Imperial College London

MEng Individual Project

Interim Report

Imperial College London

Department of Computing

Information Retrieval through Semantic Analysis of News Corpora

Author:
Akanksha Sharma

Supervisor:
William Knottenbelt

Second Marker:
Paul Bilokon

June 15, 2022

Abstract

P1: Context/motivation P2/3: Technical Contribution P4: Summarise Results

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

1 Introduction

In today's fast-growing, data-driven economy, there is an incomprehensible amount of information available at our disposal. Several thousand articles are published daily made available through a host of different online sources. For an average consumer, this can be extremely overwhelming and difficult to navigate. This poses a need for a concise and digestible medium of information that allows consumers to acquire insight into significant themes, events, and entities around their areas of interest, perhaps throughout a specific time period, without having to wade through reams of superfluous articles. This prompts performing semantic analysis on the news articles, in order to extract relevant and distilled information from the news corpora.

1.1 Motivation

Information extraction or retrieval from online news corpora is huge area of research in the natural language processing (NLP) community. With an overwhelming amount of information at disposal, the main task in information extraction entails automatic extraction of structured information such as entities, relationships between entities, etc., from unstructured sources through syntactic parsing, which exploits the structure of the natural language model focusing on lexical meaning and part-of-speech, and semantic analysis, which features the meaning of text in the corpus [1] [2]. Early works in this area involved performing identification and temporal analysis of named entities, such as persons, organisations etc. and their co-occurrence in news articles [2] [3], event detection [4] [5] and information distillation by news summarisation, sentence relevance identification [6] [7]. A more interesting way of extracting information involves generating semantic triples of type (subject, predicate, object) from news articles to generate a knowledge graph, which provides the benefit acting as a minimal representation of the information presented in news articles, whilst still retaining sufficient context. Previous works on this approach, however, rely on using ontologies and augmenting existing knowledge bases for triple extraction, making them domain-dependant [8]. Our work explores the domain-independent approach which uses the rich morpho-syntactic marking system of English (verb inflections, clausal markers, nominal case) [9] to identity the syntactic structure of triples.

Another prevalent area of research in semantic analysis of news articles follows topic modelling, which involves extracting latent topics from the news article corpora using multinomial distributions over a fixed vocabulary [10]. The motivation for this stems from extracting common themes in large corpus of articles and is often combined with the another NLP approach: sentiment analysis, to provide an insight into the sentiment surrounding the latent

topics. Common approaches for this include using Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA) models. Alternatively, clustering methods are also quite common to group articles into topics in an attempt to achieve a high-level similarity clusters of articles [11] that rely more heavily on the semantic meaning of the words rather than probabilistic distributions of keywords (as seen in LDA).

1.2 Objectives

Our work draws on the semantic analysis techniques discussed above to provide a semantic analysis engine, with the aim to help alleviate information overload by distilling the knowledge extracted from the news corpora (into topics and corresponding triples) associated with a given sector, which for the scope of this project, is the airline industry. The objectives for constructing the semantic analysis engine are as follows:

1. **Cluster articles based on the semantic similarity of their text.** Vectorise articles experimenting with different embeddings and vectorisation techniques in order to generate semantic document vectors which can be clustered using KMeans.
2. **Topic Modelling on semantic clusters to obtain information on latent topics.** Generate latent topic models to indicate key themes in the clustered article corpus, experimenting with different processing techniques on the input corpus to improve coherence of the latent topic models.
3. **Extraction of semantic triples for each topic in cluster.** Extract semantic triples to provide an overview of the entities, their relationships, the general sentiment around events and news surrounding them.
4. **Visualisation of results from information extraction** Display the results of the information extracted from the semantic analysis, semantic clusters, latent topics and triples extracted in a cohesive visualisation.

1.3 Contributions

This project focuses on building a cohesive semantic analysis tool, that incorporates the different semantic approaches and has the following contributions:

1. **Topic Extraction Engine:** A novel engine involves semantic clustering of news articles based experimenting with different document vector generation approaches, using KMeans as the clustering technique. The resultant semantic clusters then undergo topic modelling to insight into the topics and their associated articles, sentiment to a category of input data (See Chapter 4).
2. **Semantic Triple Extraction Engine:** The engine focuses on extracting semantic triples of type (subject $\rightarrow$ predicate $\rightarrow$ object) focusing the on the named entities and information surrounding them, inferring the average sentiment of the entity (See Chapter 5).
3. **Visualisation Tool:** A web-based application responsible for illustrating the results from the Topic Extraction Engine and Semantic Triple Extraction Engine in a coherent and co-

hesive visualisation by making use of circle packing topic cluster diagram and a knowledge graph respectively.

Such a product has great value and interest for not only the average consumer 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.

1.4 Ethical Considerations

This project focuses on semantic analysis of news articles with aim of knowledge distillation which from an intelligence standpoint, is extremely useful as it condenses large amounts of news to give a visual overview of an industry in a specific category over a period of time. Though this project might highlight recent developments and trends in a sector, it is extremely unlikely to influence and propagate some radicalised opinion. Additionally, given that there is no direct human involvement in this project, there is no collection or direct use of personal data.

One important ethical consideration, however, is to think about any bias introduced in the semantic analysis engine due to the nature of the input data which inherently might contain some underlying bias. For example, using the engine with a different input dataset focusing on the energy industry, which consists of majority right-wing articles talking about “climate change being a hoax’ and refusing to support clean energy alternatives, then that will be reflected in the output results (topics, associated sentiments and semantic triples) via the visualisation tool, illustrating a negative sentiment associated with clean energy. This is the nature of news publication, which is more often than not biased and polarising. This makes it very difficult to gauge what quantifies as unbiased, in fact, performing any filtration of articles to “alleviate bias” would inherently introduce bias.

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 dataset is provided by DeepSearchLabs and are scraped from online sources, in particular, Bloomberg News. 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” [12]. All other libraries and software packages used such as D3, spaCy, Gensim and AllenNLP (See Section 3.3.1) are open-source and free to use for the purposes of this project (i.e., for academic use).

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 [13] [14].

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

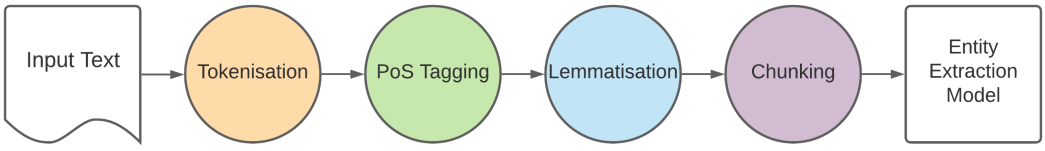

Figure 2.1: Text preparation pipeline

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 [13] [14]. 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 [13]. 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. [15] 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 POS tags is shown in Appendix Table A.1

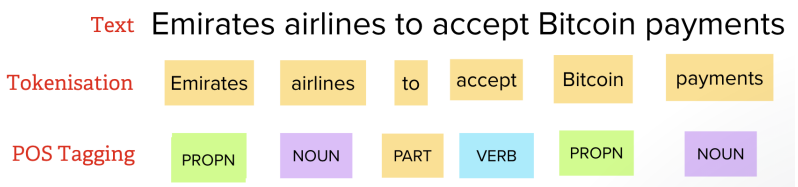

Figure 2.2: Tokenisation and POS tagging

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 [16]. 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 [16]. 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 [17]. 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 [18]. 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 [16]. 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 [16].

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.

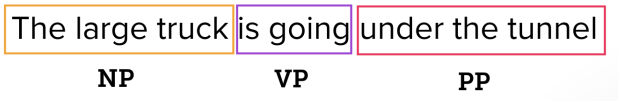

Figure 2.3: Chunking

2.2 Dependency Parsing

1: 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) [19]. 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 [20]. 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 [19].

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 [21] (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 [22] 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 [22].

2: 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 [21]. This is seen in Lee 2017 [23], a model on which a lot of recent state of the art models such as SpanBERT [24] 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 [23][22].

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)

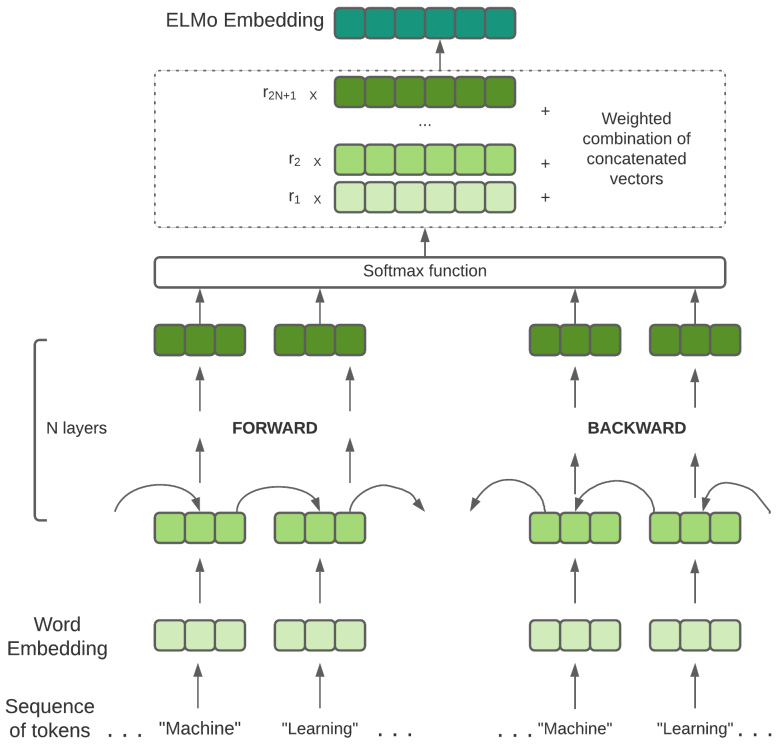

Figure 2.4: Embeddings from Language Models (ELMo) Context Representation

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 Knowledge Graph

2.8.2 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.

2.8.3 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 visualisations. [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 to not confuse the user and discourage participation.

3: Background: LDA

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

5: Background: Remove visualisation

3 | Technical Architecture

3.1 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 Figure 3.1

	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 3.1: 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.

3.2 System Design

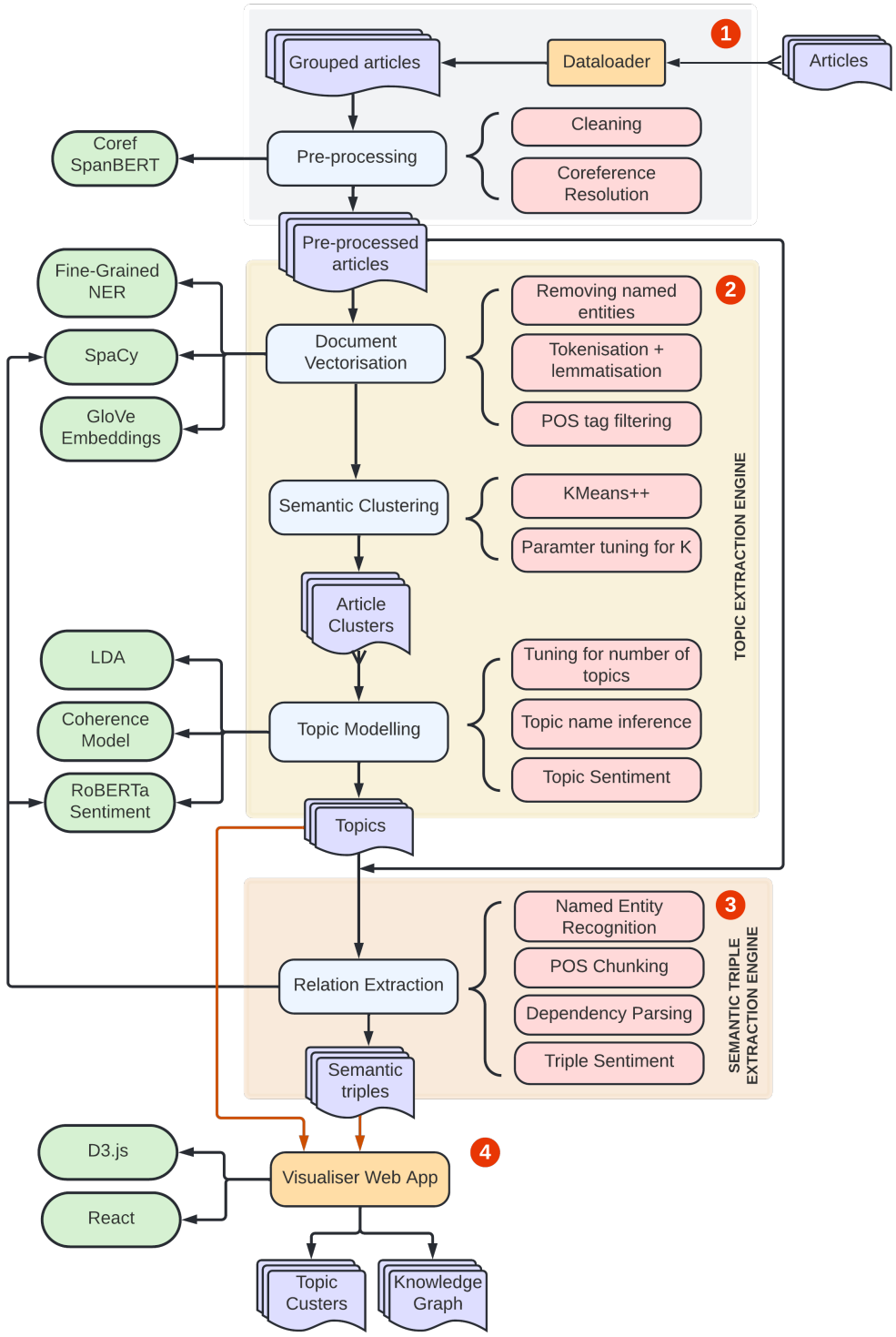

Figure 3.2: System Architecture Diagram

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 3.2, 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 Section 3.1. 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.

3.3 Technologies used

3.3.1 Libraries

SpaCy is an open-source NLP library "designed specifically for production use" [25]. 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 [26]. 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 [27]. 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 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 other alternatives was it's data-driven approach to DOM manipulation which enables building customisable interactive visualisation frameworks.

3.3.2 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 [24]. 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 [24].

RoBERTa Stanford Sentiment Treebank is a binary classifier trained on RoBERTa large for Stanford Sentiment Treebank dataset, on which it achieved 95.11% accuracy [28]. 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) [28].

Fine-Grained NER is a state of the art named entity recognition model from AllenNLP and is a re-implementation of the Lample (2016) [29] 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 2.5.1–2.4.2). 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% [29]. The motivation for using this model was this high accuracy and that the model identifies a broad range of 16 semantic types.

3.4 Visualisation Web Tool

As an extension to the core body of work done in semantic clustering, latent topic model extraction (done by Topic Extraction Engine) and inferring semantic triples (done by Semantic Triple Extraction Engine), an interactive visualisation pipeline was created to display the results from the two engines in the semantic analysis tool. This was done by developing a web interface using React and D3.js to build stateful cohesive visualisations, allowing the integration of all the different results in a web application. In the latter stages of the project, the application was hosted on Heroku [30] as was a scalable approach to accommodate user testing and aided in the evaluation discussed in Section 6.3.

4 | 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.

4.1 Data Processing

The prepared data (from Section 3.1) 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 chapter, '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.

4.1.1 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'.

4.1.2 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.

4.1.3 Named Entity Recognition

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. A key decision made was to remove all corresponding named entities form article intro in order to remove any dependency of the topics on these named entities. Eliminating this dependency resulted in better silhouette scores and coherence scores for semantic clustering and topic extraction as detailed in Section 6.1.1 and Section 6.1.4 respectively.

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.

The process of extracting these entities from the article corpus using the Fine Grained NER model (refer Section 3.3.2) and the BILOU encoding scheme for NER is detailed in Algorithm 1.

B	**I**	**L**	**O**	**U**
beginning	inside	last	outside	unit

The output from the NER model gives a set of words and their corresponding entity tags. In essence, the algorithm shows the process of joining the entity tokens to build the entity phrases starting from the 'beginning' token until the 'last' token. Words with 'Outside' as a tag mean that the word is not an entity while 'unit' refers to single word entities.

Algorithm 1 Extract Named Entities

```
entities ← empty set
words, tags ← ner_model.predict(sentence)
for all word, tag ∈ zip(words, tags) do
    if t == 'O' then
        continue
    ent_position, ent_type = tag.split('–')
    if t == 'U' then
        entities.add(word)
    else
        if ent_position == 'B' then
            ent := word
        else if ent_position == 'I' then
            ent := ent + word
        else if ent_position == 'L' then
            ent := ent + word
            entities.add(ent)
```

4.1.4 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 3.3.1). 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 6.1.2.

4.2 Semantic Clustering

The first step to generating semantic clusters of articles is to get the corresponding article vectors. This involves vectorising the individual tokens in `filtered_tokens` list for each article intro.

4.2.1 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 [31].

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, consisting of a vocabulary of 400,000 words which are represented by 300 dimensional vectors [32]. 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 6.1.3.

4.2.2 Document embeddings

In Section 4.1.4, `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 [32] 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.

Approach 3 Building on the previous approach, this method selects the top 10 most representative words based on the TF-IDF weightings and performs the weighted average to generate the article vector.

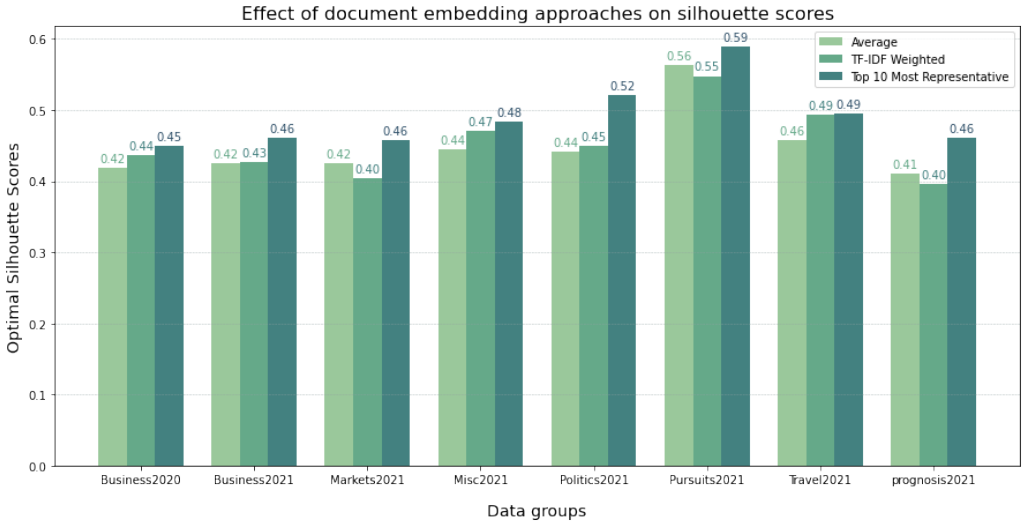

FIGURE 4.1 Effect of document vector generation approaches on clustering silhouette scores

Figure 4.1 shows how the different word embeddings approaches effect the 'goodness' of clustering indicated by silhouette scores. The consensus derived is that Approach 3, which describes calculating the article (document) vector by computing the weighted average of the top 10 most relevant words (using the TF-IDF metric), performs better than the other 2 approaches mentioned. Approach 3 results in a 9.36% improvement in silhouette score

compared to Approach 1 (simple averaging of word vectors) and an 8.37% improvement compared to Approach 2 (weighted (TF-IDF) average of all word vectors). This led to the decision of using Approach 3 as the method for document vector generation.

4.2.3 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 vectors, the vector components (features) should be independent of their standard deviation or variance to get a good covariance matrix among the features.

Principal Component Analysis

As established previously (refer to Section 4.2.1), `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 [33] 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 [34].

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

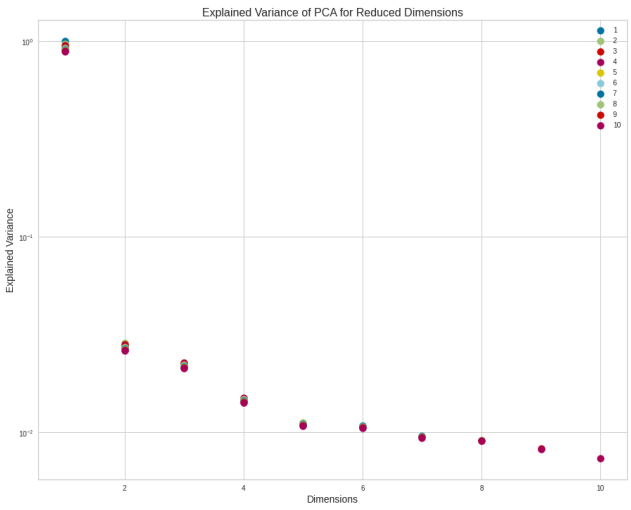

FIGURE 4.2 -

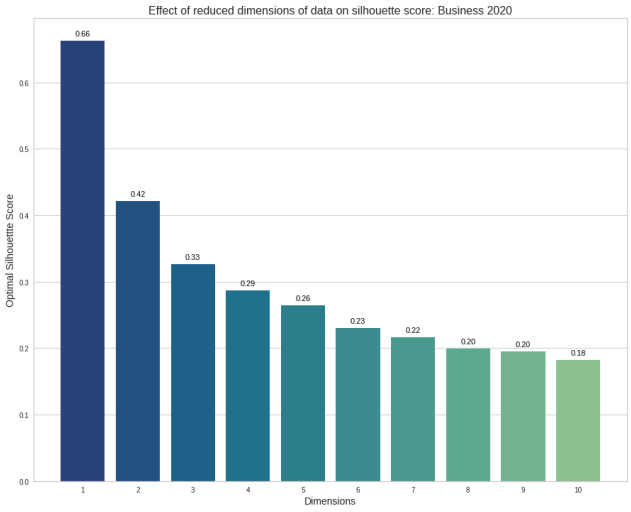

FIGURE 4.3 -

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 data, some of these attributes will not be relevant in

determining the 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 lower dimensional space, whilst minimising information loss [33].

As seen in Figure 4.2, when the article vector space is projected to dimension=1, the variance is explained entirely by dimension 1 (blue dot). As the number of dimensions increase, the number of principal components increase, the main proportion of the variance can still be explained by the first principal components (close to 1). For instance, projecting to dimension=10, the explained variances for each of the 10 principal components is as follows:

PC	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
EV	0.88	0.026	0.021	0.014	0.011	0.010	0.0094	0.0090	0.0082	0.0074

Therefore, 88% of the variance (EV) can be explained by the first principal component (PC) alone and so the article vector space can be transformed using PCA with dimensions=1. Figure 4.3 shows the effect of transforming the article vector space to different dimensions on the silhouette scores, which represent the 'goodness of clustering'. It indicates that transforming the vectors to a single dimension (using PCA) result in the highest silhouette score. Therefore, Figures 4.2–4.3 conclude that the setting the dimension to 1 for PCA transformation results in the most optimal clustering whilst retaining majority of the information (variance).

4.2.4 KMeans Clustering

Once the normalised data is mapped to the lower subspace, the articles are clustered based on their cosine similarity (4.2) using cosine distance as the distance function (4.3) and KMeans as the clustering technique. This was done using the `sklearn` library's `kmeans` function with 'euclidean distance' (4.4) as distance metric. This is feasible given the linear relationship between Euclidean distance and cosine distance [35] for normalised vectors shown in (4.6).

$$\textit{For normalised vectors, } x, y : \sum x_i^2 = 1, \sum y_i^2 = 1 \tag{4.1}$$

$$\begin{aligned} \textit{Cosine Similarity, } cos(x,y) &= \frac{\sum x_i y_i}{\sqrt{\sum x_i^2 y_i^2}} \\ &= \sum x_i y_i \end{aligned} \tag{4.2}$$

$$\textit{Cosine Distance} = 1 - cos(x,y) \tag{4.3}$$

$$\begin{aligned} \textit{Euclidean Distance, } ||x-y||_2^2 &= \sum (x_i - y_i)^2 \\ &= \sum (x_i^2 + y_i^2 - 2x_i y_i) \\ &= \sum x_i^2 + \sum y_i^2 - 2\sum x_i y_i \\ &= 1 + 1 - 2cos(x,y) \\ &= 2(1 - cos(x,y)) \end{aligned} \tag{4.4}$$

$$= 2(\textit{Cosine Distance}) \tag{4.5}$$

$$\tag{4.6}$$

Furthermore, to avoid falling into the trap of random centroids initialization, a major shortcoming of the KMeans method, the clustering was done with KMeans++. 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.

Determining the optimal number of clusters

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 Within Cluster Sum of Square distance (WCSS) and Silhouette score.

Method 1: Minimising WCSS using 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 [36].

$$WCSS = \sum_{C_k}^{C_n} (\sum_{d_i \in C_i}^{d_m} ||d_i - C_k||_2^2) \tag{4.7}$$

As seen in Figure 4.4, the elbow (bend) in the plot determines the optimal number of clusters, i.e., the k value = 4. The main drawback of using the lowest distortion score to determine the optimal number of clusters is that as long as the number of clusters (k) increases, the distortion score (WCSS) will decrease because the points will be closer to their centroids. Hence, the elbow (bend) is used to determine the minimum number of clusters with a reasonably low distortion score as seen in Figure 4.4, which gives the optimal number of clusters, k=4.

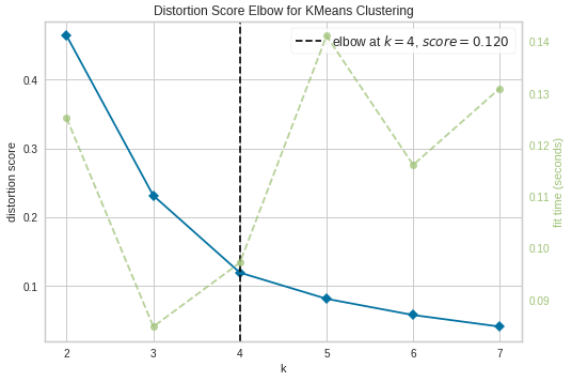

FIGURE 4.4 'Business 2020' Elbow Method Plot for k=2..7

Method 2: Silhouette score

The second approach involved using the silhouette score. Unlike, the elbow method silhouette score accounts for both how close a point is within its cluster (cohesion) [36] and other clusters (separation) [37].

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

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 0 indicates significant overlapping between the clusters, and the clusters are not distinguished. A value near -1 indicates that clusters are assigned incorrectly.

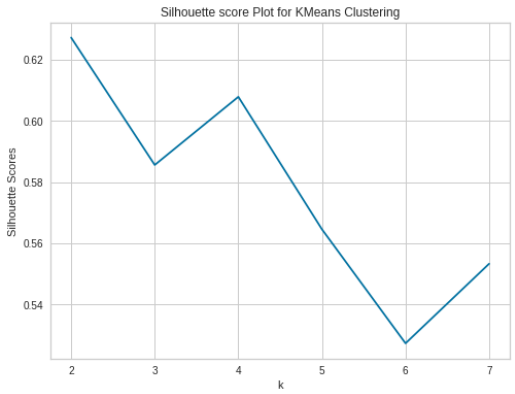

FIGURE 4.5 'Business 2020' Silhouette Plot for KMeans clustering with k=2..7

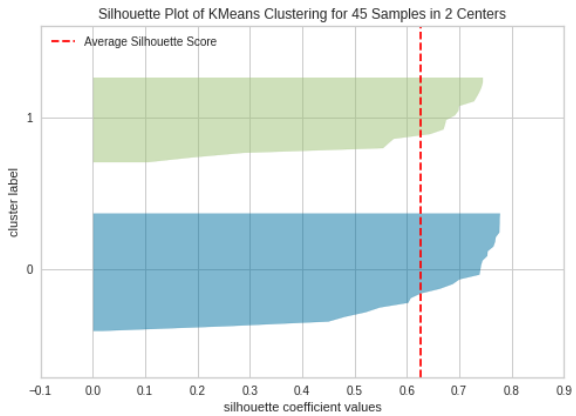

FIGURE 4.6 'Business 2020' Silhouette Diagram: score = 0.627 for k=2

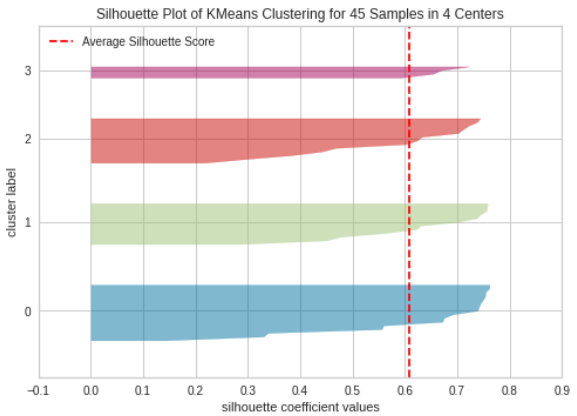

FIGURE 4.7 'Business 2020' Silhouette Diagram: score = 0.607 for k=4

From Figure 4.5, it is apparent that while selecting k=4 (as found by Elbow Method in Figure 4.4) results in a good clustering, a better clustering can be achieved by selecting k=2, as it gives a higher silhouette score, suggesting less overlap. Figures 4.6–4.7 show the silhouette diagram for number of clusters, k=2 and k=4 respectively. The cluster thickness denotes the

size of the cluster, while the width represents the sorted silhouette coefficients of instances in the cluster. Observing Figure 4.6 (k=2) reveals that both clusters are roughly equal in size, whereas Figure 4.7 (k=4), illustrates a higher disparity in size as Cluster 3 is less than half the size of others. This is indicative of some clusters being split, thereby resulting in a suboptimal silhouette score.

Out of the two approaches, the decision was made to go with the silhouette score as it factors both how compact a cluster is and how distinct it is. This is a vital consideration given that the aim of the engine is to distinctly cluster the news articles with minimal overlap, in order to minimise common topics within clusters post topic modelling (performed on each cluster).

Find n most representative docs

Once the articles are clustered, the goal is to obtain a cluster-to-article mapping. However, given the high volume of data, it is not as necessary to consider every single article in the cluster particularly those that are not close to their respective cluster centroids. A pre-determined value n_{max} is selected, and the euclidean distance from each member (article) to its centroid is computed. The articles are sorted in increasing order of distance and the top n articles are selected for each cluster. Furthermore, since the next step in the topic extraction engine entails modelling topics on these semantic clusters, it is futile to consider the sparse clusters that have very few (less than n_{min}) members as they will not spawn any good topics and are therefore omitted. These values n_{max} and n_{max} are calculated based on the mean and standard deviation of the sizes of semantic clusters obtained as shown in (4.8)–(4.9).

$$n_{min} = \lfloor mean(|C_1|, |C_2|, ..., |C_n|) - std(|C_1|, |C_2|, ..., |C_n|) \rfloor \tag{4.8}$$

$$n_{max} = \lceil mean(|C_1|, |C_2|, ..., |C_n|) + std(|C_1|, |C_2|, ..., |C_n|) \rceil \tag{4.9}$$

4.3 Topic Modelling

Post semantic clustering, the aim is to extract 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 explained in Section 2.6.1, each document (article intro) is made up of several words (`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 as well as the word distribution for each topic. The motivation for using LDA 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.

It is important to note that topic modelling LDA does not rely on semantic information, this is why the decision was made to obtain semantic clusters first and then model topics on them.

4.3.1 Corpus Decisions

From the data processing steps detailed in Section 4.1, the article intros have already been cleaned, lemmatised, filtered and tokenised to get their corresponding `filtered_tokens`,

which are void of any stopwords and named entities and only contain nouns. For the input corpus passed to the LDA, there were certain key decisions made, which were influenced by previous studies and quantitive evaluation (discussed in Section 6.1). Using the `filtered_tokens` satisfied most of these requirements which are explained below:

1. **Noun-only corpus**: Based on previous studies [10] [38], and the results from Section 6.1.5 which saw improved coherence values of topics averaged across clusters and fewer numbers of 'unpopular' topics (i.e. those with few associated articles), the decision was made to limit input corpus to nouns (satisfied by `filtered_tokens`). This aligned with the intuition that nouns are better indicators of a topic as they provide more semantic context. Opening the corpus to other POS categories, for instance, ADJ, VERB (Appendix Table A.1), will result in sub-optimal topics since the LDA model will give "fast" and "pandemic" equal importance.
2. **No Named Entities**: Similar to semantic clustering, the decision as made to omit all named entities from the LDA corpus. This was also satisfied by the `filtered_tokens` where the entities found using the Fine-Grained NER model were removed. As seen in Figure 6.10, omitting named entities improved the coherence score of the topics.
3. **Using Bigrams**: Inspired by [39, 'Beyond bag of words'], a decision made was to augment the corpus using n-grams, in particular, bigrams, rather than solely using unigrams. This allowed the model to consider a combination of two commonly occurring words across documents, e.g., "coronavirus_pandemic", "aviation_industry" etc. This was particularly relevant given the dataset of news articles containing common 'noun chunks' which can be used to infer topics. The bigrams are derived using Gensim's `Phraser` which uses colocation detection, and are appended to `'filtered_tokens'`.
4. **TF-IDF**: In LDA, the document need to be transformed into numeric feature vectors which form the corpus. Since it does not rely on the article's semantic information and builds topic estimators based on words, article 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 [40], in an attempt to minimise overlap of topics as each article will be associated with one dominant topic.

4.3.2 Determining number of latent topics

A key contributor in determining the quality of the topics extracted is tuning the num_topics parameter which represents number of topics outputted by the LDA. A low value will result in too few or overly generalised topics whereas a high value results in uninterpretable topics that ideally should have been merged [41]. Previous studies [42] [43] show that one of the best methods to determine the number of latent topics is to measure CV coherence, which has a high correlation with human judgments of topic interpretability. This metric computes the normalised pointwise mutual information (NPMI) and cosine similarity of content vectors of words, derived based on their co-occurrences [43].

This was achieved by using `CoherenceModel` with the 'c_v' setting from the Gensim library. The optimal number of topics (i.e., value resulting in the highest coherence score) is computed for each cluster by running the LDA model with different values of k varying from

min_topics (defaulted to 2) to max_topics (dependent on the size of cluster), and the LDA model with the highest resulting coherence is chosen. The motivation for bounding the max_topics was to ensure topics would have at least the minimum number of article intros (=2). Additionally, pruning the number of LDA runs had performance gain.

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-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 article belongs to the topic corresponding to the `topicId`. Of these, the most dominant topic (the highest likelihood of belonging to the topic) is selected, resulting in an 1-1 topic-article mapping.

Topic Name Inference

An augmented feature of the Topic Modelling Engine was to infer the topic names from the keywords of the topic. Algorithm 2 details the process of inferring topic names from the corresponding topic keywords. This involves splitting any bigram keywords to obtain a set of keyword tokens which are checked to ensure they are nouns, not a stopword and have an associated `Glove` vector. Using the `glove_model.most_similar()` method from Gensim, which computes the cosine similarity between an average of projection of resulting keyword vectors (`validKws`), the candidate topic names are obtained. These are again checked for validity in terms of whether they are a noun and are not augmented stopwords. The top 2 most similar topic name tokens that meet the validity checks are selected as the topic name for the given topic.

Algorithm 2 Infer Topic Name

function GETVALIDKEYWORDS($topicKeywords$, $stopwords$, $allowed\_pos = [`NOUN']$)
  $validKws \leftarrow \emptyset$
  for all $topicKw \in topicKeywords$ **do**
    $kw \leftarrow topicKw.lemma$
    if $kw \in stopwords$ **and** $kw \notin GloVe.vocab$ **and** $kw.pos \notin allowed\_pos$ **then**
      $continue$
    $validKws.append(kw)$
  return $validKws$

function GETTOPICNAME($topicKeywords$, $stopwords$)
  $kws \leftarrow$ GETKEYWORDTOKENS($topicKeywords$) ▷ *splits bigrams*
  $validKws \leftarrow$ GETVALIDKEYWORDS($kws, stopwords$)[:5]
  $topicNameCands :=$ GloVe.$most\_similar(validKws, topn = 5)$
  $validTopicNames \leftarrow$ GETVALIDKEYWORDS($topicNameCands, stopwords$)
  if $len(validTopicNames) == 0$ **then**
    return $validKws[0]$
  else
    return $validTopicNames[0:2]$

Topic Sentiment

Algorithm 3 Computing Topic Sentiment

function GETTOPICSENTIMENT($topicArticles$)
  $sentiments \leftarrow \emptyset$
  for all $article \in topicArticles$ **do**
    $sentiments.append(roBERTa\_model.predict(article.title))$ ▷ 0: N, 1:P
  $avgSent = average(sentiments)$
  if $avgSent >= 0.4$ **and** $avgSent <= 0.6$ **then**
    return 'Neutral'
  else if $avgSent > 0.6$ **then return** 'Positive'
  elsereturn 'Negative'

The last component of the Topic Modelling Engine involves sentiment analysis. In order to compute the topic sentiment, the sentiment of each article associated with the topic is computed. For this, the RoBERTa Sentiment Treebank model is used on the article title as it serves as good descriptor of the tone of article. The model outputs a binary label, where 0 indicates 'negative' sentiment and 1 indicates 'positive' sentiment. These are then averaged to get the sentiment of the topic. Based on the range of this value it is assigned one of the following: 'Positive', 'Negative' and 'Neutral' as shown in Algorithm 3.

4.4 Results and Discussion

This section focuses on the qualitative analysis of the results obtained from the Topic Extraction Engine displayed by the visualisation tool.

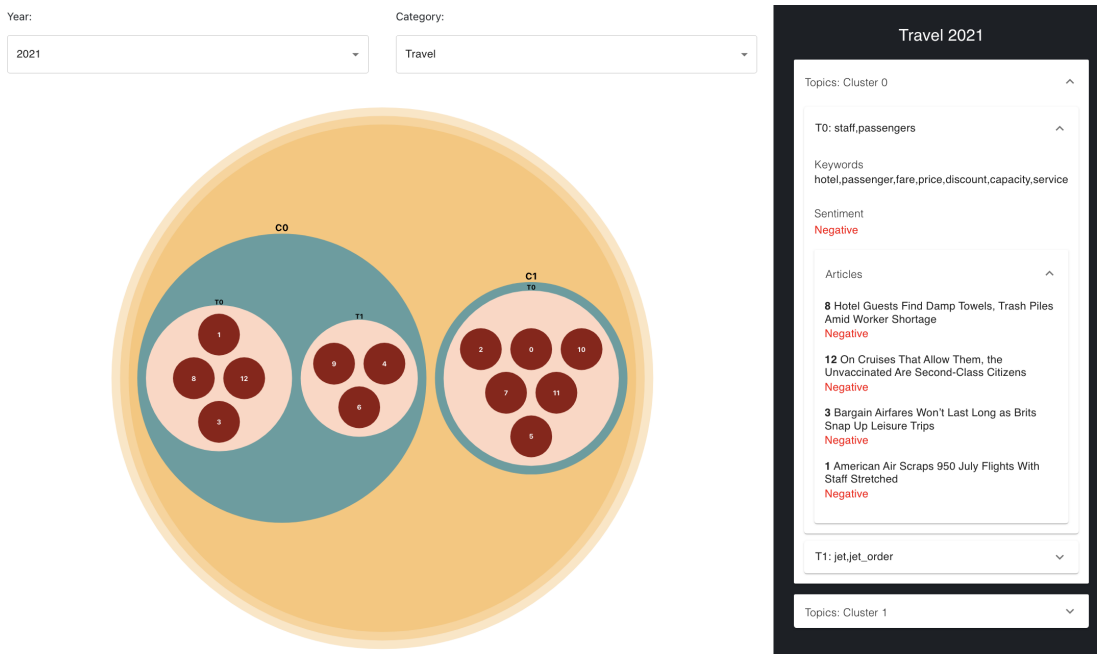

FIGURE 4.8

The aim of the was to derive semantic clusters from the article corpus for each Year-Category group and apply topic modelling from on these clusters to obtain topics and determine the articles belonging to these topics. This is shown in Figures 4.8–4.10.

Figure 4.8 displays the 'cluster-topic' graph for 'Travel2021' which has 2 semantic clusters C0 and C1, with 2 and 1 topics respectively. This particular input group contained limited data and is used as an example for ease of understanding (See Appendix ???? for additional 'cluster-topic' graphs for other input groups). We can see that one of the topics (T0) in Cluster 0 (C0) has the inferred topic name as 'staff, passengers' with the topic keywords: 'hotel', 'passenger', 'fare', 'price', 'discount', 'capacity' and 'service'. The articles within this topic (numbered 1,3,8,12) are all talk about passengers ('Hotel Guests' in article 8, 'cruise' passengers in article 12, 'British' passengers in article 3) and staff ('worker shortage' in article 8 and 'staff stretched' in article 1). The keywords are also indicative of the general theme of the articles in the topic. All the articles in this topic are have 'negative' sentiment as they generally focus on airlines struggling with poor service due to worker shortage, resulting in a 'negative' average sentiment for the topic.

It is apparent that the semantic cluster C0 focuses on airlines and planes with T0 focusing on struggles with staff and passenger service and T1 on the new jet orders. This gives confidence in the engine to extract meaningful topics for the article corpus for a given semantic cluster.

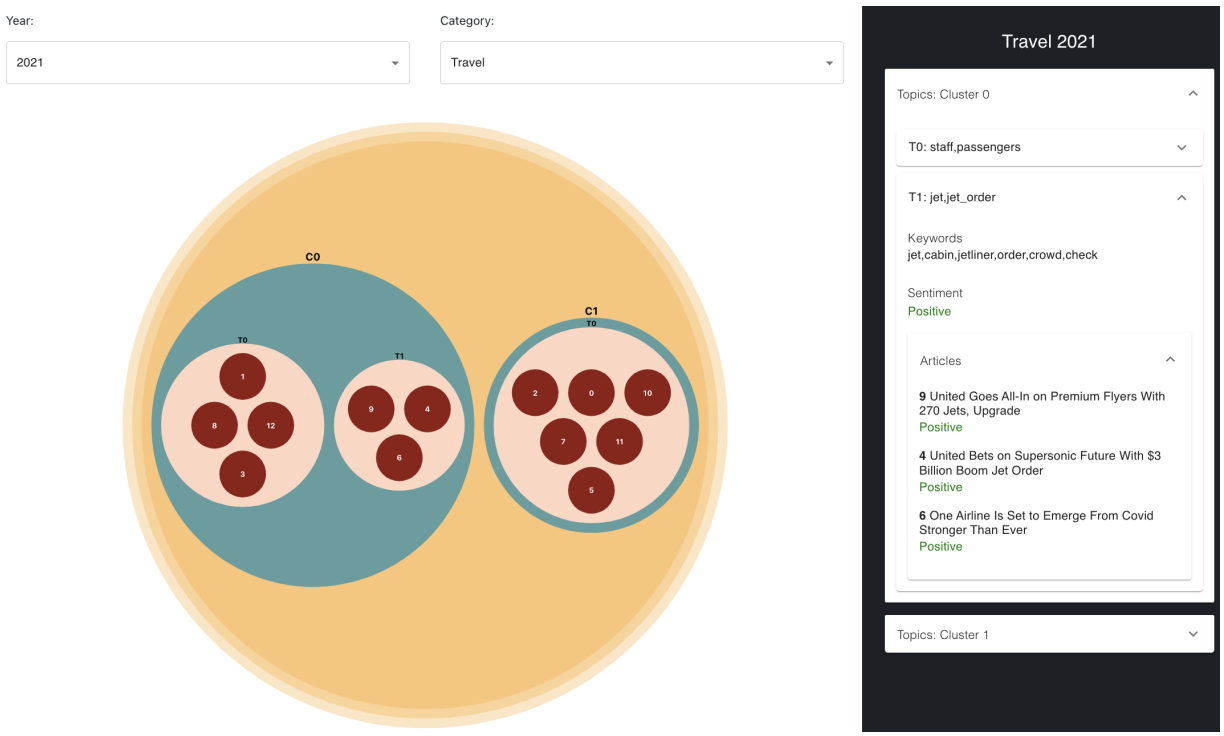

FIGURE 4.9

Figure 4.9, on the other hand, shows the other topic (T1) for the in the same semantic cluster (C0) with the inferred topic name as 'jet, jet_order' with the topic keywords: 'jet', 'cabin', 'order', 'jetliner', 'crowd' and 'check'. This topic differs from the previous topic as this focuses the positive developments made by airlines by investing into new planes and jets. All

articles in this topic (numbered 9,4,6) have a 'positive' sentiment resulting in overall average 'positive' sentiment for this topic.

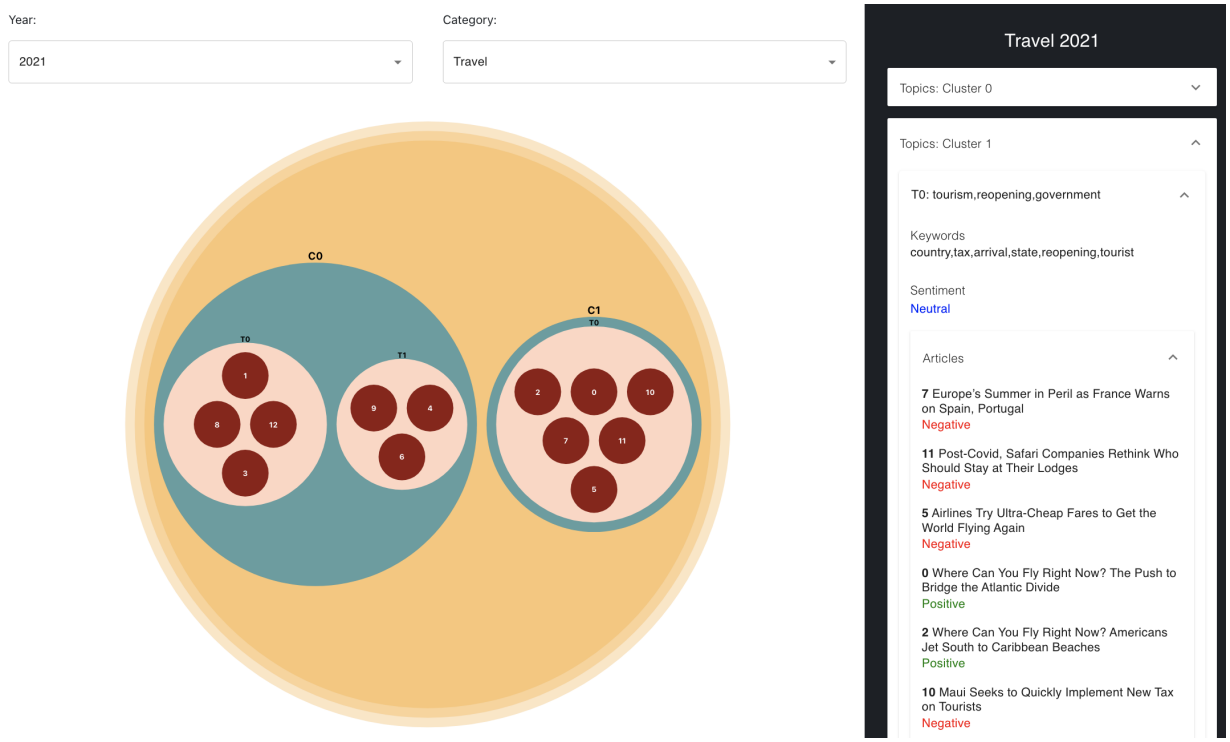

FIGURE 4.10

Figure 4.10 displays the information about the semantic cluster (C1) which has one topic with the inferred topic name 'tourism, reopening, government' and keywords: 'country', 'tax', 'arrival', 'state', 'reopening' and 'tourist'. Unlike Cluster 0 (C0) this cluster (and topic) focuses on the tourism. We can see the divided approach in the articles which talk about implementing tourist taxes (article 10) and screening guests post-Covid (article 11) as well as airlines adopting cheaper fares and pushing travel (articles 0,2 and 5). This divided approach in post-Covid travel is reflected in the sentiments of these articles, resulting in a 'Neutral' average sentiment.

Therefore, based on the Figures 4.8–4.10, we get confidence in the Topic Extraction Engine's semantic clustering, as we can see the different clusters extracted (C0: airlines and C1:tourism) and the distinct topics in these clusters, which provide an insight into the corresponding articles contributing to these topics as well as the associated sentiment regarding that topic.

Limitations

6: Topic Engine: Write about limitations

5 Semantic Triple Extraction Engine

This section focuses on the Semantic Triple Extraction Engine, focusing on triple representation and the different approaches of information extraction.

5.1 Motivation

The aim of this engine 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. One of the earlier approaches was to derive a co-occurrence graph of named entities of type 'GPE', 'PER', 'LOC', 'ORG' etc. (using Fine-Grained NER Section 3.3.2) for a topic. The limitation of this approach, was that while it was a good indicator of general entities in a topic, the amount of information extracted from the topic article corpus was not sufficient as no explicit information about *'how'* the entities were related to one another was inferred. 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 of type `subject-predicate-object`. The triple serves as a minimal representation for information in an article without losing the context. 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. These triples would then be displayed as a knowledge graph (Refer to Section 2.8.1) for each topic in a semantic cluster.

5.2 Key decisions

In an attempt to capture meaningful nodes (subjects and objects) and relations (predicate), the Semantic Triple Extraction Engine uses phrases instead of words. Figure 5.1 shows the 3 main decisions made for the representation of the triples.

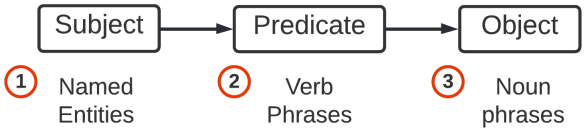

FIGURE 5.1 Decisions for Extracting Semantic Triples

1. **Named Entities as Subjects:** Given that the end goal is to build a knowledge graph (KG) from these triples, there need to be common 'subject' nodes in order to infer information about these entities from the articles, link them to other entities in the graph. For example, if the subject requirement is changed from named entities to simply noun phrases, the engine might extract two triples: one with subject 'British Airways' and another with 'British Airways spokesperson'. This is not ideal as, although both triples provide information about the named entity 'British Airways', they will be displayed as discrete triples. Therefore, using 'named entities' as subject grounds the KG to a set of named entities, for which information (semantic triples) can be extracted.
2. **Verb Phrases as Predicate:** Based on the success of previous studies [44], the engine relies on the verb-based approach for predicate extraction. It aims to extract a single relation embedded in a sentence that consists of a verb phrase sandwiched between two entities of interest. The motivation for using verb phrases as predicate is that it exploits the structured grammar present in news articles where generally the subject is usually a person, organisation, place or thing (hence, the use of named entities) and the predicate indicates what the subject is or does, which often involves a 'root verb'. These types of relations subject-verb-object are called SVO triples.
3. **Noun Phrases as Objects:** Finally, having established SVO semantic triples as the output for this engine, the remaining step is to extract the object phrases. As the SVO triples follow a pattern of noun (phrase)-verb-noun (phrase), the object of the triple relies on extracting the noun chunk (using POS tagging) after the verb as the object.

5.3 Implementation

The section highlights the process of extracting the subject, relation, object phrases from the articles in a topic in order for the knowledge graph.

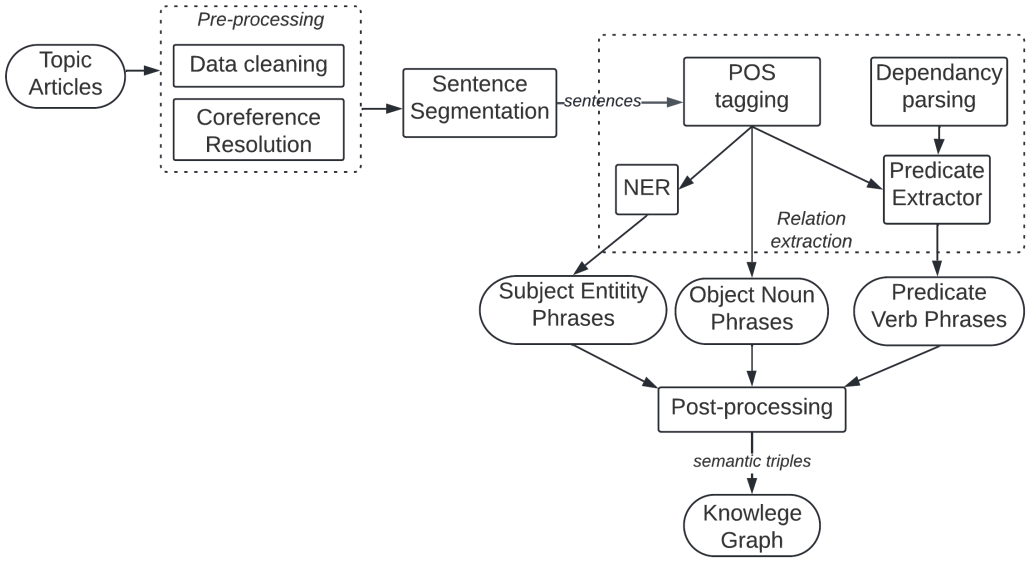

FIGURE 5.2 Overview of Semantic Triple Extraction

Processing Articles

As seen in Figure 5.2, before any information extraction, we need to ensure that the data is correctly preprocessed. For this, the coreference resolved, cleaned articles from Sections 4.1.1–4.1.2 are used. This ensures that all pronouns corresponding to the named entities are resolved, allowing more triples to be extracted from the article corpus with named entities as the 'subject'. Extracting triples on a 'cleaned' article corpus gets rid of unnecessary text, including stopwords, allowing the model to focus on 'semantically rich' words which provide more context in a smaller set of words. Each topic article then undergoes sentence segmentation (via spaCy `SentenceSplitter`) to obtain a list of sentences from which triples can be extracted.

Extracting Predicate

As discussed in Section 5.2, the predicate phrases would be extracted as verb phrases. This is done by regex matching of part-of-speech (POS) tags (refer to Appendix Table A.1) as seen below.

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

The above regex patterns do not limit the verb phrase to the 'VERB' POS tag, but instead account for other relevant information such as auxiliary verbs ('AUX'), adverbs ('ADV') and adposition ('ADP') to be pulled from the sentences as candidate verb phrases in order to retain semantic context. These are then filtered to obtain the phrases where the 'root' of the sentence is present. This is done using dependency parsing through the spaCy library. Generally, in dependency-based grammars, all words or tokens, in a sentence, barring one, are dependent 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, the root verb is derived from the dependency tree and all verb phrases containing the root verb are retained. Given the regex pattern matching criteria discussed above, each sentence might result in 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 5.3. In order to extract the most informative coherent relations, the longest verb phrase is selected.

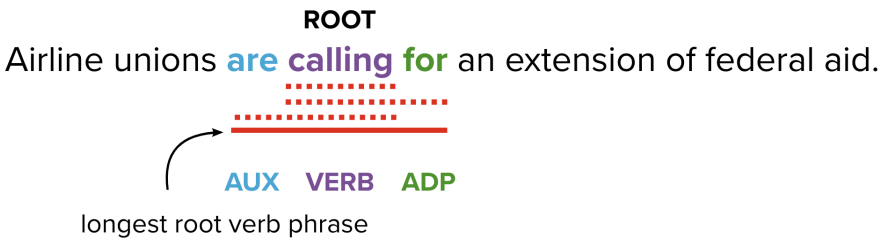

FIGURE 5.3 Verb phrases extracted from an example sentence from article corpora

Extracting Subject and Object

As discussed in Section 5.2, the 'subject' and 'object' are noun phrases (NP), particularly entity phrases for 'subject'. Therefore, in order to derive these, the 'noun chunks' are extracted from the sentence using POS tagging and chunking from the spaCy pipeline. These 'noun chunks' are base noun phrases with no nested NP and relative clauses [25] and are returned as a list of `Spans` [25], which are essentially slices of the sentence and contain information such as the 'start' and 'end' position of the sentence 'chunk' (phrase). Algorithm 4 outlines the process of extracting the subject and object phrases in the semantic triple.

Algorithm 4 Outline of Triple Extraction Procedure

function GETSUBJECTANDAUGMENTEDPREDICATE($verbPredicate, validEnts, nounPhrases$)
  for all $np \in nounPhrases$ **do**
    if $np.start < verbPredicate.start$ **then**
      $ents \leftarrow n.containsAny(validEnts)$ ▷ all entities in np
      if $ents.length == 0$ **then**
        $continue$
      $subjectEnt \leftarrow ents.sort(key = lambda\ e : e.start)[0]$
      $remnantNp \leftarrow np.partition(subjectEnt).last$ ▷ remnant phrase right of subject
      $augmentedPred = remnantNp + verbPredicate$
      $break$
  return $subjectEnt, augmentedPred$
function GETOBJECT($verbPredicate, nounPhrases$)
  for all $np \in nounPhrases$ **do**
    if $np.start > verbPredicate.start$ **then**
      $object \leftarrow np$
      $break$
  return $object$

Subject Entity and Augmented Predicate

For the subject phrase, the noun phrases (NP) that occur before (to the left of) the verb phrase ('predicate') as well as contain any of the selected set of named entities, `valid_entities`, are chosen as `subject candidates`. These `valid_entities` are derived by performing Named Entity Recognition (NER) on the sentence using the Fine-Grained NER model as detailed in Algorithm 1. As mentioned previously in Section 3.3.2, the model returns 16 semantic types (refer to Table C.1). Some of these entity types can result in unnecessary and irrelevant triples with, for instance, ('two', `'CARDINAL'`) as the subject node. Therefore, in a slight adjustment to Algorithm 1, a set of `ignore_entity_types`, containing `'DATE'`, `'TIME'`, `'CARDINAL'`, `'PERCENT'` and `'QUANTITY'`, is passed to omit extracted entities of these types. This does not mean that information about these entities is completely ignored as they are likely to be extracted as object phrases when in reference to a 'valid' subject named entity.

Of the `subject candidates`, the first occurring (based on position of `Span`) is selected as the subject phrase. As mentioned Section 5.2, the aim is to have just the named entity as the 'subject' to facilitate connectivity of triples in the KG. However, we do not lose the other

semantic information in the subject phrase. Therefore, the subject phrase is split into the entity and the remnant phrase *after* the entity, which is then prepended to the predicate. This way the information conveyed by the noun phrase is conserved, but the subject nodes are maintained as an entity. For example, for the triples: (‘Transat investors’, ‘choose receiving’, ‘cash payment’) and (‘Transat’, ‘to become part of’, ‘Air Canada’), splitting the 1^{st} triple’s subject to get ‘Transat’ and ‘investors’ and augmenting the predicate with the latter to get the updated triple: (‘Transat’, ‘investors choose receiving’, ‘cash payment’). Figure 5.4 and Figure 5.5 show the KG snippet for these triples before and after splitting the subject phrase respectively.

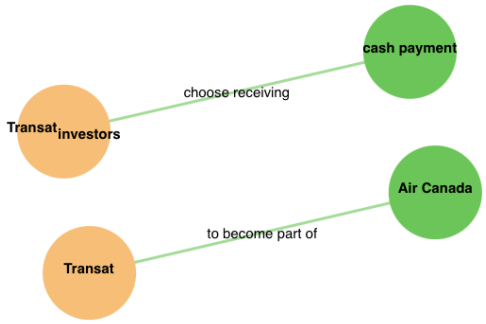

FIGURE 5.4 KG snippet before predicate augmentation

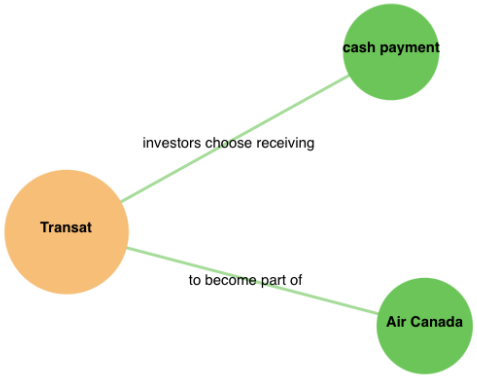

FIGURE 5.5 KG snippet after predicate augmentation

Get object phrase

Getting the object is fairly simple as shown in Algorithm 4 and involves getting the first (closest in position) ‘noun phrase’ after the ‘predicate’ (the longest root verb phrase).

Post-processing

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. Furthermore, in order to ensure conciseness of the triples and eliminate redundancy, any common substring (likely to occur from noun phrase chunking and verb phrase regex matching) is eliminated by calculating the longest common substring containing complete words between the subject and object as well as (augmented) predicate and object and removing it from the object phrase.

5.4 Results and Discussion

This advantage of using the approach discussed in Section 5.3 to extract the semantic triples is that it avoids the stringent dependency of the ‘subject’ and ‘object’ phrases on their actual

dependency tags in the dependency grammar (See Section 2.2), instead using dependency parsing only to extract the root verb in the sentence, and getting the subject and object based on the position of 'noun chunks' before and after the root verb. An alternate approach was tried relying on the dependency grammar structure to find the subject and object does not always yield great results as in a given sentence, the structure of the sentence heavily influences the information extracted. For the knowledge graph, ultimately, the relations extracted need to focus around different named entities. This means that more often than not, relevant information about entities is lost due to the 'named entity' not necessarily having one of `'nsubj'`, `'csubj'`, `'nsubjpass'`, `'csubjpass'` as the dependency tag and alternatively having `'obj'`, `'dobj'` etc. as the dependency tag (See Appendix Table B.1). The dependency sentence structure is commonly of type 'subj'-'verb'-'obj' or 'obj'-'verb'-'subj. This approach accommodates both as it only enforces that the semantic triple 'subject' is a named entity and can have either 'subj' (or related?) or 'obj' (or related?) as the dependency tag.

Knowledge graph of Semantic Triples

After extracting the qualifying semantic triples for each topic (in cluster), a Knowledge Graph of these triples is generated by the visualisation tool. Figure 5.6 shows the KG for Cluster 1 in Travel 2021 (See Figure 4.8) which has a single topic 'tourism, reopening, government'.

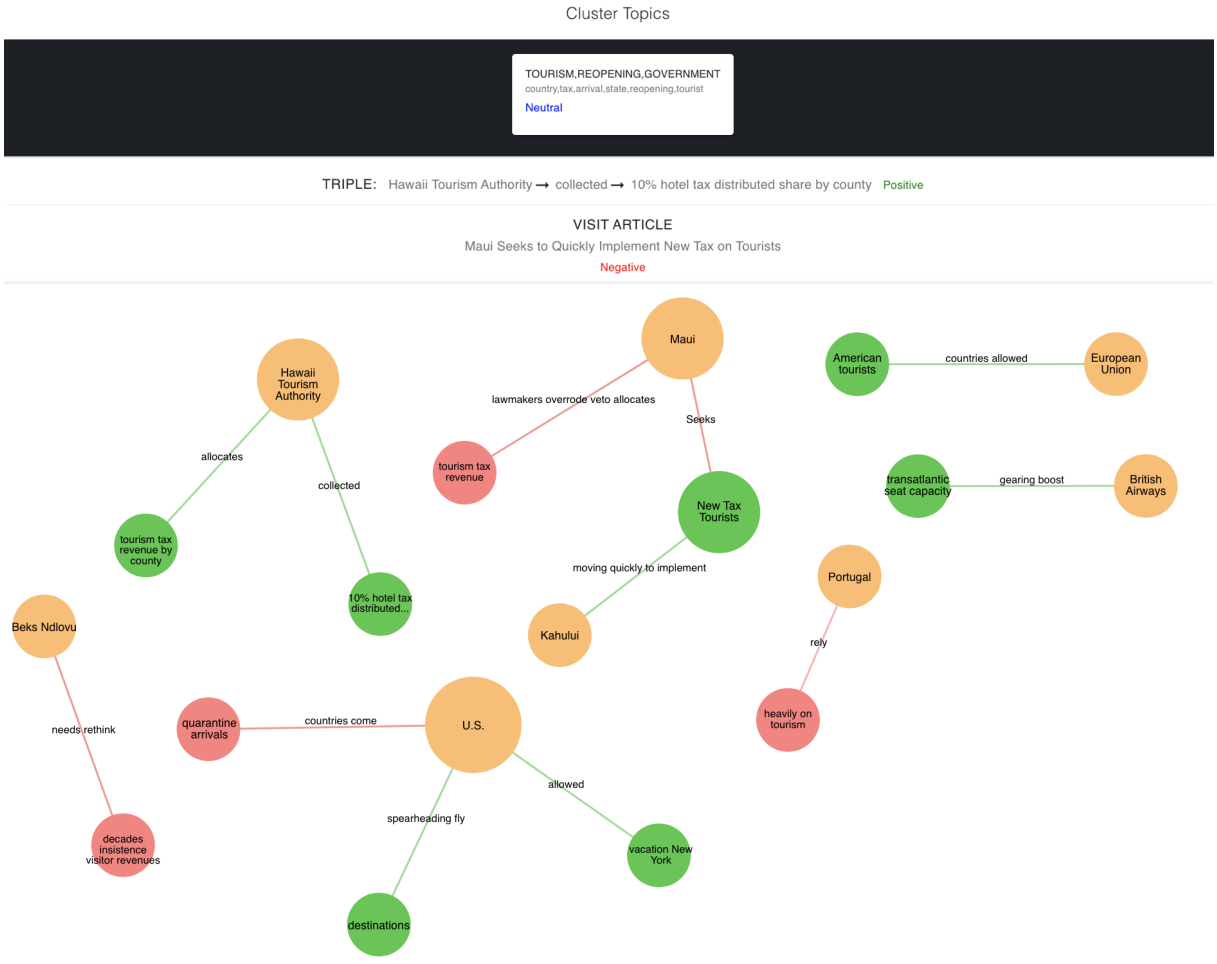

FIGURE 5.6 Knowledge Graph for topic 'tourism, reopening, government' in Cluster 1 in Travel 2021

Figure 5.6 shows the Knowledge Graph generated by the visualisation tool for the Cluster 1 (C1) which has a single topic 'tourism, reopening, government'. The 'subject' named entities are in orange, the 'object' noun phrases are in green (indicating 'positive' sentiment) or red (indicating 'negative' sentiment) and the lines joining these represent the predicate. Clicking on the predicate gives shows the entire semantic triple (and the corresponding 'triple sentiment') as well as the article (and the article 'sentiment') from which the triple was extracted. The size of a node is determined by the degree of edges incident upon the node which makes it easier to identify the key entities in a topic. For example, we see that the 'Hawaii Tourism Authority', 'U.S.' and 'Maui' have more triples associated with them as opposed to 'Portugal', making the former nodes bigger and therefore, stand out more visually. Additionally, we can see that the interconnectivity of the nodes with the example of 'Maui' and 'Kahului' which both share the 'object' node 'New Tax Tourists'. These gives us the insight that both 'Maui' and 'Kahului' which are islands in Hawaii have implemented a new tourist tax, which is further implied at by the triple: 'Hawaii Tourism Authority' → 'allocates' → 'tourism tax revenue by county'.

As mentioned prior, the colour of nodes is a representation of the sentiment of the triple. This may not always coincide with the sentiment of the article. The motivation for this was to see the general sentiment surrounding an entity which can come from the entity being mentioned in different contexts in multiple articles. For instance, the 'European Union has a 'positive' sentiment associated with it as the information we have on it is that EU countries allowed 'American tourists'. An important thing to note is that the sentiment is quite context dependent, so for example, going back to the triples 'Maui' → 'seeks' → 'new tax tourists' and 'Kahului' → 'moving quickly to implement' → 'new tax tourists', the former has a 'negative' (note the red 'predicate' line joining the nodes) sentiment while the latter has a 'positive' sentiment associated to it. This is because 'seeking tax' presents negative connotations whilst 'moving quickly to implement' has connotations of efficiency and vigour which are positive.

Limitations

One of the main limitations of our implementation of semantic triple extraction is that, given it is a verb-based approach, it extracts a single relation embedded in a sentence composed of a (root) verb phrase sandwiched between two entities of interest. Given the nature of news articles, the sentence structure is relatively complex with several subjects and objects in a single subject, often trying the maximise information extraction from a single sentence can lead to redundancies and incoherent semantic triples. Therefore, the focus of this model was that it aims to extract a single relation in a sentence provided the sentence mentioned a named entity.

Another limitation of the engine comes from the performance of the spaCy library used for sentence segmentation, POS tagging, dependency parsing as well as the state-of-the-art AllenNLP models: SpanBERT for coreference resolution, Fine-grained Named Entity Recognition and RoBERTa Stanford Sentiment Treebank (See Section 3.3.2) used for resolving coreferences in the article corpus in preprocessing, extracting the named entities from the sentences for the 'subject' node and performing sentiment analysis on the articles, topics and triples respectively. The performance of these models have a huge effect in the performance of the Relation Extraction Engine.

6 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 5.2.
4. Evaluate the legibility of the results based on user feedback.

Change this based on actual eval done

6.1 Topic Extraction Engine

6.1.1 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 4.1, 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 6.1 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 6.1 shows that omitting these pre-processing techniques deteriorates the quality of clustering, by resulting in poor silhouette scores which indicate overlapping clusters.

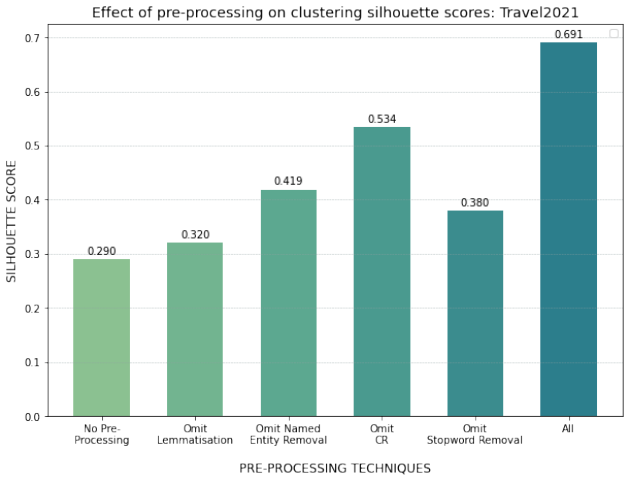

FIGURE 6.1 -

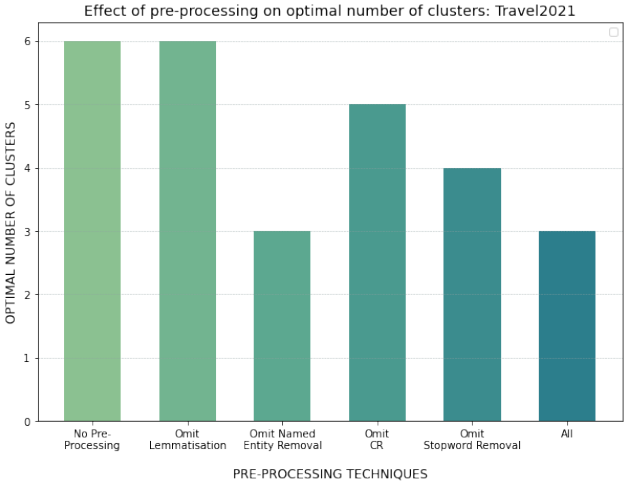

FIGURE 6.2 -

Another interesting takeaway is that post pre-processing considering the techniques in questions, usually results in lower number of clusters as seen in Figure 6.2. 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 6.1–6.2 is that performing the pre-processing techniques mentioned results in better silhouette scores with lower number of clusters, which indicates better clustering.

6.1.2 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 6.3–6.6 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 6.4, 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. The same is true for Figures 6.5–6.6 which show an 8.3% and 16.3% improvement respectively for clustering articles based on noun only article corpus.

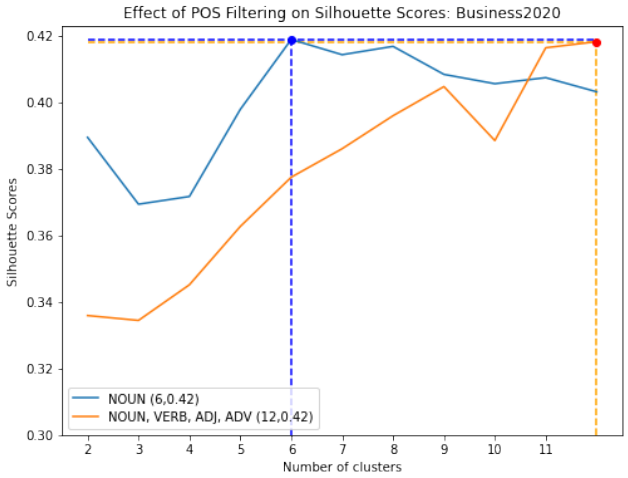

FIGURE 6.3 -

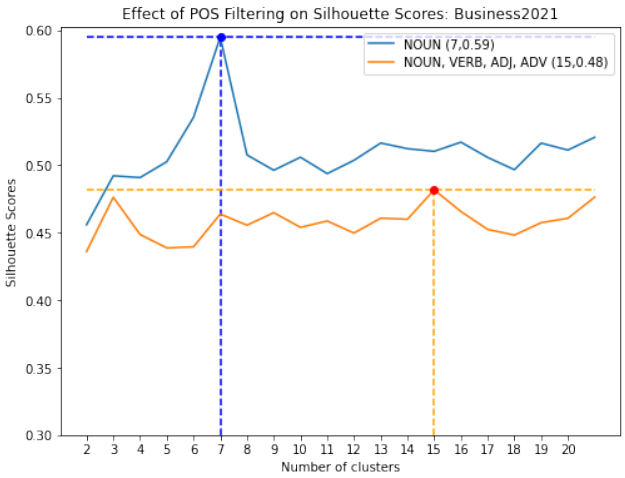

FIGURE 6.4 -

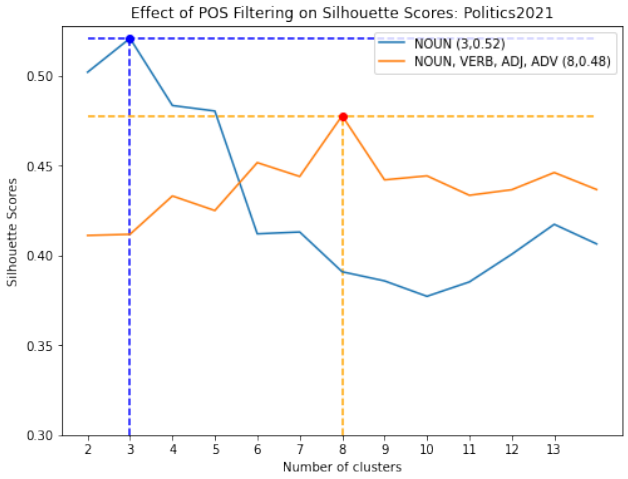

FIGURE 6.5 -

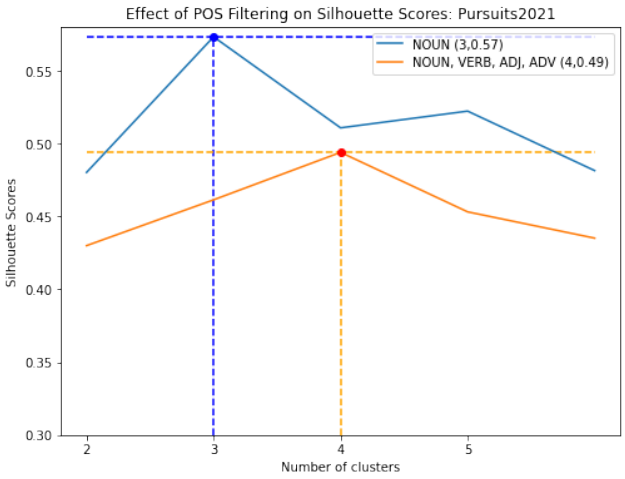

FIGURE 6.6 -

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 6.3 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.

6.1.3 Effect of different word embedding techniques on clustering

FIGURE 6.7 Effect of using TF-IDF, Word2Vec and GloVe as word embedding model on clustering silhouette scores

As discussed in Section 4.2.1, in order to cluster the articles, each article's corresponding list of filtered tokens is vectorised. Figure 6.7, 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 6.7, 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 6.1 E-

Table 6.1 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.

6.1.4 Effect of pre-processing on topic modelling

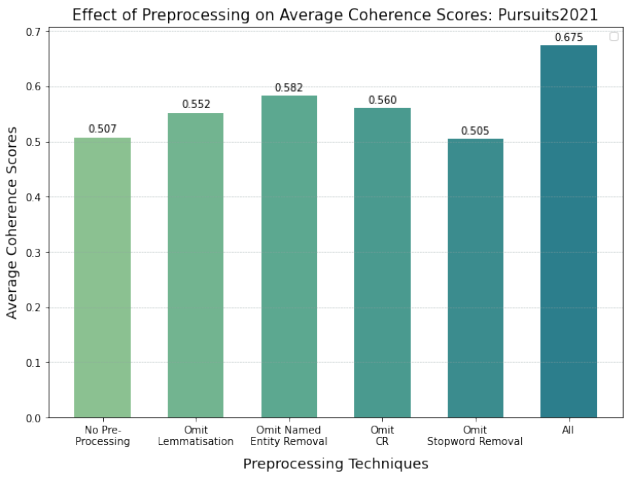

FIGURE 6.8 -

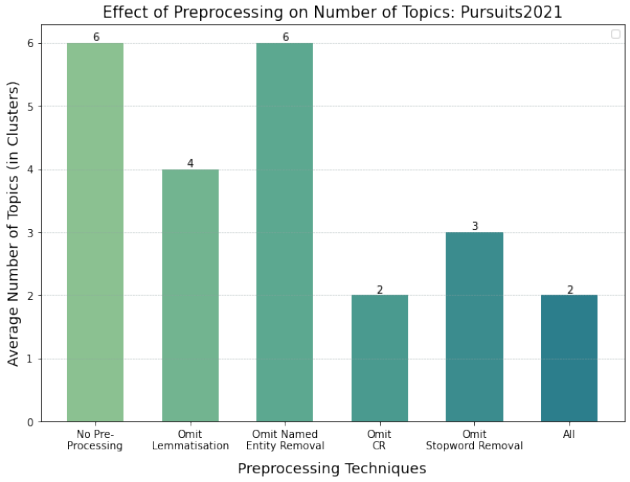

FIGURE 6.9 -

Section 6.1.1 highlights the effect of the different pre-processing techniques on the silhouette scores for semantic clustering. In a similar fashion, Figure 6.8 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 6.1.1, 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 6.8 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'.

6.1.5 Effect of POS on topic modelling

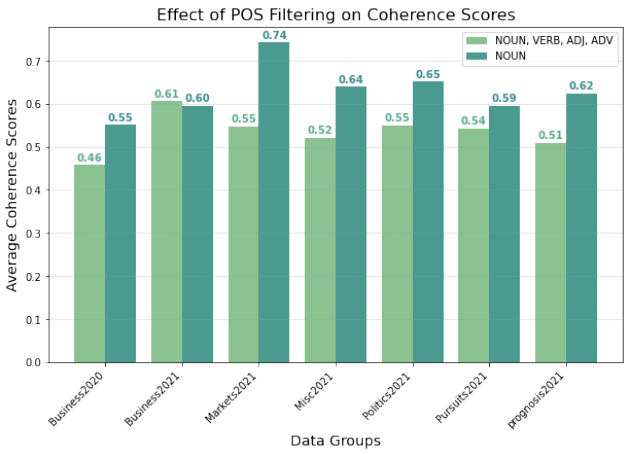

FIGURE 6.10 pos

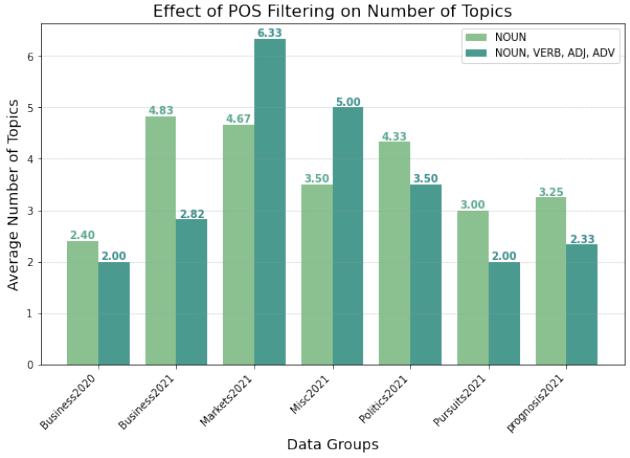

FIGURE 6.11 -

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 6.10, 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.

6.2 Time Efficiency

6.2.1 Comparing Runtimes in Various Environments

Time efficiency (in seconds)

Data group	CPU	Multiprocessing CPU	GPU
Business 2020	351.41	352.51	39.47
Business 2021	1713.60	1431.80	138.47
Markets 2021	839.75	1252.64	74.18
Misc 2021	215.41	337.49	18.60
Politics 2021	856.42	1267.96	64.87
Pursuits 2021	178.54	377.49	15.00
Travel 2021	169.52	452.80	12.92
Prognosis 2021	583.33	888.27	51.63
Total time (s)	4907.98	1920.16	414.60
Total time (min)	81.80	32.10	6.90

TABLE 6.2 E-

Table 6.2 shows the different runtimes for the pipeline when running on a CPU, with multiprocessing (on CPU) and GPU. Ideally, the pipeline is meant to be run on the GPU as it results in significant improvement in computation time, with the whole process of running the pipeline for all the input groups from the dataloader taking only 6.9 minutes in comparison. This is 11.8 times faster than running it on the CPU and 4.6 times faster than running it on the CPU with multiprocessing.

The reason for the high computation time is due to the expensive calls to the AllenNLP models, especially the Fine-Grained NER and SpanBERT Coreference and RoBERTa Sentiment models (see Section 3.3.2), which unlike spaCy, are not built for production use and therefore not as optimised for time. However, given that they are state-of-the-art models with high accuracy, their use is extremely beneficial to the project, motivating the use of multiprocessing and pipelining (see Figure 3.2) in early stages of development to allow for feasible CPU computation.

Processes are traditionally constrained to just having access to their own process memory area, but shared memory allows data structures to be shared between processes. In this case, models are stored in shared memory in order to provide centralised access to all running processes and avoid redundant copies of models for each process. This is crucial as the models in question are quite large and therefore, RAM-intensive and so having multiple copies of the models can result in the machine running out memory.

This allowed running the tool on the 8 different data groups from the dataloader simultaneously. As seen in Table 6.2, multiprocessing results in 2.55 times lower computational time compared to running with single process CPU. This is because the idle time is significantly reduced by concurrent processes during the expensive calls for coreference resolution, sentiment analysis and NER.

6.2.2 Effect of different word embedding models on GPU runtime

GPU runtime (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 6.3 Time taken to run the pipeline (on GPU) with different word embedding models

Table 6.1 highlights the time taken for running the pipeline shown in Figure 3.2 on the Tesla T4 GPU for all input data groups (from the dataloader) using different word embedding models. As established in Section 6.1.3, the Topic Extraction Engine uses `GloVe` as the word embedding model as it gives a significant improvement in the semantic clustering silhouette scores. In addition to that it also results in more efficient (faster) computation when running the pipeline for all input data with a total runtime of 414.60 seconds, which is 31.15% lower than using `Word2Vec`. The difference in runtimes for when using `TF-IDF` and `GloVe` is trivial, however, given the 33.13% improvement in silhouette score of `GloVe` over `TF-IDF`, `GloVe` proves to be the best approach in terms of time efficiency and optimal clustering.

6.3 User Evaluation

In an attempt to gain a greater understanding of the performance of our tool, user testing was conducted with a focus group of 7 people, from DeepSearchLabs to obtain qualitative user evaluation on the utility of the visualisation tool in displaying results from the Topic Extraction Engine and Semantic Triple Extraction Engine.

7 Conclusion

Todo list

7.1 Summary of achievement

7.2 Future Work

Topic Bert

A POS Tag Definitions

POS Tag	Meaning	Examples
ADJ	adjective	big, pink, first
ADP	adposition	in, to, during
ADV	adverb	very, well, tomorrow
AUX	auxiliary verb	has (done), is (doing), will (do)
CONJ	coordinating conjunction	and, or, but
DET	determiner	a, an, the
INTJ	interjection	ouch, bravo, hello
NOUN	noun	tree, air, beauty
NUM	numeral	seventy-seven, 12, IV
PART	particle	’s, not
PRON	pronoun	you, he, she
PROPN	proper noun	Mary, Anji, London
PUNCT	punctuation	. ! ?
SCONJ	subordinating conjunction	if, while, that
SYM	symbol	$, %, :
VERB	verb	jump, run, dance
X	other	scsd, aosa, oooqas

TABLE A.1 Universal POS tag set comprising core part-of-speech categories [45]

B Dependency Set Labels

Dependency Tag	Meaning
acl	clausal modifier of noun (adjectival clause)
advmod	adverbial modifier
amod	adjectival modifier
appos	appositional modifier
aux	auxiliary
case	case marking
cc	coordinating conjunction
ccomp	clausal complement
clf	classifier
compound	compound
conj	conjunct
cop	copula
csubj	clausal subject
dep	unspecified dependency
discourse	discourse element
dislocated	dislocated elements
expl	expletive
fixed	fixed multiword expression
iobj	indirect object
nmod	nominal modifier
nsubj	nominal subject
nummod	numeric modifier
obj	object
obl	oblique nominal
punct	punctuation
reparandum	overridden disfluency
root	root
vocative	vocative
xcomp	open clausal complement

TABLE B.1 Universal Dependency Tag set [46]

C NER Semantic Types

Dependency Tag	Meaning
CARDINAL	
DATE	
FAC	
GPE	
LAW	
LOC	
MISC	
MONEY	
NORP	
ORG	
PERCENT	
PERSON	
QUANTITY	
TIME	

TABLE C.1 Named Entity Types extracted by Fine-Grained NER [29]

7: Find all semantic types

Bibliography

[1] Xia C. Extracting global entities information from news. University of California, Los Angeles; 2019.

[2] Sarawagi S. Information extraction. Now Publishers Inc; 2008.

[3] Krstajić M, Mansmann F, Stoffel A, Atkinson M, Keim DA. Processing online news streams for large-scale semantic analysis. In: 2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010); 2010. p. 215–220.

[4] Costantino M, Morgan RG, Collingham RJ, Carigliano R. Natural language processing and information extraction: Qualitative analysis of financial news articles. In: Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr). IEEE; 1997. p. 116–122.

[5] Hogenboom A, Hogenboom F, Frasincar F, Schouten K, Van Der Meer O. Semantics-based information extraction for detecting economic events. Multimedia Tools and Applications. 2013;64(1):27–52.

[6] Faiz R. Identifying relevant sentences in news articles for event information extraction. International Journal of Computer Processing of Oriental Languages. 2006;19(01):1–19.

[7] Kogilavani SV, Kanimozhiselvi C, Malliga S. Summary generation approaches based on semantic analysis for news documents. Journal of Information Science. 2016;42(4):465–476.

[8] Wu T, Wang H, Li C, Qi G, Niu X, Wang M, et al. Knowledge graph construction from multiple online encyclopedias. World Wide Web. 2020;23(5):2671–2698.

[9] Tseng YH, Lee LH, Lin SY, Liao BS, Liu MJ, Chen HH, et al. Chinese open relation extraction for knowledge acquisition. In: Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers; 2014. p. 12–16.

[10] Riedl M, Biemann C. TopicTiling: a text segmentation algorithm based on LDA. In: Proceedings of ACL 2012 Student Research Workshop; 2012. p. 37–42.

[11] Bouras C, Tsogkas V. W-kmeans: clustering news articles using wordnet. In: International Conference on Knowledge-Based and Intelligent Information and Engineering Systems. Springer; 2010. p. 379–388.

[12] Exceptions to Copyright: Research; 2022. Available from: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/375954/Research.pdf.

[13] Kannan S, Gurusamy V, Vijayarani S, Ilamathi J, Nithya M, Kannan S, et al. Preprocessing techniques for text mining. International Journal of Computer Science & Communication Networks. 2014;5(1):7–16.

[14] Al-Moslmi T, Ocaña MG, Opdahl AL, Veres C. Named entity extraction for knowledge graphs: A literature overview. IEEE Access. 2020;8:32862–32881.

[15] Petrov S, Das D, McDonald R. A Universal Part-of-Speech Tagset. arXiv; 2011. Available from: https://arxiv.org/abs/1104.2086.

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

[17] Willett P. The Porter stemming algorithm: then and now. Program. 2006;.

[18] Yse DL. Text Normalization for Natural Language Processing (NLP); 2021. Available from: https://towardsdatascience.com/text-normalization-for-natural-language-processing-nlp-70a314bfa646.

[19] Zheng J, Chapman WW, Crowley RS, Savova GK. Coreference resolution: A review of general methodologies and applications in the clinical domain. Journal of biomedical informatics. 2011;44(6):1113–1122.

[20] Brown A. Pronunciation and phonetics: A practical guide for English language teachers. Routledge; 2014.

[21] Jurafsky D, Martin JH. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition;. Available from: https://web.stanford.edu/~jurafsky/slp3/21.pdf.

[22] Lee K, He L, Zettlemoyer L. Higher-order coreference resolution with coarse-to-fine inference. arXiv preprint arXiv:180405392. 2018;.

[23] Lee K, He L, Lewis M, Zettlemoyer L. End-to-end neural coreference resolution. arXiv preprint arXiv:170707045. 2017;.

[24] Joshi M, Chen D, Liu Y, Weld DS, Zettlemoyer L, Levy O. SpanBERT: Improving Pre-training by Representing and Predicting Spans. arXiv; 2019. Available from: https://arxiv.org/abs/1907.10529.

[25] spaCy 101: everything you need to know; 2022. Available from: https://spacy.io/usage/spacy-101.

[26] Natural Language Processing in Python: NLTK vs. spaCy; 2021. Available from: https://www.thedataincubator.com/blog/2016/04/27/nltk-vs-spacy-natural-language-processing-in-python/.

[27] ; 2022. Available from: https://allenai.org/allennlp/software/allennlp-library.

[28] Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, et al.. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv; 2019. Available from: https://arxiv.org/abs/1907.11692.

[29] Lample G, Ballesteros M, Subramanian S, Kawakami K, Dyer C. Neural Architectures for Named Entity Recognition. arXiv; 2016. Available from: https://arxiv.org/abs/1603.01360.

[30] cloud application platform: Heroku; 2022. Available from: https://www.heroku.com/.

[31] CHURCH KW. Word2Vec. Natural Language Engineering. 2017;23(1):155–162.

[32] Pennington J, Socher R, Manning CD. Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP); 2014. p. 1532–1543.

[33] Kriegel HP, Kröger P, Zimek A. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. Acm transactions on knowledge discovery from data (tkdd). 2009;3(1):1–58.

[34] Beyer K, Goldstein J, Ramakrishnan R, Shaft U. When is "nearest neighbor" meaningful? In: International conference on database theory. Springer; 1999. p. 217–235.

[35] Muflikhah L, Baharudin B. Document Clustering Using Concept Space and Cosine Similarity Measurement; 2009.

[36] Saputra DM, Saputra D, Oswari LD. 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. .

[37] Kodinariya TM, Makwana PR. Review on determining number of Cluster in K-Means Clustering. International Journal. 2013;1(6):90–95.

[38] Martin F, Johnson M. More efficient topic modelling through a noun only approach. In: Proceedings of the Australasian Language Technology Association Workshop 2015; 2015. p. 111–115.

[39] Wallach HM. Topic modeling: beyond bag-of-words. In: Proceedings of the 23rd international conference on Machine learning; 2006. p. 977–984.

[40] Blei DM, Lafferty JD. Topic models. In: Text mining. Chapman and Hall/CRC; 2009. p. 101–124.

[41] Syed S, Spruit M. 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. p. 165–174.

[42] Syed S, Spruit M. 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. p. 165–174.

[43] Röder M, Both A, Hinneburg A. Exploring the space of topic coherence measures. In: Proceedings of the eighth ACM international conference on Web search and data mining; 2015. p. 399–408.

[44] Hao Q, Keppens J, Rodrigues O. A verb-based algorithm for multiple-relation extraction from single sentences. In: Proceedings of the International Conference on Information and Knowledge Engineering (IKE). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp); 2017. p. 115–121.

[45] Universal POS tags; 2022. Available from: `https://universaldependencies.org/docs/u/pos/`.

[46] Universal Dependencies; 2022. Available from: `https://universaldependencies.org/u/dep/all.html`.