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 20, 2022

Abstract

In today's media climate, there is an overwhelming amount of news. Several thousand articles published daily are made available through a variety of online sources. This information overload can be difficult to navigate for the typical consumer wanting to obtain a broad understanding of a certain genre of news. This poses a need for knowledge distillation and information retrieval through semantic analysis of news corpora. Previous works have focused on isolated aspects of semantic analysis, such as temporal analysis of named entity co-occurrence, modelling topic sentiment and news summarisation through sentence relevance identification. However, very few investigations in this field offer a combined solution for concise and coherent representation of news in a particular industry.

This paper proposes an end-to-end semantic analysis pipeline that focuses on two primary aspects: topic extraction and knowledge representation through semantic triples. We present a novel topic extraction approach that combines cluster analysis and topic modelling to produce high-level semantic clusterings of news articles, from which latent topic models are derived. We explore several natural language processing techniques to establish key optimisations for evaluating and improving the 'clustering quality' and 'topic coherence' in our solution. Furthermore, we introduce a lexico-semantic approach to build a minimal representation of the information in these topics using semantic triples.

Through the cohesive integration of these components, our solution delivers a consolidated visual overview of relevant topics, entities, events, and sentiments surrounding the developments within a particular industry over a set period of time. For the scope of this project, we focus on analysing news for the airline industry.

Acknowledgements

I would like to thank my supervisor Professor William Knottenbelt for his constant guidance throughout this project and for giving me the creative freedom to guide this project. I would also like to thank the team at Deep Search Labs, in particular, Ms Maryam 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

“We are assaulted with more information than any one of us can handle”.

- Daniel Levitin, cognitive psychologist and neuroscientist

In today’s fast-growing, data-driven economy, there is an incomprehensible amount of information available at our disposal. Thousands of articles are written every day and made available through a variety of internet outlets. For the 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 for a specific time period, without having to wade through reams of superfluous articles. This prompts performing semantic analysis on the news articles to extract relevant and distilled information from the news corpora.

1.1 Motivation

Information extraction or retrieval from online news corpora is a huge area of research in the natural language processing (NLP) field. With an overwhelming amount of information available, the main task in information retrieval entails the automatic extraction of structured information such as entities, relationships between entities, etc., from unstructured sources. This is achieved 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 a text in the corpus [1] [2]. Early research in this area focused on temporal analysis of co-occurrence of named entities (e.g., persons, organisations, etc.) [2] [3], event detection [4] [5], and information distillation through news summarisation and 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 serves 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 inflexions, clausal markers, nominal case) [9] to identify the 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 a large corpus of articles and is often combined with 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 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 techniques discussed above to provide a semantic analysis engine that aims to help alleviate information overload by distilling the knowledge extracted from the news corpora (into topics and corresponding semantic triples). For the scope of this project, the news is limited to 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 to generate semantic document vectors for clustering analysis.

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, and 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 tool, i.e., semantic clusters, latent topics and triples, in a cohesive visualisation.

1.3 Contributions

This paper presents an end-to-end cohesive semantic analysis pipeline, that incorporates the different semantic analysis approaches and has the following key contributions:

1. **A novel process for topic extraction in a news corpus**: We propose our Topic Extraction Engine, which generates clusters of news articles based on the semantic similarity of their text. The generated semantic clusters are then subjected to topic modelling (through LDA) to extract latent topic models, which are optimised for coherence. For each of these topics, the engine infers a topic name, the associated articles, and the general sentiment (see Chapter 5).

2. **A domain-independent methodology for semantic triple extraction:** We propose our

Semantic Triple Extraction Engine that exploits syntactic and grammatical structure of English news articles, without any dependency on existing knowledge base and ontologies, to extract triples of type (subject $\rightarrow$ predicate $\rightarrow$ object). These are extracted for each topic in a semantic cluster to provide insight into the information and sentiment surrounding key named entities in the article corpus of a topic (see Chapter 5).

3. **A cohesive visualisation tool to display results:** A web-based application responsible for illustrating the results from the Topic Extraction Engine and Semantic Triple Extraction Engine in a coherent and cohesive visualisation by making use of circle-packing topic cluster diagrams and force-directed knowledge graphs respectively.

Such a solution 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 businesses that want to obtain an intelligent overview of key information within specific industries.

1.4 Ethical Considerations

This project focuses on the semantic analysis of news articles with the goal of knowledge distillation, which is particularly beneficial from an intelligence standpoint because it condenses enormous amounts of data to present a visual overview of an industry for different genres of news across time. Though our solution highlights recent developments and trends in a sector, it is highly unlikely that it will influence or propagate some radicalised opinion. Furthermore, given the absence of direct human involvement in our use case, there is no collection or direct use of personal data.

One key ethical aspect to consider is any bias introduced in our semantic analysis engine due to the nature of the input data, which may contain underlying bias. For example, if we use the engine with a dataset for the energy industry, which contains a large proportion of right-wing articles that mention themes like "climate change being a hoax", then this will be reflected in the output results (topics, associated sentiments and semantic triples) via the visualisation tool, indicating 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 quanlifies as unbiased, and in fact, performing any filtration of articles to "alleviate bias" would inherently introduce more bias.

Another key ethical consideration is that of using software and data with copyright licences. The usage of an online news article dataset is an important part of this study. This dataset is provided by Deep Search Labs and is scraped from online sources, particularly Bloomberg News. Using the articles (which are the Intellectual Property (IP) of the news 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 Natural Language Processing (NLP) concepts such as Coreference Resolution, Named Entity Extraction, Sentiment Analysis, Topic Modelling which are used to extract key information from the data; Word Representations; and ways to extract and visualise data though Knowledge Graphs.

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. 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 showing different aspects of text preparation is as follows:

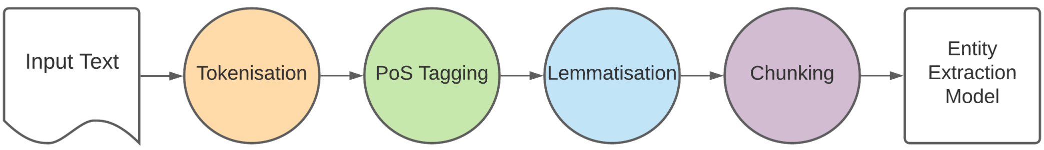


Figure 2.1: Data preprocessing pipeline

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 '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

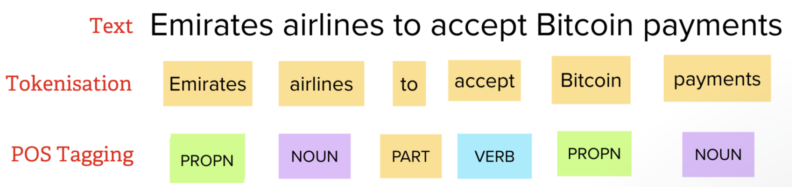


Figure 2.2: Tokenisation and POS tagging

2.1.3 Normalisation

Information retrieval focuses 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, 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 algorithms 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 remain 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 inflective variants to a 'root'

word/token [16]. Example: 'walking', 'walks', 'walked' will all reduce to 'walk'. Lemmatisation makes use of POS tags and sentence structure. This is a huge 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 extracting 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 (NP): 'The large truck', verb phrase (VP): 'is going', prepositional phrase (PP): 'under the tunnel' as shown in Figure 2.3.

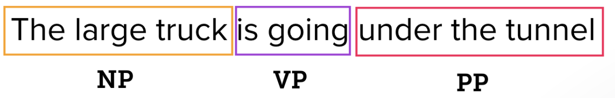


Figure 2.3: Sentence chunking

2.2 Dependency Parsing

Dependency parsing analyses the structure of a sentence based on word (token) dependencies. It uses a universal collection of tags, which is presented in Appendix Table B.1, to describe the relationship between two words, one acting as the 'head' (parent) and the other as the 'dependant' (child). A (token) word can have multiple dependents (one-to-many mapping) but can only have one head (one-to-one mapping). The 'root' word of a sentence is not the child of any other words in the sentence, and is usually a verb often known as the 'root verb'.

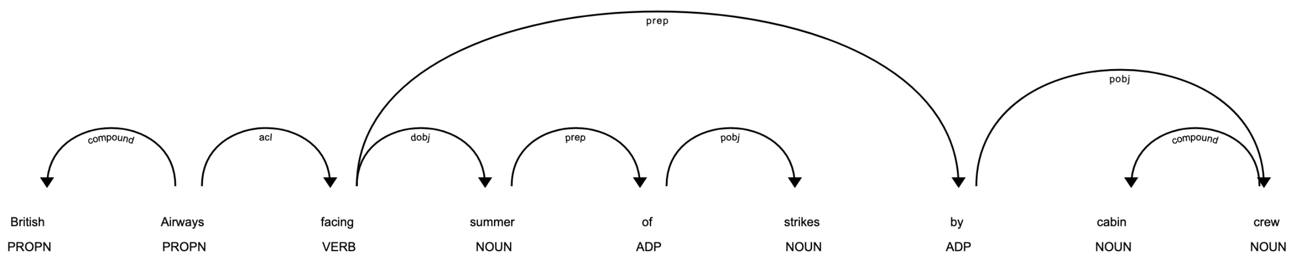


Figure 2.4: SpaCy Dependency Tree Visualisation [19]

Figure 2.4 illustrates the dependency relations and part-of-speech (POS) tags for each word (or 'token') in the sentence.

1. 'British Airways' and 'cabin crew' are noun compounds, where all the nouns (e.g., 'British') modify the rightmost noun (e.g., proper noun: 'Airways').

2. The noun ‘summer’ is ‘direct object’ of the ‘verb phrase’ or predicate ‘facing’.
3. Nouns ‘strikes’ and ‘crew’ are ‘objects of a preposition’ or ‘pobj’ as they are the ‘head’ of noun phrases, ‘strikes’ and ‘cabin crew’ respectively, following adverbs or, in this case, adpositions (‘of’ and ‘by’ respectively).

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) [20]. It is often an important step for other high-level tasks such as 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 [21]. The goal of the coreference resolution algorithms is to find and cluster the mentions by their referent. For instance, in the sentence, “The UK said they would relax travel bans”, the mentions ‘UK’ and ‘they’ refer to the same entity: ‘UK’

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. [20].

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

1. **Anaphora coreference:** Anaphoric references refer to the previous entities (the ‘antecedent’) for their meaning. For instance, in the sentence: “The aviation industry is to find its way back in 2022”, the anaphor ‘its’ follows the expressions it refers to, the antecedent: ‘the aviation industry’.
2. **Cataphora coreference:** Cataphora references refer to entities/mentions that appear 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 [22] (discussed in detail in Section 2.5). To detect spans, there are several 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 (predetermined) K antecedents for each mention [23] 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 [23].

2.3.2 Coreference Architecture

Once the candidate spans are extracted from the text, the next step is to resolve the coreferences of mentions. The most widely used architectures are as follows:

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 co-referring based on a ranking 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, but instead 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 [22]. This approach is used by the model described in Lee 2017 [24], a model on which many recent state-of-the art models such as SpanBERT [25] are based. It uses the mention ranking architecture to determine a conditional probability distribution (as seen in (2.1)) to give a configuration with the highest likelihood of representing the correct clustering.

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

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

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

In equation (2.2), the 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 $s_c f(i, j)$, the joint coreference probability of spans i and j in document D (assuming they are both mentions) referring to the same entity [24][23].

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 several different approaches 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 stopword 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 [26].

2. **Term-Frequency-Inverse Document Frequency (TF-IDF):** TF-IDF is a numerical statistic that indicates the significance of a word in a corpus (collection of documents). It is frequently used as a weighting factor in conjunction with the bag-of-words approach to represent document embeddings. The TF-IDF value is proportional to term frequency (TF) which represents the number of times a word appears in the document and is countered by the number of documents in the corpus that contain this word (IDF), thereby compensating for the fact that some words appear more frequently in general, and their high frequency is not distinct to a document [27].

3. **Word2Vec:** Word2Vec algorithms provide a much better way of word representation (or embedding) 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) [28]. They also exploit the 'locality hypothesis' which centres around the notion that words that appear together or are close to identical will be spatially close [26].

   For simplicity's sake, let's say the word 'know' is represented by a 4-dimensional binary vector [_,_,1,_], with 1 in the third dimension then vectors for words that are variants of the same underlying root word such as 'knew', 'known' etc. will also have 1 in the third dimension, and those are not (say, 'Monday') will not. Similarly, there can be a feature (dimension) (say, the 4_{th} dimension) representing a word type or category such as airlines and so, for instance, 'Emirates', 'Etihad' and 'British Airways' should also have similar vector representations with the 4_{th} dimension (feature) having a value of 1, thereby assigning them to the same group (cluster) [29].

   These Word2Vec embeddings are commonly used in the Neural Network models: Continuous Skip-Gram and Continuous Bag-of-Words (CBoW) [28].

4. **GloVe:** Another popular approach is using GloVe or Global Vectors for word representation. Unlike word2vec which leverages co-occurrence of neighbouring words within a local context, the GloVe unsupervised model focuses on global statistics of word co-occurrences over the whole corpus. Like word2vec, it also relies on the mapping words in vector subspaces using semantic similarity of words [30].

5. **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 of a word, called "contextual word vectors" which encapsulate the meaning of a word in a particular context. For instance, in the context of the sentence "I ate a banana split", 'split', is associated with food [29] and in "the ice cracked and split", 'split' is a verb indicating breakage of ice.

   This approach makes use of 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 [26]. One of the most popular approaches for this is ELMo (Embeddings from Language Models) which is discussed in Section 2.4.2.

2.4.2 Embeddings from Language Models (ELMo)

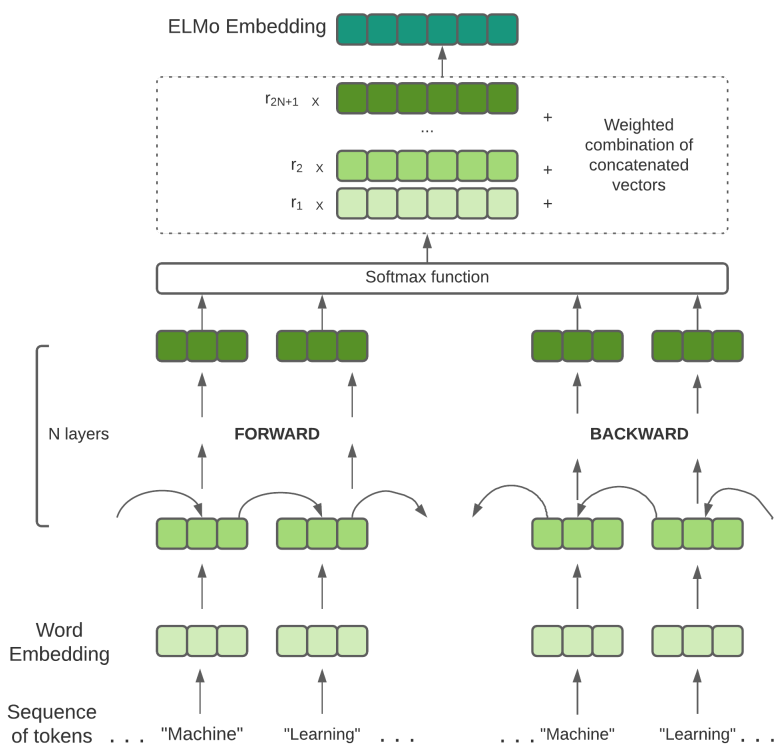


Figure 2.5: Embeddings from Language Models (ELMo) Context Representation

The outline of the ELMo model is as follows: [31]

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 [26].

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

   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 [26].
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 [26].
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 [31].
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 [26].

2.5 Named Entity Recognition (NER)

Named entity recognition (NER) is a process which extracts and classifies named entities of certain (pre-defined) types such as 'PER' (person), 'GPE' (geo-political entities), 'ORG' (organisation) from an unstructured text [14]Refer to Appendix Table C.1 for the full list of entity types used by AllenNLP and spaCy.

First, it identifies the 'names' of the entities from the pre-processed tokens commonly done by BILOU tagging (See Table 2.1) and POS tagging. It then performs sequence labelling (assigning labels/tags to each element of an input sequence) and classifies entities into the predefined types such person, location, organisation and so on [14]. Figure 2.6 highlights how the Named Entity Recognition process uses BILOU scheme to extract a ORG 'Imperial College London' and GPE: 'London'.

Tag	Meaning	Description
B	Beginning	First token of a multi-token entity
I	Inside	An inner token of a multi-token entity
L	Last	Final token of a multi-token entity
O	Outside	A single-token entity
U	Unit	A non-entity token

Table 2.1: BILOU tagging scheme

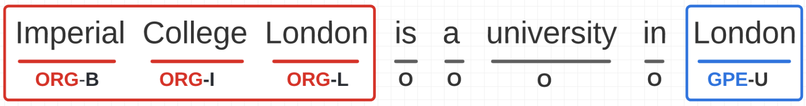


Figure 2.6: NER using BILOU

Conditional Random Fields

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 [32].

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:

$$s' = \underset{s}{\operatorname{argmax}}\, P(s|p)$$

Therefore, it is fairly simple to determine the 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]) [32], where 'PER', 'LOC', 'ORG' refers to person, location and organisation respectively.

The NER models are usually trained on specific domains and thereby only extract certain pre-defined types of entities such as PER, LOC, ORG etc. (see Appendix Table C.1), thereby making them domain-dependent.

2.5.1 Named Entity Disambiguation and Linking

Named Entity Disambiguation (NED): 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 [14].

A common approach for disambiguating entities is to use context representation using the entirety of the text to find co-occurrence of 'entity mentions' to establish 'candidate' entities and use an existing knowledge base through 'Named Entity Linking' to learn about the entities. The information from the knowledge can be used in conjunction with candidate-entity ranking approaches, which involve ranking candidate entries and retrieving the one with the highest probability for a target mention. An example in [33](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 as 'New York' (in USA) "(88 times) more than with the New York in England (0 times)" in the knowledge base. Quantifying the impact of co-occurrences, can be done by using an 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 ($s \neq t$) [33].

Additionally, **Named Entity Linking (NEL)** uses a standard (unique) International Resource Identifier (IRI) [34] for each disambiguated entity as described in a Knowledge Base (say, Linked Open Data (LOD) Cloud). Entity mentions are annotated by some NER algorithms, but they are restricted to the pre-defined types (such as persons, locations, organizations) [14]. Knowledge base (KB) hosts millions of entities and therefore NEL can be used to ground mentions of entities in some text to a central KB. An example mentioned in [35] highlights this: "David Murray recruited from Positive Black Soul" grounds Wikipedia articles for 'David Murray' (saxophonist) but not the 'David Murray' (musician) (disambiguation based on context).

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 or topics per industry from vast collections of news articles (grouped by industry) [36].

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):** 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 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 Principal Component Analysis (PCA) are used. A drawback of LSA is that it makes the orthogo-

nality 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 [36].

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 multinomial distribution (mixture) of topics with each topic itself being a mixture of (i.e., multinomial distribution) of words [37]. It is important to note that each topic can have multiple words and 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 [38].

   In order to extract the key topics within a volume of data (which, for the scope of this project, is news), some sort of metric is needed to extract the relevance of a topic. One such approach is highlighted in this paper [38](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. $I(t)$ represents the documents (i.e., news articles) obtained on day t [38](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 (by 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 heavy 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 [38] 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.

Topic 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 2.7). 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 indicator variable may be used to make that distinction [39].

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 [39] [40].

It is important to understand the structure of the sentiment (or opinion). Using the definition in this book [39], 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 [40].

The main steps of sentiment analysis involve:

1. Pre-processing the raw text (e.g. news articles) using techniques such as lemmatisation, Parts-of-Speech Tagging (POS) and stopword removal to break down each text document into its components (such as phrases, tokens etc.) [41].
2. Identifying sentiment-bearing phrases and assigning them a sentiment score, which can then be combined to assign a multi-layered sentiment.

For multi-layered sentiment, it is useful to use the notion of subjectivity (strength) and polarity scores at global and entity levels as mentioned in this paper [42], where global 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 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 [43] 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 [40]. The advantage with them is that they be customised to the domain. A state-of-the-art sentiment analysis model is RoBERTa (Robustly optimised Bi-directional Encoder Representations Transformers approach) which uses context-rich vector space embedding and attention masks to indicate to the model which tokens should get more attention, and which should not [44].

2.7.2 Considerations in scoring sentiment data

The scoring metric can make 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 [41]. 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 [42] 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 a negative polarity of -1. Similarly, when accounting for modifiers, 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 and 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 [42].

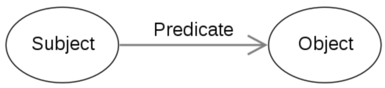
Another potential consideration is the use of duplicate news articles having an effect on the sentiment score. For this reason, the model should eliminate the redundancy of articles by not considering those duplicated articles [42].

2.8 Semantic Triples Knowledge Graph

Modelling events as a set of semantic triples is a well-established technique in the field of news summarisation. There can be different rules for extracting entities and relations from the text (in this case, news articles) which result in the extraction of different types of relationships. These form a minimal representation of the corpus, are generally of type $subject \rightarrow predicate \rightarrow object$ and are called SPO triples.

To generate this triples, the text corpus can be segmented into and sentences and using the grammatical structure of the sentence, the subject, predicate and object can be extracted [45] [46].

An example of the SPO triple extracted from the sentence “Elon Musk is the CEO of Tesla” is ‘Elon Musk’ (as subject), ‘CEO of’ (as predicate) and ‘Tesla’ (as object).



These triples are usually presented using a **Knowledge Graph**. A Knowledge Graph is a semantic graph which consists of nodes (vertices) and directed edges. The ‘subjects’ and ‘objects’ are represented as nodes and are usually entities and/or concepts. An entity can include a real-world physical object, for instance, a location (e.g. London), person (e.g. Paul McCartney), or an organization (e.g. WHO). Concepts, on the other hand, essentially refer to the general categories (that entities can belong to) such as airlines, websites, etc. [45].

The ‘predicates’ are represented as (directed) edges in a knowledge graph and are representative of the semantic relationships between entities and/or concepts derived by the descriptions of entities which have formal semantics. These descriptions usually highlight the key information about an entity and are interlinked (i.e., the description of one entity contributes to another) thereby resulting in the formation of a graph [45] as shown in Figure 2.7.

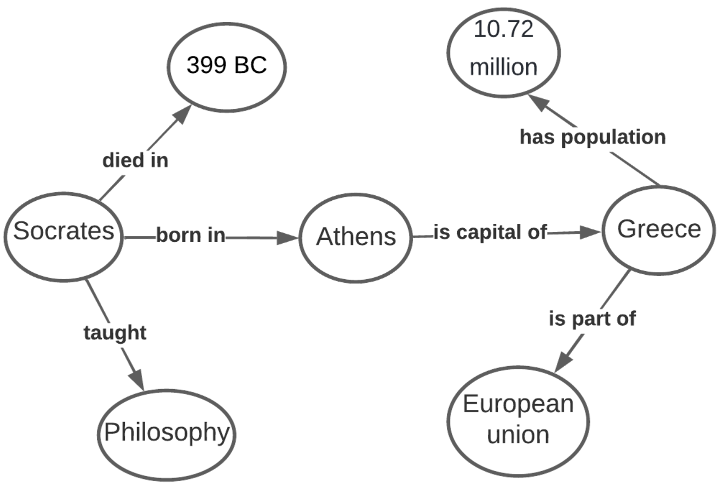


Figure 2.7: An example knowledge graph

Triple Store

Given their particular, consistent structure, a collection of triples is often stored in purpose-built databases called Triplestores. A set of these triples represent an RDF graph, where the constituent of the triple (RDF term) can be of three types: "internationalized resource identifier (IRI) [34], literal and blank nodes". The "subject can be an IRI or blank node, the predicate must be an IRI and lastly, an object can be an IRI, a literal or a blank node." [46](p. 66). The RDF store complies with W3C RDF and SPARQL standards. This is extremely useful as generally RDF triple stores use a query API SPARQL which enables the retrieval of relevant information [47].

Ontology

Ontology is a form of semantic knowledge representation. They can be used as a framework to build/ augment a knowledge graph. They model generalised types of objects (concepts, e.g. person), not the specific entities (e.g. Obama) [48].

An ontology consists of: distinct types of objects called classes, the properties/ attributes of these classes as well as the relationships between the classes (directed edges). E.g., The class 'Person' has a property 'has_DoB' and there is a relationship (directed edge) labelled 'has_DoB' from 'Person' to the class 'Date' [48].

Two extremely common frameworks to model ontology are Resource Description Framework (RDF) and Web Ontology Language (OWL). They are standard frameworks thereby allowing easy exchange/ migration of data [48] [45].

Why are knowledge graphs useful?

Knowledge graphs construction can be particularly useful as:

- They are able to extract information from semi-structured, structured, and/or unstructured data sources, combining knowledge to a well-represented graph structure [45].
- They incorporate the functionality of multiple storage types. For example, like databases, queries can be used for data retrieval and like knowledge bases, data stored has formal semantics, which can be used to interpret and infer data [49].
- They can perform logical reasoning/ inference by exploiting graph structure. For instance, by traversing the graph, it can be inferred that C is grandmother of A if B is mother of A and C is mother of B (transitive relations) [49].

Examples of Big Knowledge Graphs

- Google knowledge graph (GKG): Google uses GKG to incorporate semantic search functionality into its search engine for topic inference from the search query. It contains over "3.5 billion facts over 500 million objects/ entities" [45](p.68)
- DBPedia: is a crowd-sourced, multilingual knowledge base (RDF) which uses Wikipedia to extract structured data. It has approximately "24.9 million things in 119 languages, with 4.0 million things in English" [45](p.68).

- Wikidata: is a free, multilingual, crowd-sourced knowledge base, It contains structured data and supports Wikimedia projects. Wikidata has more ”than 59 million entities”. It has a powerful SPARQL query interface allowing for ad-hoc visualisations [14].

2.9 Principles of Visualisation

For the scope of the project, there needs to be some visualisation tool to display the results of semantic analysis of news. Therefore, in this section, we look at the important considerations to build coherent, effective visualisations.

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 (in this case, news articles) can then be processed to transform and summarise it, extracting key pieces of information deemed essential to the application. This transformed data can then be stored in our desired format and utilised to output the visualisation.

The following highlight the key principles 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 [50].
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 [50]. Additionally, it is important to provide context for these visualisations as well by using standardised labelling and indexing [51].
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 to how to fit the visualisation by dynamically scaling it to the virtual space as it grows [50].
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[51].
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 see the 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 | 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 Deep Search Labs (DSL) as a collection of news articles scraped from Bloomberg News, containing information such as the URL of the 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 the 'date published' column as the existing categories in the data 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 standard deviation of the value counts (size) of the groups was because the variance in the sizes of groups was too high. Therefore, the aim was to retain the larger year-category groups but omit the extremely small groups. This ensured that the input data was of significant size to yield relevant results before any further processing was done.

3.2 System Design

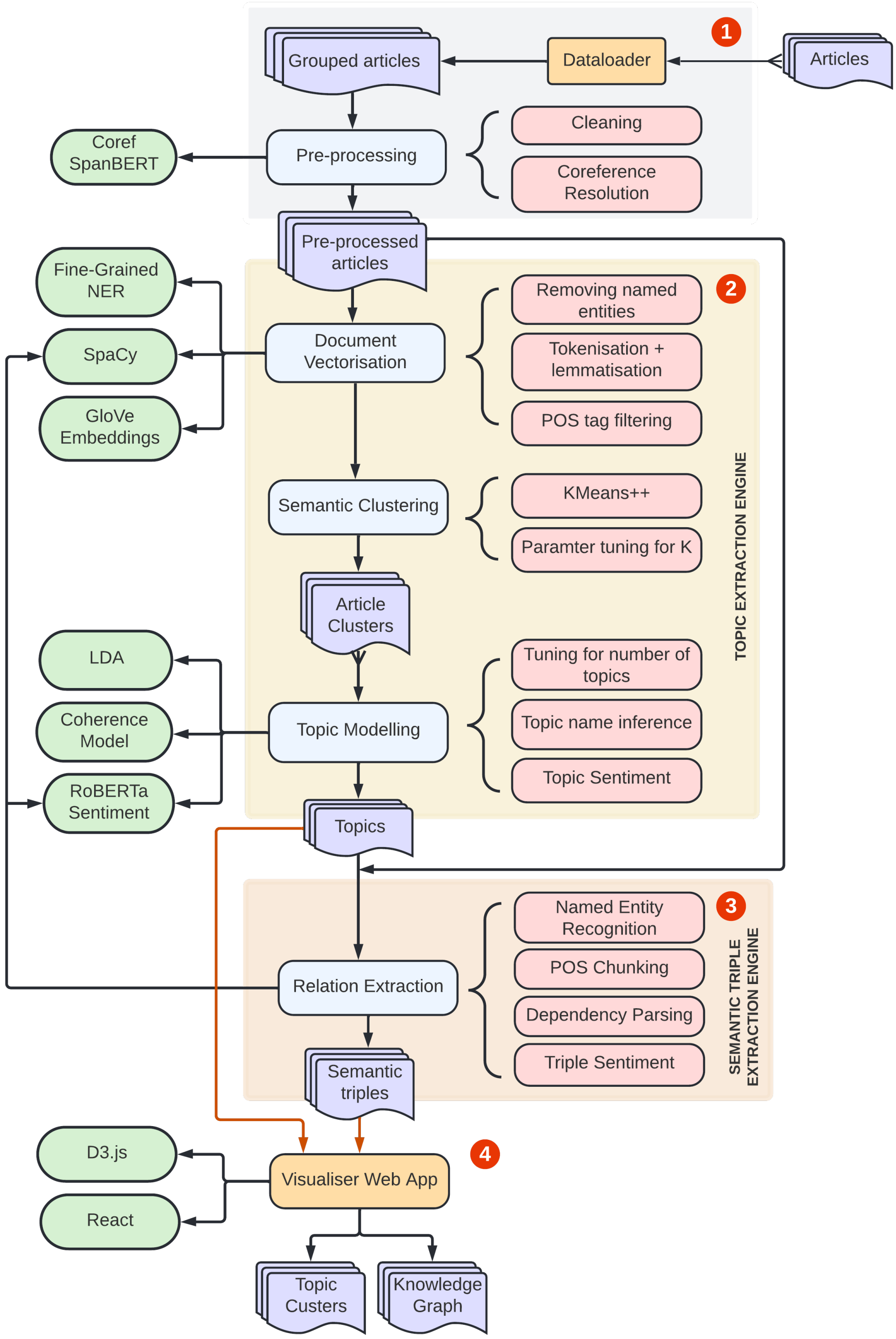


Figure 3.2: System Architecture Diagram

As established previously, 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 topic in a semantic cluster, this engine extracts triples of type subject-predicate-object from sentences in articles associated with the topic. Each triple corresponds to an article and is accompanied by its sentiment.
4. **Visualisation App:** Finally, the semantic clustering and latent topic models are visualised using circle packing graphs and the semantic triples are visualised using force-directed graphs developed using D3.js.

3.3 Technologies used

3.3.1 Libraries

SpaCy is an open-source NLP library "designed specifically for production use" [52]. It uses state-of-the-art pre-trained models and in-built pipelines for data processing techniques such as tokenisation, dependency parsing, 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 [53]. 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 [52].

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` as well as the Latent Dirichlet Allocation (LDA) model used for topic modelling.

AllenNLP is an open-source NLP library built on PyTorch and offers a variety of well-engineered existing state-of-the-art model implementations [54]. 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 is its 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 state-of-the-art pre-trained models which have proven to be useful in other domains for this problem. The models used in different parts of our solution 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 [25]. 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 [25].

RoBERTa Stanford Sentiment Treebank is a binary classifier trained on RoBERTa large for the Stanford Sentiment Treebank dataset, on which it achieved 95.11% accuracy [44]. 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, in which the model predicts ‘masked’ (hidden) words by learning their representation using words occurring to the left and right of the ‘masked’ word in the sentence (bi-directional). RoBERTa uses dynamic masking (i.e., different parts of the sentence are masked for each epoch), unlike BERT, which uses static masking (i.e., the same parts of the sentence are masked each epoch) [44].

Fine-Grained NER is a state-of-the-art named entity recognition model from AllenNLP and is a re-implementation of the Lample (2016) [55] model. It makes use of bi-directional LTSM with a Conditional Random Field layer (CRF) and uses two types of embeddings: character embeddings and ELMo embeddings (see Sections 2.5–2.4.2). In this paper [55], a comparison study for the performance of the model for English NER on the CoNLL-2003 test set highlights that it outperforms several other models giving an F1 score of 90.94% [55]. The motivation for using this model was this high accuracy and that the model identifies a broad range of 16 semantic types (see Appendix Table C.1).

3.4 Visualisation Web Tool

As an extension to the core body of work done including 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 these 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 our work, the application was hosted on Heroku [56] as a scalable approach to accommodate user testing and evaluation discussed in Section 6.4. This tool adopted the principles of visualisations discussed in Section 2.9 such as incorporating user interactivity and an intuitive consistent layout with labelled information.

4 | Topic Extraction Engine

The semantic analysis tool shown in Figure 3.2 focuses on 2 main text mining components 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) comprises a list of articles that were published in that year and belong to the category. Processing the entire text for each article in the dataset would result in intensive computational load without a significant benefit, and increase the 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 8 sentences of the article using spaCy's `SentenceSplitter`. Throughout the course of this chapter, 'intro' and 'article' will be used interchangeably to represent a single news article in the corpus.

Once the article introductions ('intros') 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 sentences "Sean Doyle is the CEO of British Airways. Prior to this, he was the CEO of Air Lingus", if the coreference resolution was done at 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 needs to be 'cleaned' to ensure that the news articles retrieved are informative and useful. This involves the following steps:

1. Removing any unnecessary characters e.g., trailing ellipsis, parenthesis etc.
2. Stopword removal using spaCy's default stopword list for the pre-trained English model, containing words such as 'its', 'moreover', 'that', 'are' etc. This stoplist is augmented with a custom list of words such as 'told', 'said', 'airline', 'flight' etc. which, given our dataset, i.e., online articles for the airline industry, occur in the majority of the articles. These words do not contribute to any relevant information specific to the article and are therefore removed in an attempt to reduce redundancy. Similarly, common phrases in online articles such as "Read here", "For more information" etc. are also removed. This significantly reduced the amount of text per article.

4.1.3 Named Entity Recognition

These pre-processing steps aimed 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 from the '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 6.2.1 respectively.

This step was done before tokenisation as named entities are often multi-word entities 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 all instances of 'New' and 'Haven' from the 'article intro'. Therefore, regular expression patterns (see Appendix Table D.1) were used to remove all the occurrences of the extracted entities in an article. This included removing all entity types (see Appendix Table C.1) which included, but were not limited to, 'PER', 'LOC', 'GPE', 'MONEY', 'DATE', 'ORG' etc. 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 the named entities from the article corpus involved using the Fine-Grained NER model (see Section 3.3.2) and the BILOU encoding scheme for NER (see Section 2.5) and is detailed in Algorithm 1. The output from the NER model gives a set of words and their corresponding entity tags. In essence, the algorithm shows the process of obtaining single-word entities (with 'Unit' tag) and multi-word entities (by joining entity tokens to build the entity phrases starting from the 'Beginning' token until the 'Last' token) and ignoring words with the 'Outside' tag as they do not refer to an entity.

Algorithm 1 Extract Named Entities

$entities \leftarrow \emptyset$
$words, tags \leftarrow ner\_model.predict(sentence)$
for all $word, tag \in zip(words, tags)$ **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 is cleaned and void of the pre-determined entity types, the article intros are tokenised and normalised. This was done using the spaCy library's `Tokeniser` and `Lemmatiser` respectively, which are pre-trained on the English language model: `en_core_web_lg` (see 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 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 the `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 The Term Frequency-Inverse Document Frequency statistical metric is used to assess the relevance of a token to an article in a collection of articles. Representing the tokens as their TF-IDF measure 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 is multiplied by the inverse document frequency to weigh down the terms whose high frequency is not unique to a document (see Section 2.4.1). A dense document-term matrix using these TF-IDF values 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 [57].

GloVe The final approach was to use Stanford's `GloVe` (global vector) embeddings from Gensim. `GloVe` has been pre-trained on Wikipedia and Gigaword 5, consisting of a vocabulary of 400,000 words which are represented by 300 dimensional vectors [30]. 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 all *relevant* tokens for each article intro. As mentioned above, the filtered tokens were vectorised using the `GloVe` embeddings [30] resulting in a list of vectors corresponding to the `filtered_tokens`. To generate the document vectors from these word vectors, three approaches were tested:

Approach 1 — The tokens in the `filtered_tokens` list for each 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 — Instead of simply averaging token vectors, the document vector for each article 'intro' was calculated using a weighted sum of corresponding word (token) vectors (from `filtered_tokens`). This was done by using a TF-IDF 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 'intro' vector.

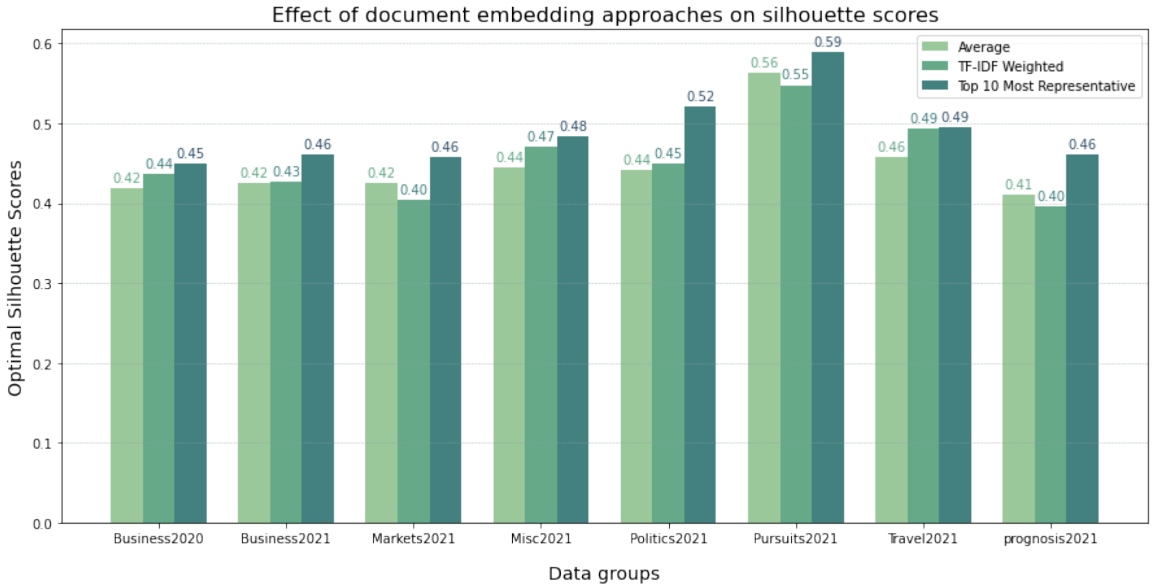


FIGURE 4.1 Effect of document vector generation approaches on clustering silhouette scores where Approach 1 = Average, Approach 2 = TF-IDF Weighted, and Approach 3 = Top 10 most Representative

Figure 4.1 shows how the different document (article) embedding approaches affect the 'goodness' of the clustering indicated by silhouette scores. The significance of these scores for clustering is explained in detail in Section 4.2.5. The consensus derived is that Approach 3, which describes calculating the article 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

Before performing any dimensionality reduction, the first step is to normalise the article vectors to eliminate redundancy in data and standardise the features (components of the vector). Clustering article vectors involves minimising a 'distance' metric. If different vector components (features) have different scales, derivatives will tend to align along dimensions with higher variance, resulting in poor convergence. For instance, for a given dataset, if dimension 1 is in the 100s and dimension 2 is in 1s, then dimension 1 will have a much higher effect on the overall distance, making the clustering biased towards it. Therefore, normalisation (or feature scaling) is crucial for clustering as it controls the variability in a collection of document vectors, re-scaling the feature components to share a common scale, thereby improving clustering. This is also a useful prerequisite for principal component analysis (PCA) of article vectors, as 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 (see Section 4.2.1), `GloVe` embeddings are used for article vector generation. Clustering the document vectors becomes inefficient and meaningless at these high dimensions as the concept of distance becomes less precise when the number of dimensions increases [58]. This is because the volume of space increases exponentially making the available data sparse - 'curse of dimensionality'. For any given point in this high n-dimensional space, the difference in 'distance' (euclidean) between the closest point d_{min} and the farthest point d_{max} with respect to d_{min} becomes negligible [59].

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

The concept of clustering these articles relies on grouping similar articles together based on their features (vector components). However, given the high dimensional data, some of these features are not significantly relevant in determining the clusters. The goal is to find the vector components with the most variance across the article vectors to represent the features. This is accomplished by performing principal component analysis (PCA) on the normalised high-dimensional input vector space to map it to a lower dimensional space, whilst minimising information loss [58].

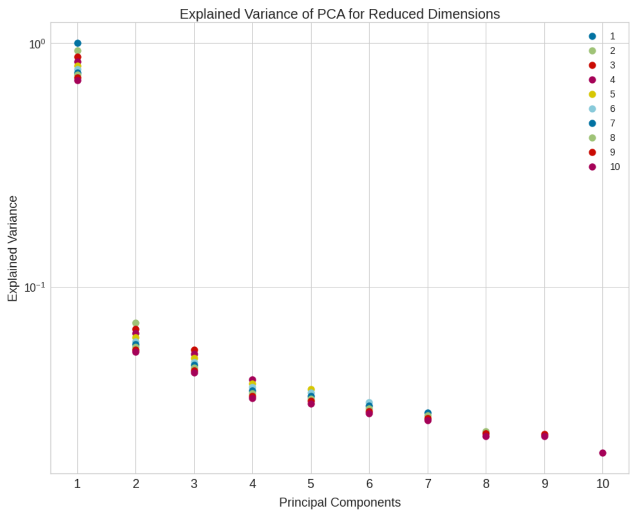


FIGURE 4.2 Proportion of explained variance across principal components for transformed data to dimensions=1..10

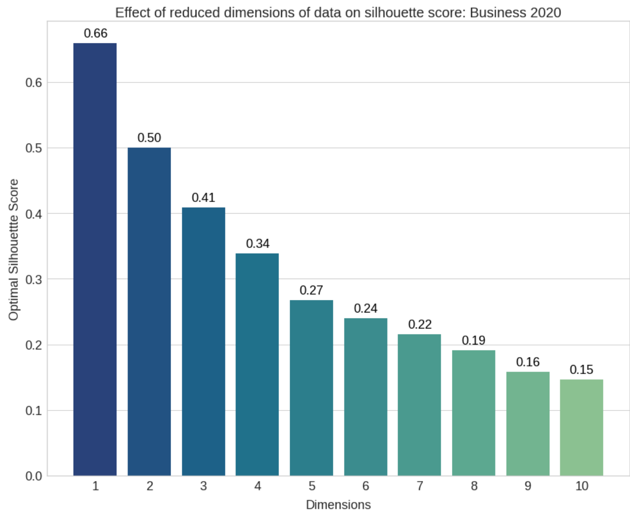


FIGURE 4.3 Optimal silhouette scores for data with dimensions=1..10 transformed using PCA

As seen in Figure 4.2, when the article vector space is projected to dimension=1, the variance is explained entirely by the 1^{st} principal component (indicated by the blue dot). As the number of dimensions increase, i.e. the number of principal components increase, the main proportion of the variance (close to 1) can still be explained by the first principal component. Projecting to 10 dimensions, the proportion of explained variances (as a percentage) for each of the 10 principal components is as follows:

PC	1	2	3	4	5	6	7	8	9	10
EV(%)	70.6%	5.4%	4.4%	3.5%	3.3%	3.0%	2.8%	2.4%	2.4%	2.1%

Therefore, even at 10 dimensions, 70.6% of the variance can be explained by the first principal component (PC) alone and therefore, the article vector space can be transformed using PCA with dimension=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'(see Section 4.2.5). It shows that transforming the vectors to a single dimension (using PCA) results in the highest silhouette score. Therefore, Figures 4.2–4.3 conclude that the transforming the article vectors to 1 dimension using PCA results in the most optimal clustering whilst minimising information loss.

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 `kmeans` function exposed by the `sklearn` library but by using 'euclidean distance' (4.4) (instead of 'cosine distance') as the distance metric. This is feasible given the linear relationship between euclidean distance and cosine distance [60] for normalised vectors shown in (4.6).

$$\begin{aligned}
For\ normalised\ vectors,\ x,y: \sum x_i^2 = 1, \sum y_i^2 = 1 && (4.1)\\
Cosine\ Similarity,\ cos(x,y) &= \frac{\sum x_i y_i}{\sqrt{\sum x_i^2 y_i^2}}\\
&= \sum x_i y_i && (4.2)\\
Cosine\ Distance &= 1 - cos(x,y) && (4.3)\\
Euclidean\ Distance,\ ||x-y||_2^2 &= \sum (x_i - y_i)^2 && (4.4)\\
&= \sum (x_i^2 + y_i^2 - 2x_i y_i)\\
&= \sum x_i^2 + \sum y_i^2 - 2\sum x_i y_i\\
&= 1 + 1 - 2cos(x,y)\\
&= 2(1 - cos(x,y)) && (4.5)\\
&= 2(Cosine\ Distance) && (4.6)
\end{aligned}$$

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.

4.2.5 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 (also known as, 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 (or distortion score) which is the sum of the squared distance between each point in a cluster from its corresponding centroid [61] as shown in 4.7.

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

where C is the cluster centroid and d is the data point in each cluster.

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.

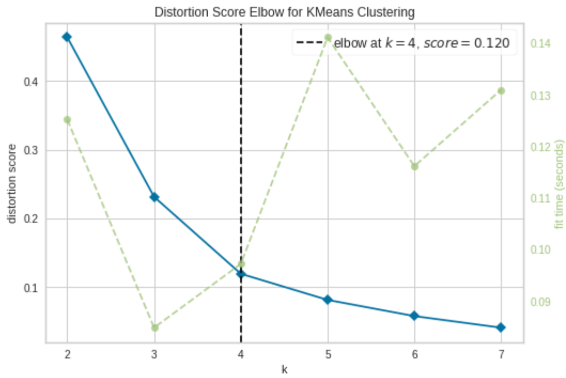


FIGURE 4.4 Elbow Method Plot for KMeans clustering with k=2..7 for input group: 'Business 2020'

Method 2: Silhouette score

The second approach involved using the silhouette score. Unlike the elbow method, the silhouette score accounts for both how 'close' a point is within its cluster (cohesion) [61] as well as how close it is to other clusters (separation) [62].

$$Silhouette\ Score = mean_i(\frac{S_i - C_i}{\max(S_i, C_i)}) \tag{4.8}$$

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

The silhouette score for a clustering as shown in 4.8 is the mean of the silhouette score for all points i in the data. It ranges from -1 to 1. A score near 1 implies that clusters are well separated and distinct, while a score close to 0 implies that there is extensive overlapping between the clusters and that the clusters cannot be differentiated. Finally, a score close to -1 implies that clusters are incorrectly assigned.

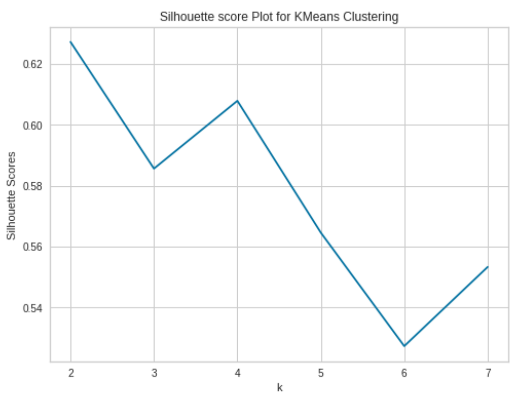


FIGURE 4.5 Silhouette Plot for KMeans clustering with k=2..7 for input group: 'Business 2020'

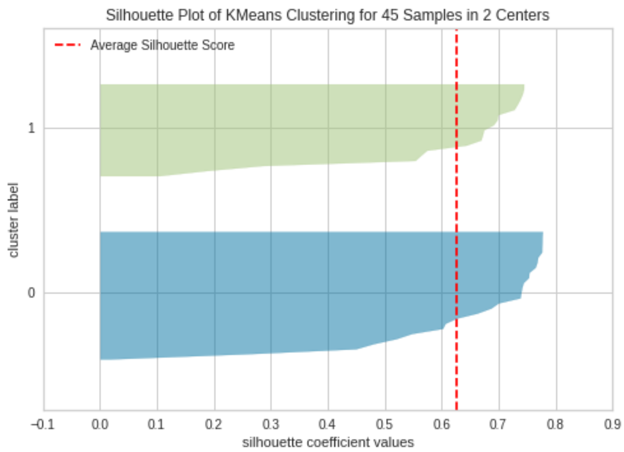


FIGURE 4.6 'Business 2020' Silhouette Diagram: score = 0.627 for k=2

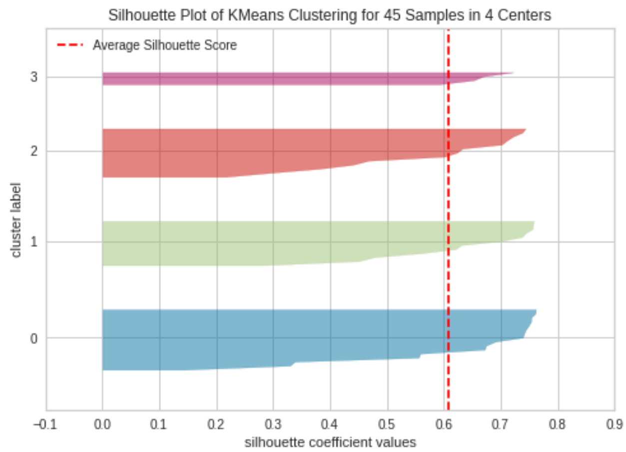


FIGURE 4.7 'Business 2020' Silhouette Diagram: score = 0.607 for k=4

From Figure 4.5, it is apparent that although 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, and 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 the 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 silhouette score as the clustering comparison statistic as it factors in both, how compact a cluster is and how distinct it is. This is a vital consideration given that the Topic Extraction Engine aims 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 the cluster-to-article mapping. However, given the high volume of data, factoring every single article in the cluster is not necessary, particularly those that are far from their respective cluster centroids. Therefore, the articles are sorted in increasing order of (euclidean) distance from the centroid 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. The n_{max} and n_{max} values are computed based on the mean and standard deviation of the sizes of semantic clusters, $|C|$, as shown in (4.9)–(4.10).

$$n_{min} = \lfloor mean(|C_1|, |C_2|, ..., |C_n|) - std(|C_1|, |C_2|, ..., |C_n|) \rfloor \tag{4.9}$$
$$n_{max} = \lceil mean(|C_1|, |C_2|, ..., |C_n|) + std(|C_1|, |C_2|, ..., |C_n|) \rceil \tag{4.10}$$

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 LDA 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 an article. For the scope of this problem, each article was associated with 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 6.2). Using the `filtered_tokens` satisfied most of these requirements which are explained below:

1. **Noun-only corpus**: Based on previous studies [10] [63], and the results from Section 6.2.2 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 (see Appendix Table A.1), resulted in suboptimal topics since the LDA model will give "fast" and "pandemic" equal importance.

2. **No Named Entities**: Similar to semantic clustering, the decision was 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. The aim of this was to prevent the LDA model from fitting topics to named entities instead of common nouns (which indicate key themes in a corpus).

3. **Using Bigrams**: Inspired by [64, '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 that news articles often contain 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 documents need to be transformed into numeric feature vectors which form the input corpus. Since it does not rely on the article's semantic information and builds topic estimators based on words, articles and topics, it uses frequency vectors obtained by Bag-of-Words or Term Frequency-Inverse Document Frequency (TF-IDF). The latter was chosen over the more common Bag-of-Words [65], in an attempt to minimise overlap of topics as each article will be associated with one dominant topic.

4.3.2 Determining optimal number of latent topics

A key contributor in determining the quality of the topics extracted is tuning the `num_topics` parameter which represents the number of topics outputted by the LDA. A low value will result in too few or overly generalised topics whereas a high value can result in uninterpretable topics that ideally should have been merged [66]. Previous studies [67] [68] 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 [68].

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 for `num_topics` varying from min_topics (defaulted to 2) to max_topics (which is dependent on the size of the semantic cluster). 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 reduces unnecessary computation, improving performance.

Dominant Topic-Article Mapping

Once the optimal value of `num_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 the article belonging to the topic) is selected, resulting in a 1-1 topic-article mapping.

4.3.3 Topic Name Inference

An augmented feature of the Topic Modelling Engine was to infer the topic names from the topic keywords. Algorithm 2 details this process. This involves splitting any bigram keywords to obtain a set of keyword tokens which are validated to ensure they are nouns, not a stopword and have an associated `Glove` embedding. 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 enforced to be nouns and checked against the augmented stopwords (see Section 4.1.2). The top 3 most similar topic name tokens that meet the validity checks are selected as the 'topic name' for the given topic. If there are no valid `topic_name_candidates`, the first valid keyword is used as the topic name as the keywords from the LDA are outputted in descending order of importance.

Algorithm 2 Infer Topic Name

```
function GetValidKeywords(topicKeywords, stopwords, allowed_pos =['NOUN'])
    validKws ← ∅
    for all topicKw ∈ topicKeywords do
        kw ← topicKw.lemma
        if kw ∈ stopwords  and kw ∉ GloVe.vocab  and kw.pos ∉ allowed_pos then
            continue
        validKws.append(kw)
    return validKws

function GetTopicName(topicKeywords, stopwords)
    kws ← GetKeywordTokens(topicKeywords)                          ▷ splits bigrams
    validKws ← GetValidKeywords(kws, stopwords)[: 5]
    topicNameCandidates := GloVe.most_similar(validKws, topn = 5)
    validTopicNames ← GetValidKeywords(topicNameCandidates, stopwords)
    if len(validTopicNames) == 0 then
        return validKws[0]
    else
        return validTopicNames[0 : 2]
```

4.3.4 Topic Sentiment

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. The RoBERTa model trained on Stanford Sentiment Treebank (see Section 3.3.2) is used on the (coreference resolved) article 'title', without any data cleaning or POS tag filtering (see Section 4.1.2), to obtain the article sentiment. Data cleaning is omitted because removing stopwords for sentiment analysis is often not a good idea as it can change the meaning of the text, especially by eliminating negations such as 'not', 'n't' etc. Additionally, by not restricting the input to nouns, the RoBERTa model can account for the adjectives and adverbs that contribute to the strength of the sentiment associated with a particular entity. The article 'title' is used instead of the entire article as the title provides the main topic sentence for the article, serving as a good descriptor of the tone of the article. News articles mention several target entities (some more relevant than others) and the sentiment around each of these will equally influence the overall article sentiment. This diverts focus from the main point of the article (summarised nicely by the title), resulting in a sentiment that does not coincide with the key tone of the article as it is influenced by too much noise.

Given that the model is a binary classifier, it produces a binary label, where 0 represents 'negative' sentiment and 1 represents 'positive' sentiment. The sentiments of the articles linked with a topic are then averaged, and the topic is assigned one of three sentiments based on the range of the resulting value: 'Positive,' 'Negative,' or 'Neutral' as shown in Algorithm 3.

Algorithm 3 Computing Topic Sentiment

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

4.4 Results and Discussion

For topic extraction, usually, one of either cluster analysis or topic modelling is performed on the input data. By combining these methods, the Topic Extraction Engine provides high-level semantic grouping through clustering and finer-grained grouping through topic models (using LDA), thereby improving the coherence of the extracted latent topics. Using POS filtering, stopword removal, lemmatisation and TF-IDF to pre-process the LDA corpus allows for distinct topics with minimal overlap in the semantic clusters (indicated by relatively high coherence scores in Section 6.2). The confidence in this engine is obtained through qualitative analysis of its results, displayed by the visualisation tool as shown in Figures 4.8–4.10.

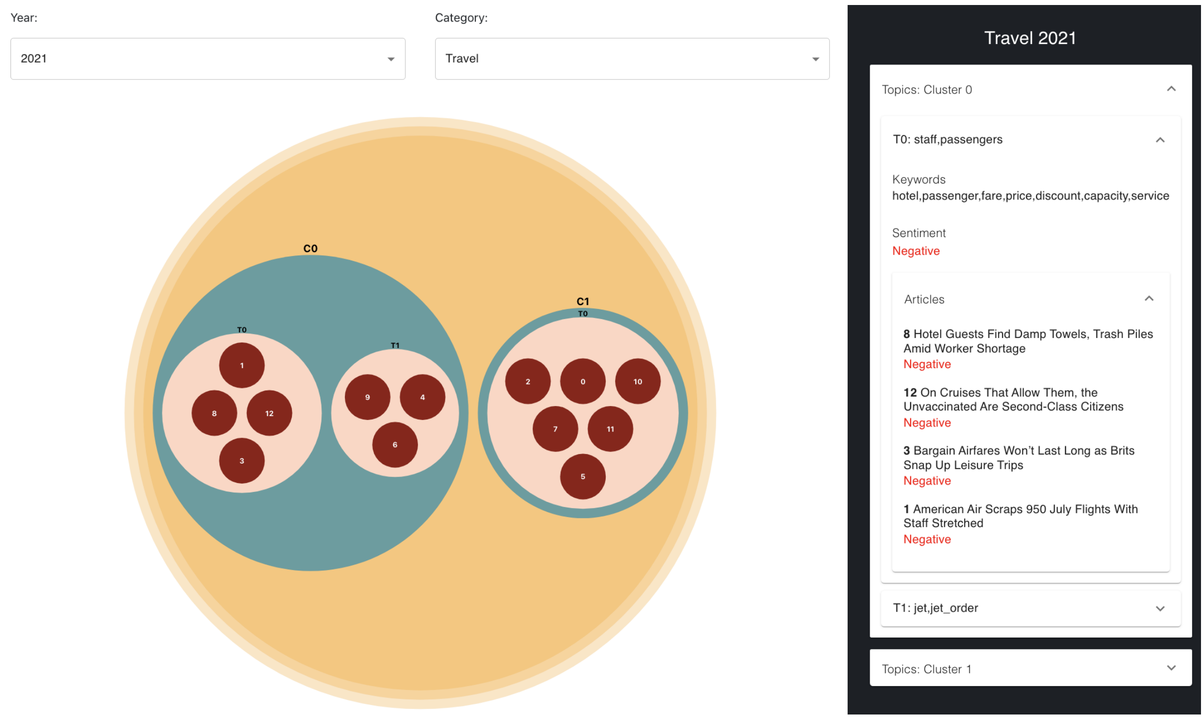


FIGURE 4.8 Cluster-Topic diagram for Cluster 0 (C0) in 'Travel 2021', with an expanded view of topic information for topic T0

The objective of this engine is to derive semantic clusters from the article corpus for each Year-Category group, apply topic modelling on each of these clusters to obtain meaningful, isolated latent topics (represented by the ‘topic name’ and ‘keywords’) and provide topic information such as topic sentiment and associated articles. Additionally, we want to maintain that the distinction among the clusters is significant and intuitive by way of the topics within them.

Figure 4.8 displays the ‘cluster-topic’ graph for ‘Travel2021’ which has 2 semantic clusters ‘C0’ and ‘C1’, with 2 and 1 topic(s) respectively. In particular, this figure displays the expanded ‘accordion’ for the first topic ‘T0’ for Cluster 0 (‘C0’). This particular input group contained limited data and is used as an example for ease of understanding. We can see that topic (T0) in Cluster 0 (C0) has the inferred topic name of ‘staff, passengers’ with the topic keywords: ‘hotel’, ‘passenger’, ‘fare’, ‘price’, ‘discount’, ‘capacity’ and ‘service’. The articles for this topic (numbered 1,3,8,12) 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 for this topic have a ‘negative’ sentiment as they generally focus on airlines struggling with poor service due to worker shortage, resulting in an average ‘negative’ sentiment for the topic.

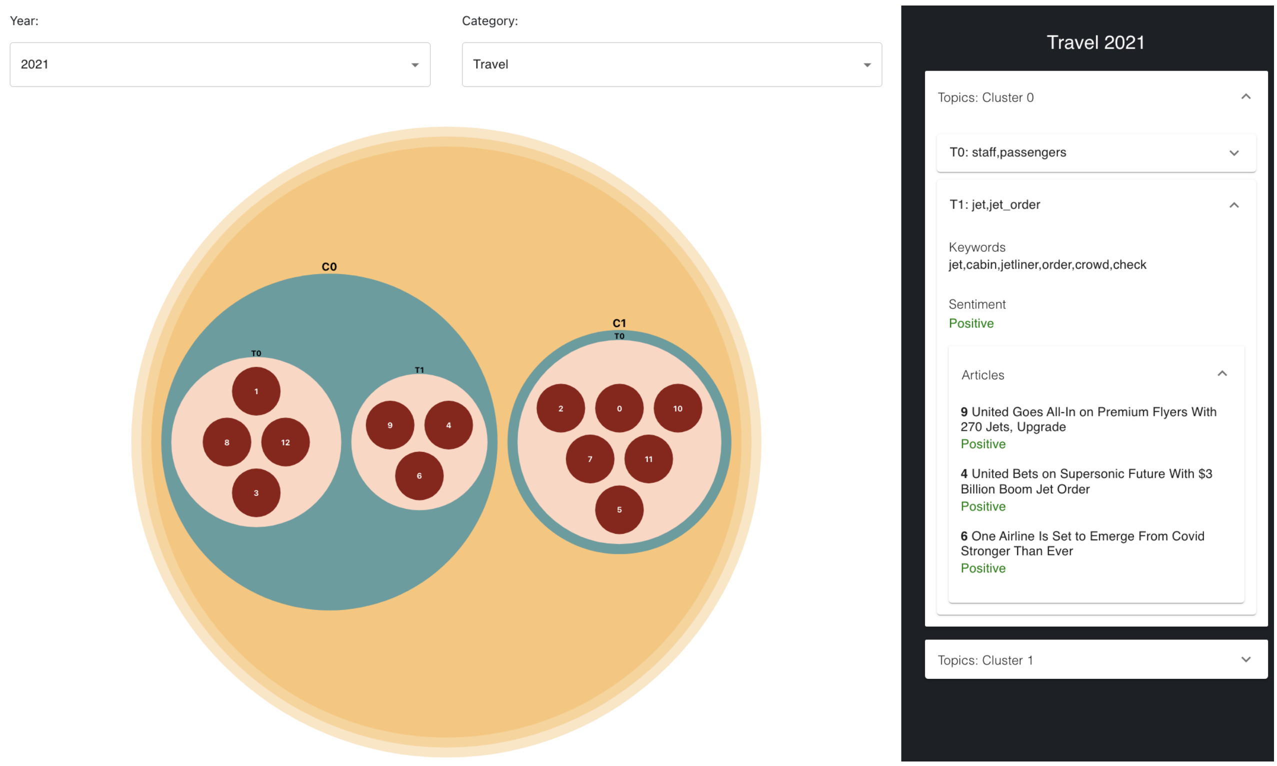


FIGURE 4.9 Cluster-Topic diagram for Cluster 0 (C0) in ‘Travel 2021’, with an expanded view of topic information for topic T1

Similarly, Figure 4.9, shows the expanded ‘accordion’ for the other topic (T1) in the same semantic cluster (C0) with the inferred topic name of ‘jet, jet_order’ and topic keywords: ‘jet’, ‘cabin’, ‘order’, ‘jetliner’, ‘crowd’ and ‘check’. This topic differs from the previous topic as it focuses on the positive developments made by airlines by investing in new planes and jets.

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

Therefore, based on the information presented in Figures 4.8–4.9 about topics T0 and T1 respectively, for the semantic cluster C0, we can infer that the high-level semantic grouping in C0 focuses on ‘airlines’ and ‘aircraft’ with T0 focusing on the airlines struggling with staff and passenger service and T1 on investment in aircraft through new jet orders. This gives confidence in the Topic Extraction Engine’s ability to extract meaningful distinct topics from the article corpus for a given semantic cluster.

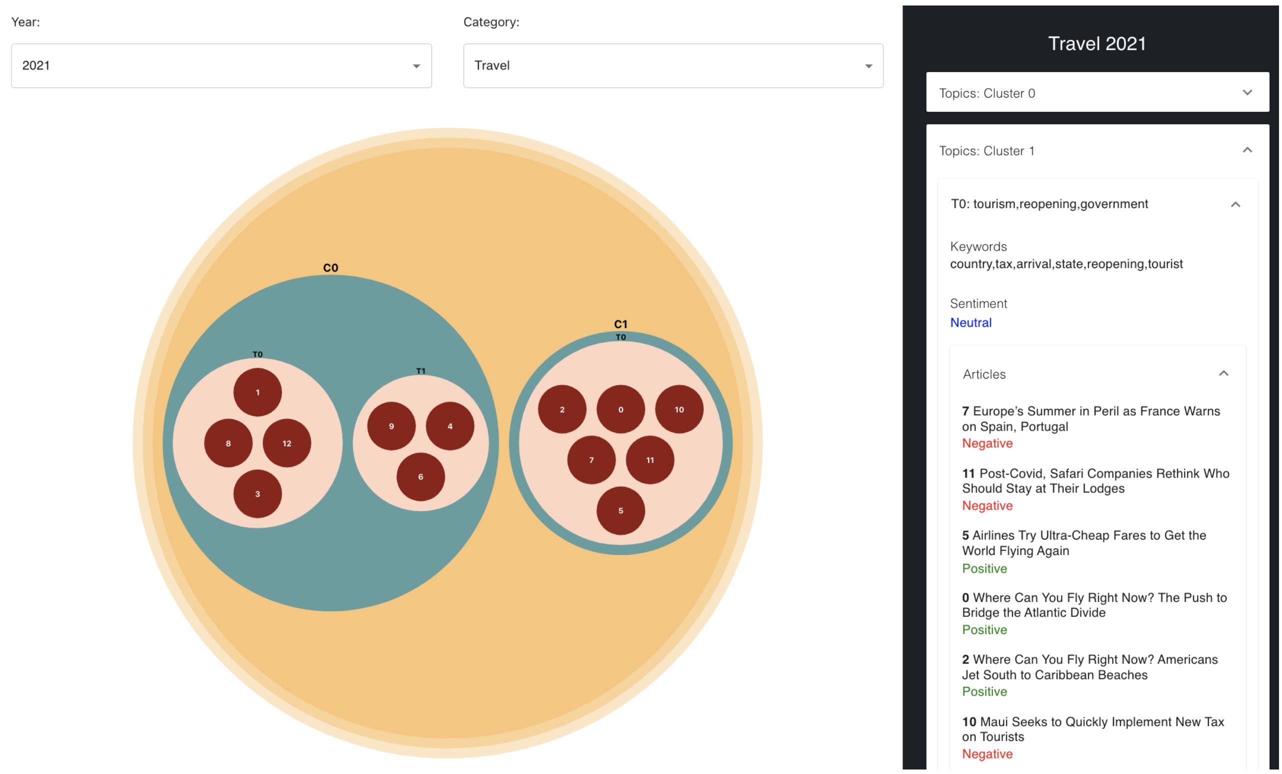


FIGURE 4.10 Cluster-Topic diagram for Cluster 1 (C1) in ‘Travel 2021’, with an expanded view of topic information for the only topic T0

Figure 4.10 displays the information about a different semantic cluster (C1) which has only one topic (T0) 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 tourism. The articles associated with the topic TO indicate a divided outlook towards airline tourism as some articles talk show reluctance in resuming travel as they talk about implementing new tourist taxes (e.g., article 10) and screening guests post-Covid (e.g., article 11), while others imply an eager approach towards promoting tourism as they mention airlines adopting cheaper fares and pushing travel (articles 0,2 and 5). This divided view associated with post-Covid travel is reflected in the sentiments of these articles, thereby resulting in a ‘Neutral’ average sentiment for topic T0 in Cluster 1.

Therefore, based on results seen in Figures 4.8–4.10, we also get confidence in the Topic Extraction Engine’s semantic clustering, as we can see the distinction in the different clusters extracted: C0 which focuses on airlines, jet orders and C1, which focuses on resuming tourism. Furthermore, the qualitative differences between the topics within clusters are also apparent, which successfully provide insight into the relevant articles contributing to these topics, as well as the associated sentiment regarding that topic.

4.4.1 Limitations

One of the limitations of the Topic Extraction Engine is the inherent limitation of topic models. There is a certain ambiguity associated with the term 'topic'. While they provide a general grouping of themes in the text, they can not be interpreted as highly nuanced article classification. For topics with several associated articles, it can sometimes be difficult to interpret the topic, particularly the 'topic name'. This can be attributed to the fact that keywords for large topics (i.e., associated with several articles) will have a higher variance in semantic similarity in terms of their `GloVe vector embeddings`. This could result in the current Topic Name Inference approach, which uses the `gloVe.most_similar()` function, not being able to extract a 'meaningful' topic name due to semantic dissimilarity between topic keywords.

As mentioned earlier, for the scope of our work, the decision was made that each article is associated with one dominant topic. This can result in the engine omitting certain topics, labelling them insignificant as there are insufficient articles associated with them. This may be false as it may be the case that there are sufficient articles that have these topics as an underlying theme, but just not as the dominant topic.

Additionally, as expected, the performance of the Topic Extraction is limited by the performance of the state-of-the-art models such as SpanBert for coreference resolution, RoBERTa for sentiment analysis and Fine-grained NER for named entity extraction.

5 Semantic Triple Extraction Engine

This section focuses on the Semantic Triple Extraction Engine, particularly on triple representation, implementation, and the key decisions for information extraction.

5.1 Motivation

The engine aims 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 (see Section 3.3.2) for each topic in a semantic cluster. The limitation of this approach was that while it was a good indicator of the 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 of the content discussed in articles for a given topic, the decision was made to extract semantic triples of type `subject-predicate-object`. The triples serve as a minimal representation of information in an article without losing its context. The process of extracting these relations involved inferring both syntactic and semantic dependencies between tokens in a sentence, in particular relying on a lexico-semantic approach by making use of tokenisation, dependency parsing, part-of-speech tagging and named entity recognition. These triples would then be displayed as a knowledge graph (see Section 2.8) 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.

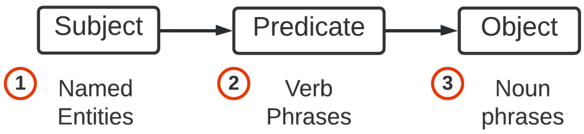


FIGURE 5.1 Decisions for Extracting Semantic Triples

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 to infer information about these entities from the articles and 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 the 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 [69], 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 the predicate is that it exploits the structured grammar present in news articles where generally the subject is 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 relations of type $subject \rightarrow verb \rightarrow object$ 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. Since the SVO triples follow a pattern of $noun(phrase) \rightarrow verb \rightarrow noun(phrase)$, 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 and object phrases from the articles in a topic in order to form an entry for the knowledge graph.

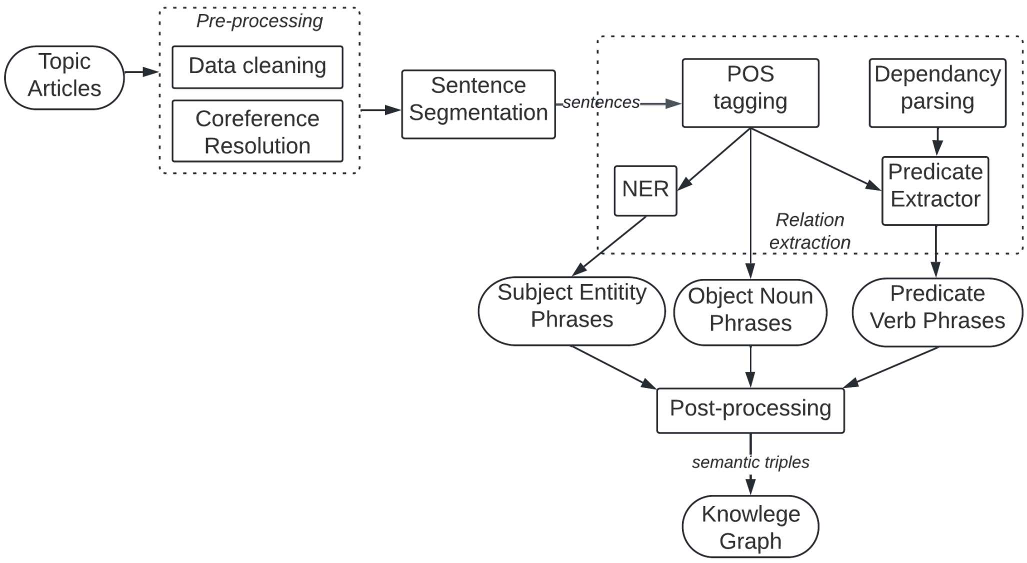


FIGURE 5.2 Overview of Semantic Triple Extraction

5.3.1 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. The `updated_stopwords` from Section 4.1.2 is changed slightly to remove any negations such as 'not', 'won't' etc. While these words were not useful in the Topic Extraction Engine which used a noun-only corpus, negations are relevant when extracting triples as removing these will result in incorrect relations/predicates in the semantic triples and are therefore retained. Each topic article then undergoes sentence segmentation via spaCy's `SentenceSplitter` to obtain a list of sentences from which the triples can be extracted.

5.3.2 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) using the following list of matching patterns.

```
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 list contains two matching patterns using regex operators (see Appendix Table D.1) and POS tags to find different permutations of substrings (i.e., verb phrases) to satisfy these patterns. This process does not limit the verb phrase to the 'VERB' POS tag but instead accounts for other relevant information such as auxiliary verbs ('AUX'), adverbs ('ADV') and adpositions ('ADP') to be extracted from the sentences as candidate verb phrases in order to retain semantic context. This process is known as 'verb chunking'.

Each of these patterns is based on the commonly occurring grammatical patterns in the English language. For example, the first pattern has an optional requirement (0 or 1 instance) for auxiliary verb ('AUX') followed by an optional requirement for particle ('PART') followed by a strict requirement (at least one instance) for 'VERB' followed by an optional requirement for 'ADP' adposition. Such a pattern allows for the extraction of different combinations of verb chunks such as 'has(AUX) been(VERB)', 'has(AUX) not(PART) been(VERB)', 'has(AUX) not(PART) been(VERB) in(ADP)', 'contains(VERB)', 'belongs(VERB) to(ADP)' etc. Similarly, the other pattern tries to extract a verb chunk with an optional 'VERB' followed by 0 or more instances of 'ADV' (adverbs) followed by at least one instance of 'VERB', resulting in verb phrases such as 'increasing(VERB)', 'slowly(ADV) increasing(VERB)', 'is(VERB) slowly(ADV) increasing(VERB)' etc.

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, which is commonly 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.

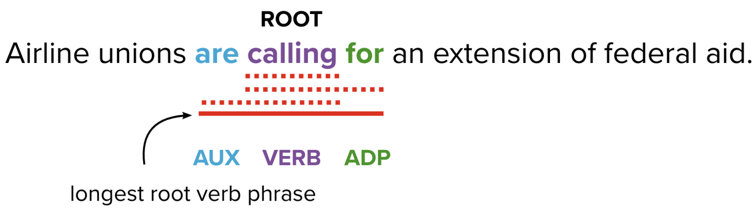


FIGURE 5.3 Verb phrases extracted from an example sentence from article corpora

5.3.3 Extracting Subject and Object

As discussed in Section 5.2, the 'subject' and 'object' are noun phrases (NP), particularly entity phrases for 'subject'. Therefore, 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 [52] and are returned as a list of `Spans` [52], 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**
      $ents \leftarrow np.containsAny(validEnts)$ ▷ 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

In order to extract the subject phrase, we get the first noun phrase (NP) that occurs before (to the left of) the verb phrase (i.e., predicate) as well as contains at least one of the selected set of named entities (i.e., `valid_entities`). The named entities contained in the noun phrase (`np`) are stored as candidates for the subject (`ents`). The `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 (see 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 the named entity extraction procedure detailed in Algorithm 1, a set `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 one, based on the position of the entity `Span` in the sentence, is selected as the subject phrase. As mentioned in Section 5.2, the aim is to have just the named entity as the subject to facilitate the connectivity of triples in the knowledge graph (KG). However, we do not want to 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 node is 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')` results in a common subject node between the triples as shown in Figure 5.5. Without this, the KG would be poorly connected with 2 distinct triples as shown in Figure 5.4.

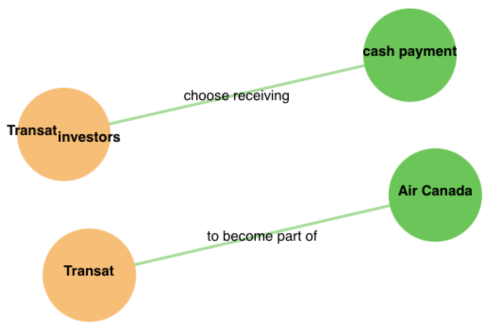


FIGURE 5.4 KG snippet before predicate augmentation

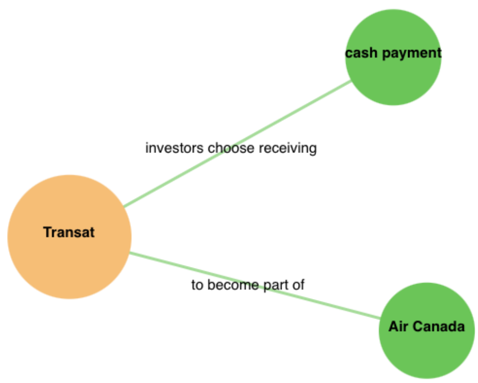


FIGURE 5.5 KG snippet after predicate augmentation

Get object phrase

Getting the object is fairly simple as shown in Algorithm 4 and involves getting the first 'noun phrase' (based on position of `Span`) after the 'predicate' (the longest root verb phrase).

Post-processing

Once the semantic triples are obtained from the article corpus for each topic, they undergo filtering. This involves ensuring that all components of the triple are not null (or an empty string) and the triples where the relation/predicate contain words like "said" and "told" (including all their grammatical forms) are omitted. This is done because, very commonly in news articles, direct quotes are made by entities of the forms "$\langle quote \rangle$, $said$ $\langle entity \rangle$" or "$\langle entity \rangle, said \langle quote \rangle$". The root verb extracted by the engine in these cases would be 'said' and the triples would be of type ($\langle junk \rangle$, $said$, $\langle entity \rangle$) or ($\langle entity \rangle$, $said$, $\langle junk \rangle$) respectively, where $\langle junk \rangle$ implies incoherent subject or object nodes, resulting in poor relations that do not add to the quality of information extracted for a particular entity.

Furthermore, 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

The advantage of using the approach discussed in Section 5.3.3 to extract the semantic triples is that it avoids the stringent dependency of the 'subject' and 'object' phrases on their dependency tags in the dependency grammar (see Section 2.2). Instead, it uses dependency parsing only to extract the root verb in the sentence, and gets 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 and it did not always yield great results as the information extracted was heavily influenced by the grammatical structure of the sentence, which, in the case of news articles, can get very complex due to several nested dependencies. 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 was lost due to the entity being identified as the object dependency ('obj') in the grammar with a dependency tag such as `'iobj'`, `'dobj'` instead of having one of the subject tags ('subj') such as `'nsubj'`, `'csubj'`, `'nsubjpass'`, `'csubjpass'` (see Appendix Table B.1). The dependency sentence structure is commonly of type 'subj'-'verb'-'obj' or 'obj'-'verb'-'subj, where 'subj' and 'obj' include all the variants of the subject and object dependency tags respectively. Our current approach accommodates for both these grammatical structures as it only enforces that the 'subject' in the semantic triple is a named entity and therefore, it can have either 'subj' or 'obj' (including variants) as the dependency tag.

Knowledge graph of Semantic Triples

After extracting the qualifying semantic triples for each topic (in a semantic cluster), a force-directed (knowledge) graph of these triples is generated by the visualisation tool. Figure 5.6 shows the graph for Cluster 1 in Travel 2021 which has a single topic 'tourism, reopening, government'.

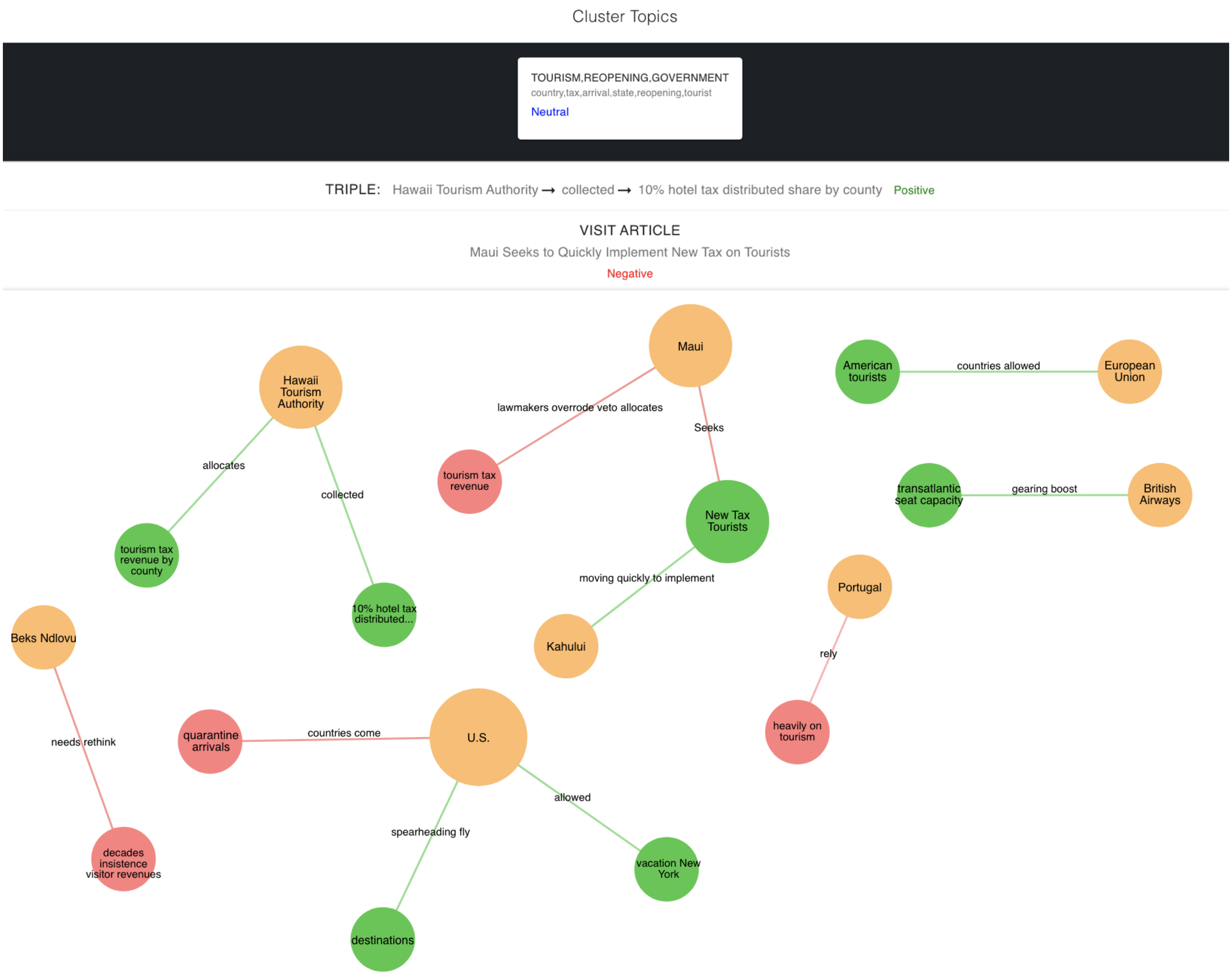


FIGURE 5.6 Knowledge Graph for topic 'tourism, reopening, government' in Cluster 1 in 'Travel 2021'

Figure 5.6 shows the graph generated by the visualisation tool for Cluster 1 (C1) which has a single topic 'tourism, reopening, government'. The 'subject' nodes (i.e., named entities) are in orange, the 'object' nodes are in green (indicating 'positive' sentiment) or red (indicating 'negative' sentiment) and the edges joining these represent the 'predicate'. Clicking on the predicate 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 it, thereby making 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 from a visual aspect. Additionally, we can see the interconnectivity of the nodes with the example of 'Maui' and 'Kahului' which both share the 'object' node 'New Tax Tourists'. This gives the insight that both 'Maui' and 'Kahului' which are islands in Hawaii have implemented a new tourist tax, which is further verified 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 to 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. For example, going back to the triples 'Maui' → 'seeks' → 'new tax tourists' and 'Kahului' → 'moving quickly to implement' → 'new tax tourists', we see that the former has a 'negative' sentiment (implied by the red 'predicate' line joining the nodes) 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 enthusiasm which are positive.

Limitations

One of the main limitations of the Semantic Triple Extraction Engine is that, given its verb-based approach, it extracts a single relation embedded in a sentence composed of a (root) verb phrase sandwiched between two entities of interest. News articles lend themselves to a sentence structure that is relatively complex with several subjects and objects in a single sentence. Often, trying the maximise information extraction from a single sentence can lead to redundancies and incoherent semantic triples. Therefore, the focus of this model to extract a single meaningful relation (i.e. triple) from a sentence provided that it mentions a named entity (of certain allowed types as discussed in Section 5.3.3).

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 has a huge impact (both in terms of accuracy and time) on the performance of the Semantic Triple 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 clustering and topic modelling.
2. Investigate the effect of different word embedding models on the quality of clustering.
3. Investigate the effect of different word embedding models and runtime environments on the performance (in terms of time efficiency).
4. Evaluate the legibility of the results from the Topic Extraction Engine and Semantic Triple Extraction Engine based on user feedback.

6.1 Quantitative evaluation of semantic clustering

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 (see Section 4.1) as a prerequisite for article vector generation for semantic clustering. These techniques included coreference resolution (CR), lemmatisation, stopword removal and named entity removal. The motivation behind this was to gauge the effect of these techniques on the silhouettes scores and incorporate those that the result in the most optimal scores, thereby resulting in improved 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 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, lemmatisation, named entity and stopword removal. A silhouette score near 1 indicates dense, nicely separated clusters, while a score near 0 indicates significant cluster overlap (i.e., the distance between them is insignificant). Figure 6.1 shows the general trend that omitting any of these pre-processing techniques deteriorates the quality of clustering, by resulting in poor silhouette scores which indicate overlapping clusters.

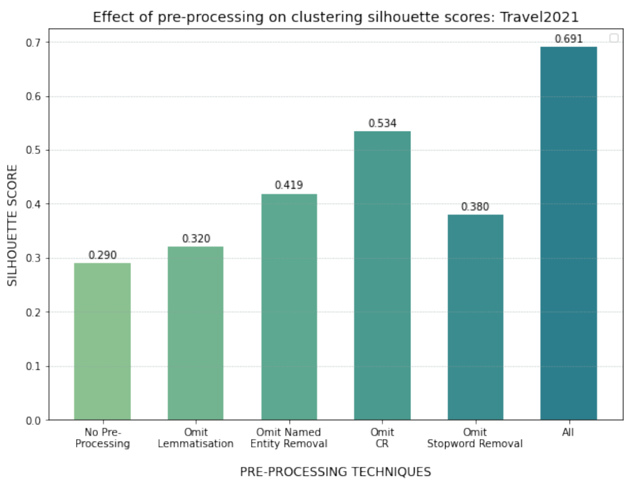


FIGURE 6.1 Silhouette scores of clusterings obtained by omitting different pre-processing techniques for input group 'Travel 2021'

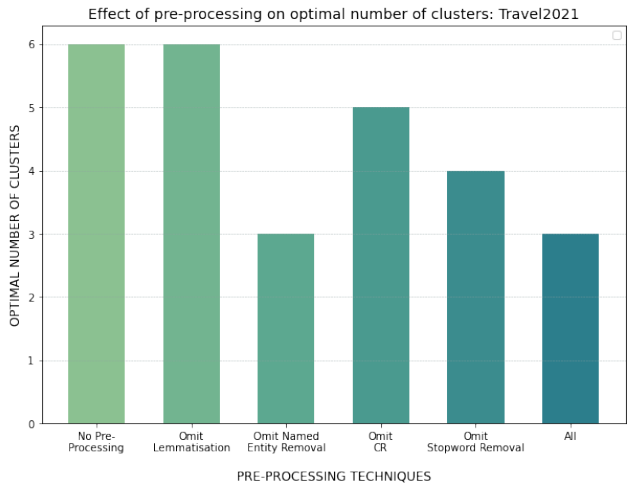


FIGURE 6.2 Optimal numbers of clusters obtained by omitting different pre-processing techniques for input group 'Travel 2021'

Another interesting observation as seen in Figure 6.2 is how pre-processing affects the optimal number of clusters. The Topic Extraction Engine determines this by running the KMeans(++) clustering algorithm with a range of different 'k' (see Section 4.2.5) and selecting the optimal 'k' (number of clusters) as the one resulting in the highest silhouette score. Generally, the aim is to get the best silhouette score with the smallest number of clusters. This is because we do not want the engine 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, therefore, we want the engine to achieve a high-level semantic grouping of articles into few, sufficiently large, distinct clusters. In the example shown in Figures 6.1–6.2, the input group 'Travel 2021' has a relatively low number of 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 latent topics extracted from these will be modelled on an extremely small corpus of about 2 articles and will most likely be incoherent. For the same data group, performing all the techniques mentioned (i.e., coreference resolution, lemmatisation, named entity and stopword removal), results in a much lower number of clusters: 3. In conclusion, the trend established by Figures 6.1–6.2 is that by performing the pre-processing techniques mentioned, the engine is able to cluster the articles in a Year-Category group in fewer as well as more distinct clusters indicated by the lower optimal number of clusters and higher silhouette scores.

6.1.2 Effect of filtering by POS tags on clustering

Another key decision made in the Topic Extraction Engine in an attempt 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 only use noun tokens to represent each article. As mentioned earlier, in order to obtain the optimal number of clusters (optimal 'k'), the engine performs KMeans(++) clustering with different values of 'k', choosing the one which results in the highest silhouette score. Figures 6.3–6.6 show the optimal cluster number and silhouette score for the Year-Category data groups: Business2020, Business2020, Politics2021 and Pursuit2021 respectively, comparing these values for when allowed POS tags are just 'NOUN' and when they contain 'NOUN', 'VERB', 'ADJ' and 'ADV'. These figures highlight the general trend that having a noun-only tokens list for article vector generation (for semantic clustering) gives a higher silhouette score with 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 the tokens list contain 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.

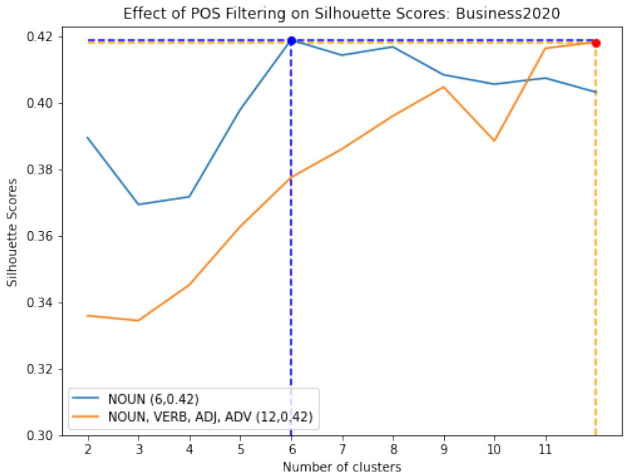


FIGURE 6.3 Optimal silhouette score and number of clusters for 'Business 2020'

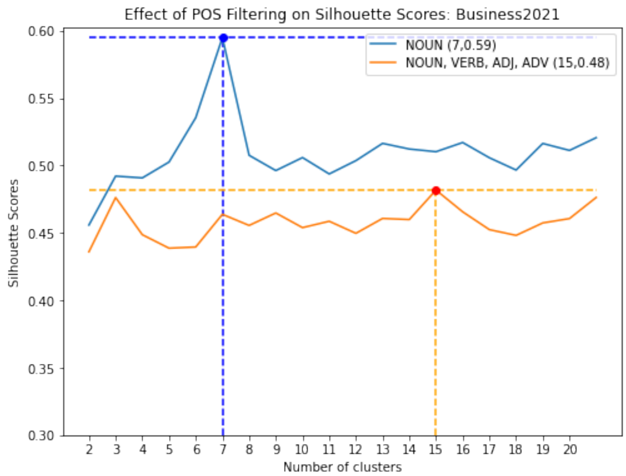


FIGURE 6.4 Optimal silhouette score and number of clusters for 'Business 2021'

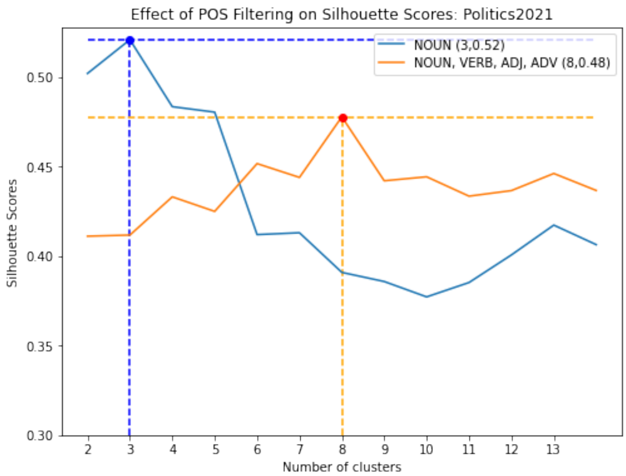


FIGURE 6.5 Optimal silhouette score and number of clusters for 'Politics 2021'

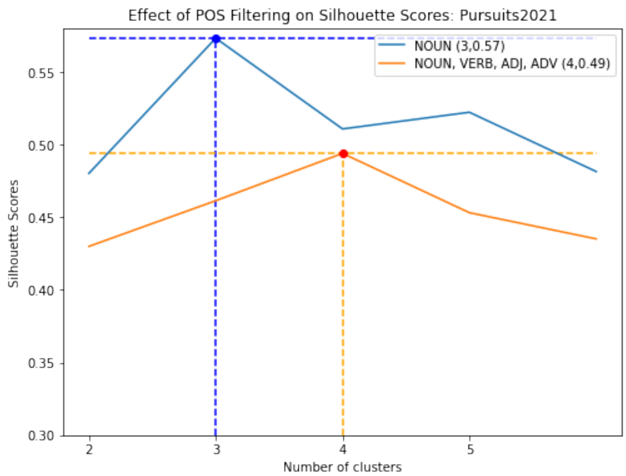


FIGURE 6.6 Optimal silhouette score and number of clusters for 'Prognosis 2021'

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 compared to 12 clusters with nouns, adjectives, verbs and adverbs. For example, *['coronavirus', 'aviation', 'industry', 'border', 'restriction', 'travel, 'quarantine']* is a more concise and general representation of an article as opposed to *['job', 'wipe', 'bad', 'come', 'worker', 'tell', 'worker', 'coronavirus', 'spend', 'lengthy', 'drastically', 'quarantine']*. Adjectives, adverbs and verbs are not meaningful on their own and their generic nature means they are not unique to articles. Including them in the `filtered_tokens` means they equally contribute to the article vectors as any of the distinctive nouns (e.g., 'pandemic') in the `filtered_tokens`, thereby making the article vector less precise. Furthermore, the more tokens used for generating the article vectors, the greater the potential to introduce noise, resulting in a higher variance among vectors, and making the corpus difficult to cluster. This results in a suboptimal clustering with more clusters than necessary as shown in the figures.

6.1.3 Effect of different word embedding techniques on clustering

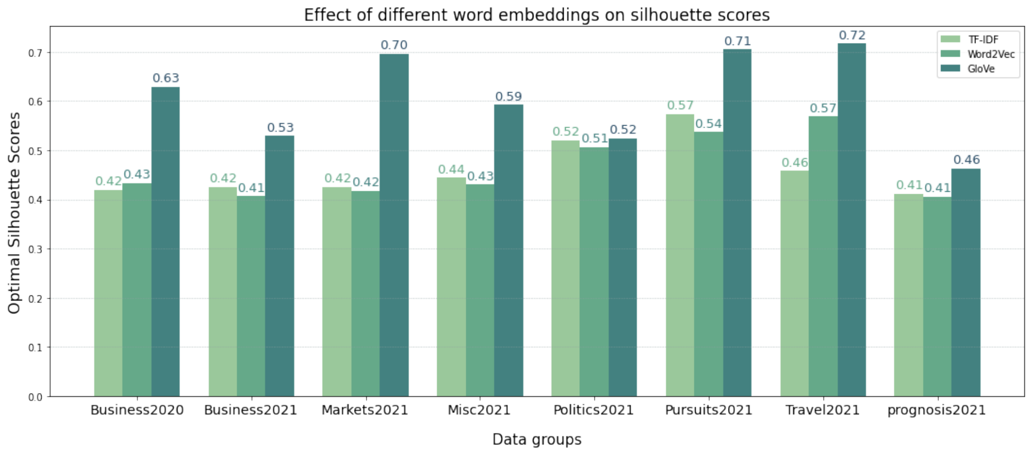


FIGURE 6.7 Effect of using TF-IDF, Word2Vec and GloVe as word embedding model on clustering silhouette scores

As discussed in Section 4.2.1, 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 thereby providing a quantitive measure of how the 'goodness' of semantic clustering is influenced by the word embedding approaches. As indicated by the bar 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 both to TF-IDF and word2Vec.

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 31.90% improvement when compared to word2Vec. Based on these results, it is evident that using GloVe is the optimal approach for word embedding for token vectorisation for semantic clustering. Additionally, we gain confidence in using GloVe as the silhouette scores for the clustering of most input groups are around 0.7 (as seen in Figure 6.7) which shows evidence of distinct clusters of articles. When clustering news articles, especially articles pertaining to a specific industry within a category, obtaining a perfect clustering with a silhouette score of 1 is not realistic, as there are bound to be common underlying themes resulting in some overlap.

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 Percentage improvement for silhouette scores when using GloVe over TF-IDF and Word2Vec for all Year-Category data groups

6.2 Quantitative evaluation of topic modelling

6.2.1 Effect of pre-processing on topic modelling

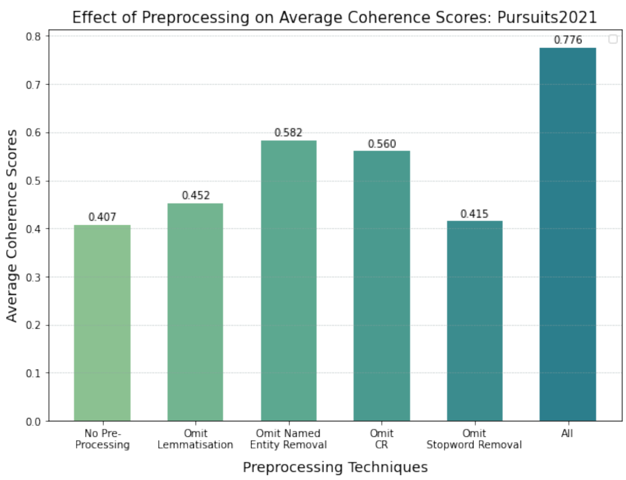


FIGURE 6.8 Average coherence score of all topic models for each semantic cluster in 'Pursuits 2021'

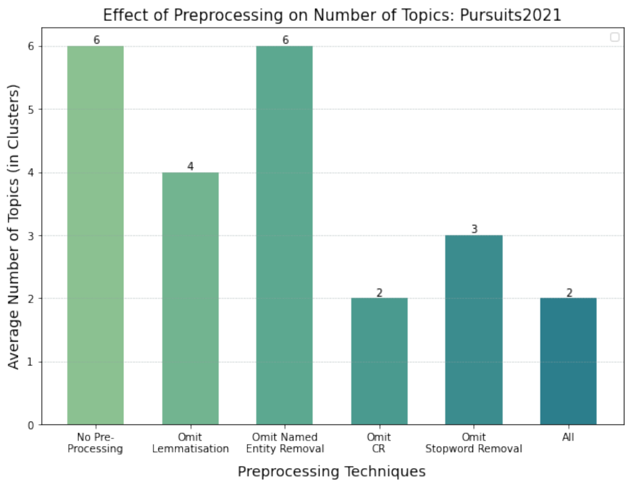


FIGURE 6.9 Average number of topics for each semantic cluster in 'Pursuits 2021'

Similar to Section 6.1.1, this section also focuses on the effect of pre-processing techniques but instead on the topics extracted from semantic clusters in Year-Category groups. The topics extracted from each clustering results in a topic coherence score and these are averaged over all clusters in a Year-Category input group to give an average coherence score (for the input group). The coherence score is a useful metric for evaluating the quality of topics extracted as it assesses the degree of semantic similarity between high-scoring terms in a single topic. In similar trends as seen for semantic clustering, Figures 6.8–6.9 show that applying all the pre-processing techniques (represented by the 'All' bar) results in a significantly higher average coherence score with less number of topics compared to other approaches.

In the case of omitting coreference resolution, the (average) coherence score falls (to 0.560) 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. Omitting lemmatisation similarly results in a drop in coherence (and an increase in number of topics) as the tokens are not resolved to their base forms affecting the frequency of common words (e.g. tourist and tourists will be treated as different keywords).

As mentioned in Section 4.1.2, stopword list contains the default spaCy stopwords (around 326 words), augmented with a custom list of frequently occurring unnecessary words and phrases specific to the input data such as 'told', 'said' (as news articles commonly quote entities, especially people) and 'airline', 'flight etc. (since our news articles are specific to the airline industry). The latter are removed as they can be thought of as overarching themes spanning many topics but not contributing to a distinct topic. The idea is that we do not want the model to fixate on these commonly appearing contextless stopwords and output junk topics, therefore removing them results in better topic coherence (0.776 for 'All' in Figure 6.8) compared to when they are retained which results in a much lower topic coherence of 0.415.

As established prior, named entity removal is done to allow the engine to find distinct topics within a cluster that are not dependent on these entities. For instance, let's assume that there is a huge overlap between 'pilot unions' and 'British Airways'. Retaining named entities in the input corpus could result in the model selecting 'British Airways' as a topic instead of 'pilot unions'. Figure 6.9, indicates that retaining named entities significantly increases the number of topics (6). Some of these topics are 'junk', implied by the lower average coherence score of 0.582 in Figure 6.8 compared to 0.776 for 'All'. This makes sense as named entities can be associated with several topics resulting in overlapping semantic clusters, which may be split (to improve silhouette score) but result in more (incoherent) topics for these split clusters, lowering the average coherence. As expected, omitting all pre-processing results in the lowest average coherence score of 0.401 and the highest number of topics (6).

6.2.2 Effect of filtering by POS tags on topic modelling

The `filtered_tokens` list (for each article) that was used to generate the article vectors for clustering, is passed to the LDA model to extract topics from each of the semantic clusters for a Year-Category group. To determine the effect of different allowed POS tags on the quality of topics extracted, the coherence score measure is used. Figure 6.10 shows that the noun-only tokens strategy performs better than the nouns, adjectives, verbs, and adverbs approach since the average coherence scores across all (Year-Category) groups are generally higher, with the noun-only method resulting in an 18.2% improvement in the coherence score. This is because, as mentioned in Section 6.1.2, there are several generic adverbs, adjectives and verbs that occur across articles in high frequency. Including these in the LDA corpus for topic extraction increases the potential for the model to select these as keywords (high-scoring terms) for a topic. This would result in low semantic similarity among the keywords in topics, resulting in poor coherence scores. Additionally, using these words would result in the model fitting to the noise, resulting in a higher number of incoherent 'junk' topics. This is apparent from Figures 6.10–6.11 which show that using 'NOUN, VERB, ADJ, ADV' results in lower coherence scores for higher number of topics.

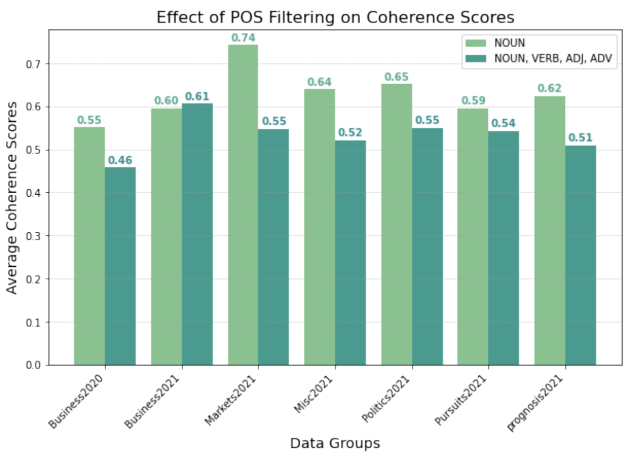


FIGURE 6.10 Average coherence score for all input data groups for different allowed POS tags

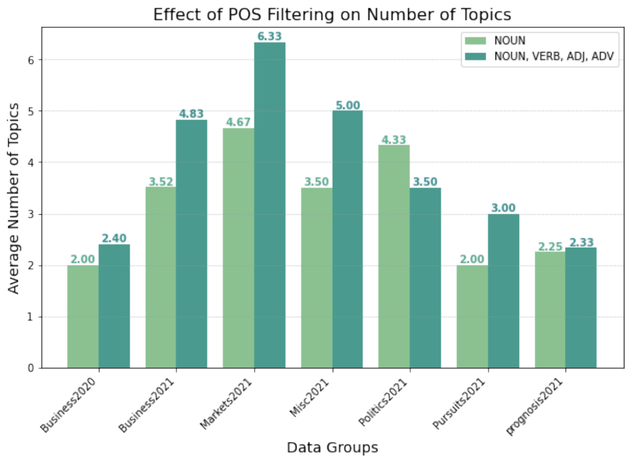


FIGURE 6.11 Average number of topics for all input data groups for different allowed POS tags

6.3 Time Efficiency

6.3.1 Comparing Runtimes in Various Environments

Data group	CPU	Multiprocessing (CPU)	Tesla T4 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 Time taken to run the Semantic Analysis Engine pipeline for all data groups(in seconds)

Table 6.2 shows the different runtimes for the pipeline when running on a CPU, with multiprocessing (on CPU) and GPU (Tesla T4). 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 for CPU is due to the expensive calls to the AllenNLP models, particularly the Fine-Grained NER, SpanBERT Coreference and RoBERTa Sentiment models (see Section 3.3.2), which unlike spaCy, are not built for production use and therefore not optimised for time. However, given that they are state-of-the-art models that achieve high performance for their respective NLP tasks, their use is extremely beneficial to the project. This motivated the use of multiprocessing and designing the engine architecture as a pipeline which takes an input Year-Category data group and outputs the results for the visualisation tool (see Figure 3.2), to allow for feasible CPU computation.

Processes are traditionally constrained to just having access to their own process memory, however, shared memory permits data structures to be shared between processes. The AllenNLP models in question are significantly large and RAM-intensive and therefore, having multiple copies of the models (one for each process) can result in the machine running out of memory. Therefore, it is crucial to store these models in shared memory to provide centralised access to all running processes and avoid redundant copies of models. This allowed running the tool on the 8 different data groups simultaneously, resulting in a performance gain in terms of time efficiency as seen in Table 6.2, which shows that using multiprocessing results in 2.55 times lower computational time compared to running with a 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.

6.3.2 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	547.92	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 effect of different word embedding models on the time taken to run the Semantic Analysis Engine pipeline on the Tesla T4 GPU for all data groups. 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. Furthermore, 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 when using `TF-IDF` and `GloVe` is trivial, however, given the 33.13% improvement (see Figure 6.7) in silhouette score of `GloVe` over `TF-IDF`, `GloVe` proves to be the best approach in terms of time efficiency and optimal clustering.

6.4 Qualitative 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, including members from Deep Search Labs to obtain a qualitative evaluation of visualised results from the Topic Extraction Engine and Semantic Triple Extraction Engine.

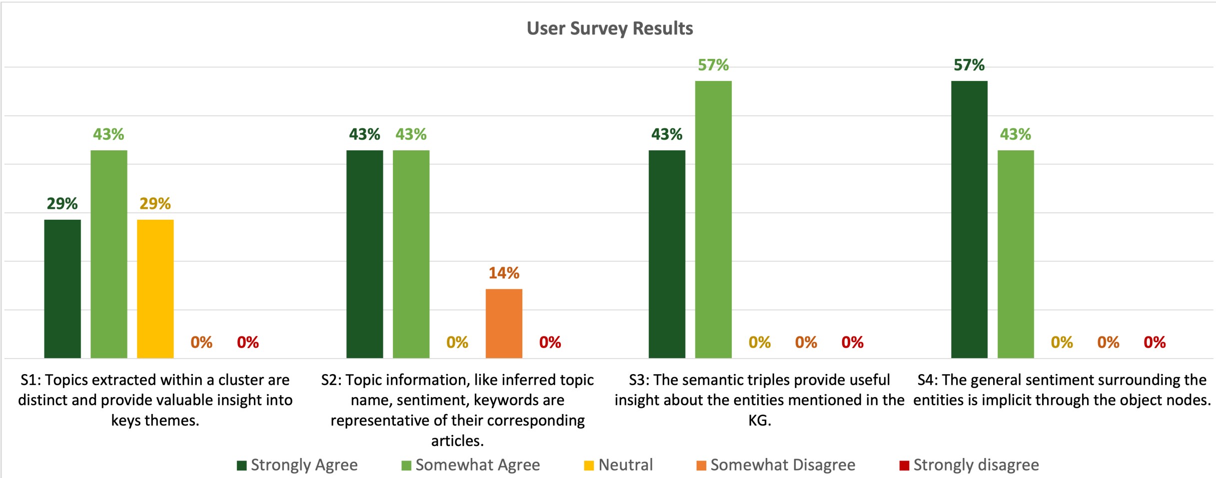


FIGURE 6.12 Response summary for the survey of 4 statements based on a 5-point agreement scheme, where 0 suggests strong disagreement and 5 suggests strong agreement

The evaluation was conducted in the form of a survey where the users were asked to score their agreement for 4 statements based on 5 point scale. In order to ensure uniformity to get a more accurate user perception of our solution, the respondents were asked to focus on the results for the Year-Category input group Travel 2021. Figure 6.12 shows the response summary for the 4 statements, where S1 and S2 focus on the cluster-topic graphs for displaying results from Topic Extraction Engine and S3 and S4 focus on the force-directed (knowledge) graph for displaying triples from the Semantic Triple Extraction Engine.

The responses show that over 72% of the respondents agree (with the other 29% remaining indifferent) with statement 1 (S1) that the Topic Extraction Engine is able to extract fairly distinct, coherent latent topics from the semantic clusters. For S2, while 86% respondents agree that the articles within the topic make sense based on the topic information, 14% (1 of 7) disagree. This is understandable, because as identified in Section 4.4.1, assigning an article a single dominant topic may not always be intuitive. The responses for S3 and S4 were overwhelming positive, with a 100% agreement for both statements verifying that the semantic triples in the graph provide useful insight about key entities (the relevance of entities depicted by the size of their nodes) and the general sentiment associated with these entities (depicted via green-red colour scheme for positive and negative sentiment respectively). Comparing the agreements responses for S1-S2 with S3-S4, we can infer the qualitative user evaluation gives greater confidence in the Semantic Triple Extraction Engine compared to Topic Extraction Engine, although both were received well as there were no strong disagreements with any of the statements, which generalise the objectives of the engines.

7 Conclusion

7.1 Summary of achievement

The aim of this paper was to propose a solution that provides a distilled and coherent representation of the news for a particular industry, offering valuable insight into the pertinent themes, events, and entities within a specific genre of news for a set period of time. Our work has succeeded in this through our Semantic Analysis Engine, which focuses on two main aspects: topic extraction and information retrieval through semantic triple extraction.

Our Topic Extraction engine performs cohesive semantic analysis of news for the airline industry by grouping the articles in a specific category (such as Travel, Business, etc.) into semantic clusters that are used to model topics (Chapter 4). This engine uses a combination of popular methods, such as KMeans cluster analysis and latent topic modelling using LDA, to provide a high-level semantic grouping through clustering as well as a finer-grained grouping through topic models. We incorporate a range of different natural language processing techniques such as coreference resolution, POS tag filtering, stopword removal, lemmatisation, etc., and explore different article embedding approaches to improve the 'quality' of semantic clustering by optimising for the silhouette score. Similarly, to model meaningful distinct topics, we optimise for the coherence scores.

Our Semantic Triple Extraction Engine exploits the rich morpho-syntactic structure of the English language and adopts a lexico-semantic approach using dependency parsing and POS tagging to provide a domain-independent solution (without using ontologies and an existing knowledge base) for triple extraction. The semantic triples extracted for each topic in a semantic cluster provide a minimal knowledge representation of the underlying information such as the 'key entities', their relationships to events and other entities, as well as the sentiment of the news surrounding them (Chapter 5).

Based on both quantitative results (such as silhouette and coherence scores) as well as qualitative user feedback (Chapter 6), the final product gives confidence in the extraction of relevant information through topics and semantic triples and displays these results through graphs generated by the visualisation tool in our end-to-end semantic analysis engine (see Figure 3.2). By carrying out extensive research into related works, as well as thorough investigation and evaluation to optimise the different components of our semantic analysis engine, we present a novel system that combines several NLP techniques to develop a user-friendly, end-to-end pipeline for visualising semantic analysis of news automatically.

7.2 Wider applications

The current solution for semantic analysis of news proposed in this paper focuses on a single industry, i.e., the airline industry. A wider application for this project could involve using the proposed tool across different industries combining to see how the news across industries affects each other and what common themes or topics, if any, can be extrapolated from the semantic analysis. This could be useful not only for the average consumer but also relevant from a business intelligence standpoint where companies can gain insight into, for instance, the market trends across industries such as airline, energy etc., or travel trends in hotel management and airline industries.

Additionally, collecting more data across different years would be useful as it would allow for a more substantial temporal semantic analysis of the news associated with an industry, where the user can infer how the topics and content of the news (through triples) change through different periods of time.

7.3 Future Work

1. **Sentence relevance for article summarisation:** The current approach for generating the semantic clusters and topics uses the 'article intros' rather than the entire article (to avoid redundancy). The 'intros' consist of the first 8 sentences of the article as the introductions of news articles generally provide a good summary whilst retaining sufficient context. An alternative approach to the 'article intros' involves obtaining the extractive text summary of the articles by selecting the most relevant sentences in the corpus through sentence ranking algorithms [70] [71].
2. **Linking to existing Knowledge Base for Disambiguation and Data Augmentation:** The current approach simply relies on the entities extracted from processing raw text from the articles to generate the knowledge graph (of semantic triples). This has the potential of introducing redundancy as the aliases of named entities (e.g., BA and British Airways) do not get resolved to the same entity. A beneficial improvement for the future would be to use named entity linking (NEL) [45] and store the semantic triples in an RDF triplestore where the 'subject entity' is represented with a unique International Resource Identifier (IRI) [34]. This allows for Named Entity Disambiguation (see Section 2.5.1) by removing the dependency of subject nodes on the aliases of named entities.
3. **Experiment with ELMo embeddings for document vectorisation.** The challenges with approaches such as Word2Vec and GloVe word embeddings are that they provide a single context-independent representation for a word and struggle with "out-of-vocabulary (OOV) words" [26], i.e., words that were never encountered prior will often be represented as a random vector which is not ideal. A potential improvement over this would be to use ELMo to allow context-dependent word embeddings (see Section 2.4.2).

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 [72]

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 [73]

C NER Semantic Types

Entity Type	Meaning	Examples
CARDINAL	Numerals	'200,00', 'two'
DATE	Absolute, relative dates, including festivals	'3^{rd} September', 'two days', 'Christmas'
EVENT	Named hurricanes, battles, sporting events	'Football World Cup', 'Hurricane Katrina'
FAC	Airports, building, bridges, highways, etc.	'London Bridge', 'Heathrow'
GPE	Geo-Political Entities such as countries, cities, states	'London', 'Alaska', 'UK'
LANGUAGE	Any named language	'English', 'French'
LAW	Named documents made into laws	'The First Amendment', 'Newton Laws of Motion'
LOC	Non-GPE locations, mountain ranges, bodies of water etc.	'Thames', 'Himalayas'
MONEY	Monetary values	'over three dollars', '$300'
MISC	Miscellaneous, unclassified entities	
NORP	Nationalities or Religious or Political groups	'Indians', 'Christians'
ORDINAL	Number indicating position	'first', 'second'
ORG	Organisations, including companies, institutions etc.	Microsoft, PayPal
PERCENT	Percentage	'three percent', '40%'
PERSON	Name of person, including fictional	'Harry Potter', 'Obama'
PRODUCT	Objects, vehicles, company products etc.	'Alexa', 'Siri'
QUANTITY	Measurements, including weights or distance	'200m', 'three kilograms'
TIME	Time periods smaller than a day	'this minute', '2 hours'
WORK_OF_ART	Include books, songs, paintings etc.	'Mona Lisa', 'The Legend of Zelda'

TABLE C.1 Named Entity Types extracted by Fine-Grained NER [55]

D Common Regular Expressions

Operator	**Meaning**
.	Match any character (except line break)
?	Match zero or one times
*	Match zero or more times
+	Match one or more times
$A \mid B$	OR operand, match A or B
^	Anchor, matches from start of string
$	Anchor, matches from end of string
{n}	Quantifier, match exactly ‘n’ times
{n, m}	Quantifier, match between ‘n’ to ‘m’ times
[AB]	Character group, match characters which are A and/or B
[^AB]	Negated character group, match between characters not A or B

TABLE D.1 Common regular expression (i.e., Regex) syntax

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] SpaCy: Linguistic Features; 2022. Available from: https://spacy.io/usage/linguistic-features.

[20] 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.

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

[22] 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.

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

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

[25] 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.

[26] Maslennikova E. ELMo Word Representations For News Protection. In: CLEF (Working Notes); 2019. .

[27] Rajaraman A, Ullman JD. Mining of massive datasets. Cambridge University Press; 2011.

[28] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781. 2013;.

[29] Smith NA. Contextual word representations: A contextual introduction. arXiv preprint arXiv:190206006. 2019;.

[30] 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.

[31] Melamud O, Goldberger J, Dagan I. context2vec: Learning generic context embedding with bidirectional lstm. In: Proceedings of the 20th SIGNLL conference on computational natural language learning; 2016. p. 51–61.

[32] Jurafsky D, Martin J. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. vol. 2; 2008.

[33] Lašek I, Vojtáš P. Various approaches to text representation for named entity disambiguation. International Journal of Web Information Systems. 2013;.

[34] Dürst M, Suignard M. Internationalized resource identifiers (IRIs); 2005.

[35] Hachey B, Radford W, Curran JR. Graph-based named entity linking with wikipedia. In: International conference on web information systems engineering. Springer; 2011. p. 213–226.

[36] Rinaldi AM, Russo C, Tommasino C. A semantic approach for document classification using deep neural networks and multimedia knowledge graph. Expert Systems with Applications. 2021;169:114320.

[37] Blei DM, Ng AY, Jordan MI. Latent dirichlet allocation. Journal of machine Learning research. 2003;3(Jan):993–1022.

[38] Hisano R, Sornette D, Mizuno T, Ohnishi T, Watanabe T. High quality topic extraction from business news explains abnormal financial market volatility. PloS one. 2013;8(6):e64846.

[39] Liu B. Sentiment analysis and opinion mining. Synthesis lectures on human language technologies. 2012;5(1):1–167.

[40] Kanakaraj M, Guddeti RMR. NLP based sentiment analysis on Twitter data using ensemble classifiers. In: 2015 3Rd international conference on signal processing, communication and networking (ICSCN). IEEE; 2015. p. 1–5.

[41] Benamara F, Cesarano C, Picariello A, Recupero DR, Subrahmanian VS. Sentiment analysis: Adjectives and adverbs are better than adjectives alone. ICWSM. 2007;7:203–206.

[42] Godbole N, Srinivasaiah M, Skiena S. Large-Scale Sentiment Analysis for News and Blogs. Icwsm. 2007;7(21):219–222.

[43] Balahur A, Steinberger R, Kabadjov M, Zavarella V, Van Der Goot E, Halkia M, et al. Sentiment analysis in the news. arXiv preprint arXiv:13096202. 2013;.

[44] 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`.

[45] Yan J, Wang C, Cheng W, Gao M, Zhou A. A retrospective of knowledge graphs. Frontiers of Computer Science. 2018;12(1):55–74.

[46] Consortium WWW, et al. RDF 1.1 concepts and abstract syntax. 2014;.

[47] Berven A, Christensen OA, Moldeklev S, Opdahl AL, Villanger KJ. A knowledge-graph platform for newsrooms. Computers in Industry. 2020;123:103321.

[48] Ehrlinger L, Wöß W. Towards a definition of knowledge graphs. SEMANTiCS (Posters, Demos, SuCCESS). 2016;48(1-4):2.

[49] A Guide to Knowledge Graphs; 2022. Available from: `https://towardsdatascience.com/a-guide-to-the-knowledge-graphs-bfb5c4027`.

[50] Bhalla K. Three Principles of Good Visualization — CU Management; 2022. Available from: `https://www.cumanagement.com/blogs/2018/11/05/three-principles-good-visualization`.

[51] The 12 Basic Principles Of Data Visualization; 2022. Available from: `https://datafloq.com/read/the-12-basic-principles-of-data-visualization/`.

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

[53] 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/`.

[54] AllenNLP Library; 2022. Available from: `https://allenai.org/allennlp/software/allennlp-library`.

[55] 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`.

[56] Cloud Application Platform: Heroku; 2022. Available from: `https://www.heroku.com/`.

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

[58] 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.

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

[60] Muflikhah L, Baharudin B. Document Clustering Using Concept Space and Cosine Similarity Measurement. In: 2009 International Conference on Computer Technology and Development. vol. 1; 2009. p. 58–62.

[61] 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. .

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

[63] 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.

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

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

[66] 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.

[67] 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.

[68] 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.

[69] 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.

[70] Ježek K, Steinberger J. Automatic text summarization (the state of the art 2007 and new challenges). In: Proceedings of Znalosti. Citeseer; 2008. p. 1–12.

[71] Madhuri J, Kumar RG. Extractive text summarization using sentence ranking. In: 2019 International Conference on Data Science and Communication (IconDSC). IEEE; 2019. p. 1–3.

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

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