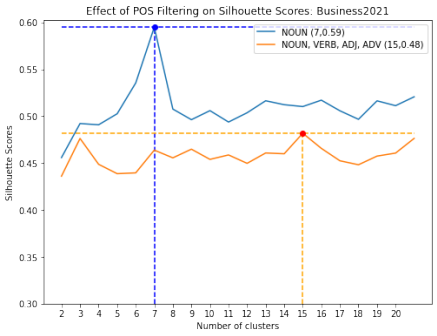


FIGURE 7.4 -

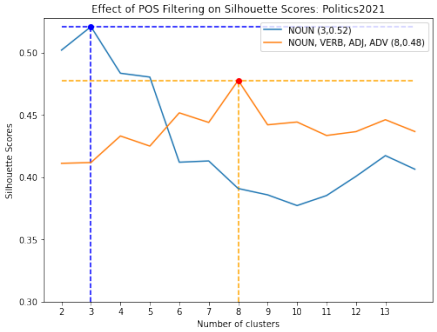


FIGURE 7.5 -

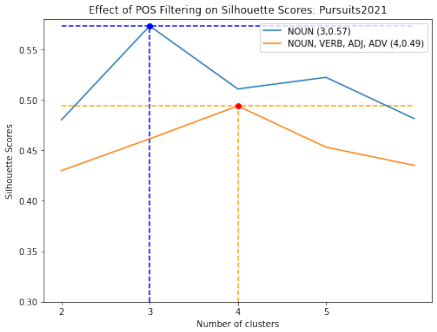


FIGURE 7.6 -

More evidently, it is important to note that the optimal number of clusters is much lower with a noun-only corpus as seen in the figures above. Figure shows that the silhouette score of 0.42 is achieved with only 6 clusters for noun-only tokens list compared to 12 with nouns, adjectives, verbs and adverbs. This makes sense as introducing adverbs, adjectives, verbs to the tokens list means that the article vectors will account for these tokens and result in a higher variance in the article vectors, making the corpus more difficult to cluster and invariable resulting in more number of clusters. This is not ideal as the aim is to extract topics form these semantic clusters and having vectors depend on adjectives and adverbs do not really contribute heavily to the semantic meaning. For example, *['coronavirus', 'aviation', 'industry', 'ticket', 'sale', 'company', 'cash', 'border', 'restriction', 'lack', 'appetite', 'travel, 'quarantine']* is more concise and general representation of an article as opposed to *['job', 'wipe', 'bad', 'come', 'worker', 'tell', 'worker', 'coronavirus', 'accord', 'calculation', 'aviation', 'industry', 'suffer', 'destroy', 'ticket', 'sale', 'strip', 'company', 'cash', 'world', 'drastically', 'cut', 'border', 'restriction', 'lack', 'appetite', 'travel', 'particularly', 'internationally', 'people', 'worried', 'contract', 'coronavirus', 'spend', 'lengthy', 'period', 'quarantine']*. The more tokens used for generating the document vectors, the greater the potential to introduce noise, resulting in high variance in these vectors, and therefore, sub-optimal clustering.

### 7.1.3 Effect of different word embedding techniques on clustering

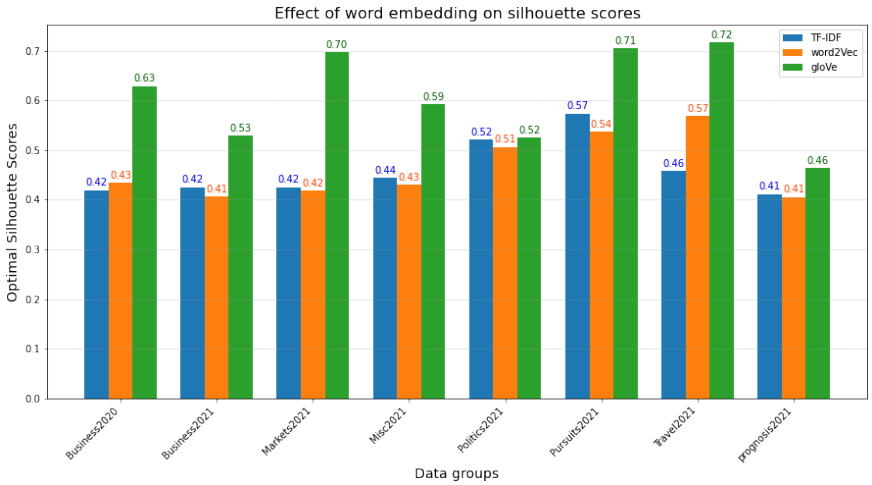


FIGURE 7.7 E-

As discussed in Section 5.2.1, in order to cluster the articles, each article's corresponding list of filtered tokens is vectorised. Figure 7.7, highlights the effect of the different word embedding techniques, in particular, TF-IDF, Word2Vec and GloVe on the silhouette scores of clusters which are a quantitative measure of of good the clustering is. As indicated by the chart in Figure 7.7, using GloVe to vectorise the articles gives a major improvement in the silhouette scores across all the Year-Category input groups, compared to TF-IDF and word2Vec.

GloVe Percentage Improvement

| Data group | TF-IDF | Word2Vec |
|---|---|---|
| Business 2020 | 50.14% | 45.02% |
| Business 2021 | 24.45% | 30.13% |
| Markets 2021 | 63.91% | 66.68% |
| Misc 2021 | 33.40% | 37.81% |
| Politics 2021 | 0.76% | 3.71% |
| Pursuits 2021 | 23.06% | 31.41% |
| Travel 2021 | 56.58% | 26.11% |
| Prognosis 2021 | 12.72% | 14.32% |
| Average | 33.13% | 31.90% |

TABLE 7.1 E-

Table 7.3 shows the (percentage) improvement in silhouette scores when using GloVe over TF_TDF and Word2Vec. We see a 33.13% improvement in clustering (by means of improved silhouette scores) when using GloVe instead of TF-IDF and and 31.90% improvement when compared to word2Vec. Based on these results, it is evident that using GloVe is the optimal approach for word embedding or token vectorisation for semantic clustering. This is attributed to the fact that most of the silhouette scores are around 0.7 which

shows evidence of distinct clusters of articles from which topics can be extracted. When clustering news articles, obtaining a perfect clustering with silhouette score of 1 is very difficult, especially when trying to cluster articles in a particular category within a particular industry.

### 7.1.4 Effect of pre-processing on topic modelling

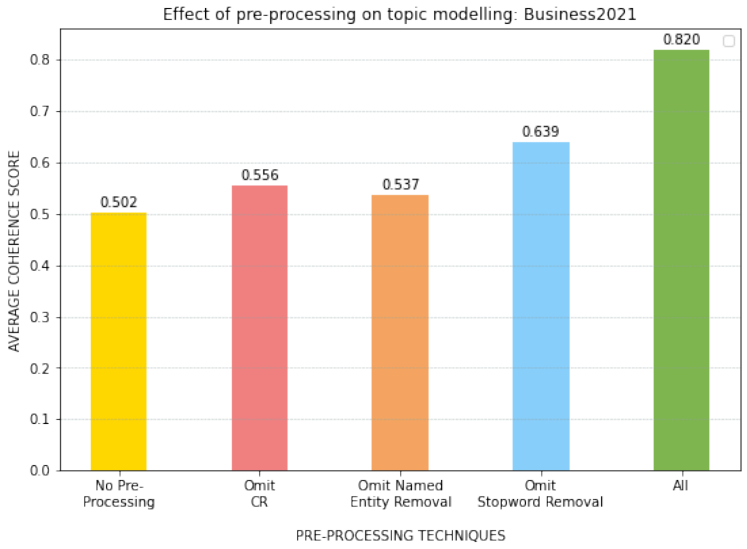


FIGURE 7.8 E-

30: Potentially add the mean median and range of coherence scores

Section 7.1.1 highlights the effect of the different pre-processing techniques on the silhouette scores for semantic clustering. In a similar fashion, Figure 7.1.5 depicts the effect of those pre-processing techniques on the topics extracted from all semantic clusters in each Year-Category groups. The topics extracted from each cluster results in a topic coherence score and these are averaged over all clusters to give an average coherence score for a Year-Category group as seen in the figure. In similar trend shown in 7.1.1, the average coherence score of the topics is significantly higher when all the pre-processing techniques are applied which include coreference resolution, named entity removal and stopword removal (represented by the 'All' bar) compared to other approaches. As mentioned previously, coherence score is a metric that assesses the degree of semantic similarity between high-scoring terms in a single topic.

In the case of omitting coreference resolution, the (average) coherence score falls as word tokens such as 'it', 'they', 'those' (essentially, unresolved pronouns) do not contribute to the semantic similarity with other high scoring terms in the corpus. Therefore, resolving these pronouns to their respective nouns, means that the frequency of the nouns increases and they are more likely to contribute to the topics in the corpus.

The stopwords that are removed in pre-processing contain around 326 words provided as the default stopwords from the SpaCy library for the large English language model `en-core-wen-lg`. These include words such as 'its', 'an', 'the', 'for', 'always', 'here', 'there', 'which', 'moreover' etc. This stopwords list is augmented with a custom list of stopwords which add words such as 'told', 'said', 'airline', 'flight' etc. which are specific to the input data. The significance of this custom list is that most articles in the data are quoting an entity usually a person, and therefore words such as 'told', 'said' etc. appear in high frequency but do not provide any relevant information about the content of the articles themselves. Similarly, given the article corpus is specifically for the airline industry, majority articles will have words such as 'airline', 'flight' etc. and these are also overarching across majority topics and do not contribute to a distinct topic. The idea is that we do not want the model to fixate on these commonly appearing contentless stopwords and output junk topics, hence why their removing them results in better topic coherence (0.82 denoted by the 'All' bar) as seen in 7.1.5 compared to when they are retained which results in a lower topic coherence of 0.639.

Similarly for named entity removal, the aim is to find distinct topics within a cluster that are not dependant on the named entities. For instance, in the input data there are several articles that mention 'British Airways' and there are articles cover latent topics such as 'pandemic' and 'pilot unions'. For arguments sake, let's presume that there is a huge overlap between 'pilot unions' and 'British Airways'. Retaining named entities in the input corpus would result in 'British Airways' the model selecting British Airways as a topic instead of 'pilot unions'.

31: Add the cv vs umass here or in topic modelling

### 7.1.5 Effect of POS on topic modelling

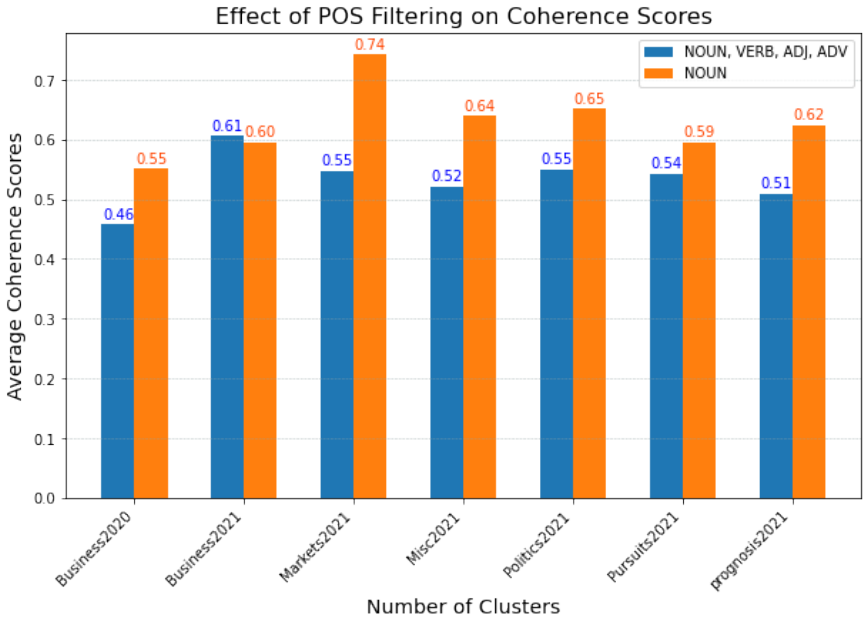


FIGURE 7.9 E-

The filtered tokens list (for each article) that was used to generate the article vectors for clustering, was passed to the LDA model to extract topics from each of the semantic clusters for a Year-Category group. In order to determine the quality of topics extracted, the coherence score measure is used. From Figure , it is apparent that the noun only tokens approach to extract tokens performs better than when noun, adjectives, verbs and adverbs are used as the average coherence scores are generally higher for the former approach. Across all the data groups, we see an average coherence score improvement of 18.2% with the noun only topic extraction approach.

## 7.2 Semantic Triple Extraction Engine

### 7.2.1 Entity as Subjects

## 7.3 Time ?????????

Time efficiency (in seconds)

| Data group | TF-IDF | Word2Vec | GloVe |
|---|---|---|---|
| Business 2020 | 44.67 | 33.82 | 39.47 |
| Business 2021 | 256.19 | 130.56 | 138.47 |
| Markets 2021 | 87.30 | 75.00 | 74.18 |
| Misc 2021 | 16.67 | 17.92 | 18.60 |
| Politics 2021 | 60.72 | 71.06 | 64.87 |
| Pursuits 2021 | 15.00 | 14.38 | 15.00 |
| Travel 2021 | 13.06 | 12.85 | 12.92 |
| Prognosis 2021 | 53.61 | 59.39 | 51.63 |
| Total time (s) | 414.98 | 547.922 | 414.60 |
| Total time (min) | 6.92 | 9.12 | 6.90 |

TABLE 7.2 E-

Time efficiency (in seconds)

| Data group | CPU | CPU with Multi-processing | GPU |
|---|---|---|---|
| Business 2020 | 351.41 | | 39.47 |
| Business 2021 | 1713.6 | | 138.47 |
| Markets 2021 | 839.75 | | 74.18 |
| Misc 2021 | 215.51 | | 18.60 |
| Politics 2021 | 856.42 | | 64.87 |
| Pursuits 2021 | 178.54 | | 15.00 |
| Travel 2021 | 169.52 | | 12.92 |
| Prognosis 2021 | 583.33 | | 51.63 |
| Total time (s) | 4908 | | 414.60 |
| Total time (min) | 81.8 | | 6.90 |

TABLE 7.3 E-

## 7.4 User Evaluation

# Chapter 8

# Conclusion

# Todo list

## 8.1 Summary of achievement

## 8.2 Future Work

Topic Bert

# Bibliography


[1] S. Kannan, V. Gurusamy, S. Vijayarani, J. Ilamathi, M. Nithya, S. Kannan, and V. Gurusamy, "Preprocessing techniques for text mining," *International Journal of Computer Science & Communication Networks*, vol. 5, no. 1, pp. 7–16, 2014.

[2] T. Al-Moslmi, M. G. Ocaña, A. L. Opdahl, and C. Veres, "Named entity extraction for knowledge graphs: A literature overview," *IEEE Access*, vol. 8, pp. 32 862–32 881, 2020.

[3] S. Petrov, D. Das, and R. McDonald, "A universal part-of-speech tagset," 2011. [Online]. Available: https://arxiv.org/abs/1104.2086

[4] V. Balakrishnan and E. Lloyd-Yemoh, "Stemming and lemmatization: a comparison of retrieval performances," 2014.

[5] P. Willett, "The porter stemming algorithm: then and now," *Program*, 2006.

[6] D. L. Yse, "Text normalization for natural language processing (nlp)," 2021. [Online]. Available: https://towardsdatascience.com/text-normalization-for-natural-language-processing-nlp-70a314bfa646

[7] J. Zheng, W. W. Chapman, R. S. Crowley, and G. K. Savova, "Coreference resolution: A review of general methodologies and applications in the clinical domain," *Journal of biomedical informatics*, vol. 44, no. 6, pp. 1113–1122, 2011.

[8] A. Brown, *Pronunciation and phonetics: A practical guide for English language teachers*. Routledge, 2014.

[9] D. Jurafsky and J. H. Martin, "Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition." [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/21.pdf

[10] K. Lee, L. He, and L. Zettlemoyer, "Higher-order coreference resolution with coarse-to-fine inference," *arXiv preprint arXiv:1804.05392*, 2018.

[11] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, "End-to-end neural coreference resolution," *arXiv preprint arXiv:1707.07045*, 2017.

[12] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," 2019. [Online]. Available: https://arxiv.org/abs/1907.10529

[13] "spacy 101: everything you need to know," 2022. [Online]. Available: https://spacy.io/usage/spacy-101

[14] "Natural language processing in python: Nltk vs. spacy," 2021. [Online]. Available: https://www.thedataincubator.com/blog/2016/04/27/nltk-vs-spacy-natural-language-processing-in-python/

[15] 2022. [Online]. Available: https://allenai.org/allennlp/software/allennlp-library

[16] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[17] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," 2016. [Online]. Available: https://arxiv.org/abs/1603.01360

[18] "cloud application platform: Heroku," 2022. [Online]. Available: https://www.heroku.com/

[19] K. W. CHURCH, "Word2vec," *Natural Language Engineering*, vol. 23, no. 1, p. 155–162, 2017.

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

[21] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *Acm transactions on knowledge discovery from data (tkdd)*, vol. 3, no. 1, pp. 1–58, 2009.

[22] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" in *International conference on database theory*. Springer, 1999, pp. 217–235.

[23] L. Muflikhah and B. Baharudin, "Document clustering using concept space and cosine similarity measurement," pp. 58–62, 2009.

[24] D. M. Saputra, D. Saputra, and L. D. Oswari, "Effect of distance metrics in determining k-value in k-means clustering using elbow and silhouette method," in *Sriwijaya International Conference on Information Technology and Its Applications (SICONIAN, 341–346. Indonesia: Atlantis Press*, 2020.

[25] T. M. Kodinariya and P. R. Makwana, "Review on determining number of cluster in k-means clustering," *International Journal*, vol. 1, no. 6, pp. 90–95, 2013.

[26] M. Riedl and C. Biemann, "Topictiling: a text segmentation algorithm based on lda," in *Proceedings of ACL 2012 Student Research Workshop*, 2012, pp. 37–42.

[27] F. Martin and M. Johnson, "More efficient topic modelling through a noun only approach," in *Proceedings of the Australasian Language Technology Association Workshop 2015*, 2015, pp. 111–115.

[28] H. M. Wallach, "Topic modeling: beyond bag-of-words," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 977–984.

[29] D. M. Blei and J. D. Lafferty, "Topic models," in *Text mining*. Chapman and Hall/CRC, 2009, pp. 101–124.

[30] S. Syed and M. Spruit, "Full-text or abstract? examining topic coherence scores using latent dirichlet allocation," in *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2017, pp. 165–174.

# Appendix

## Coocc