Improving Use of Implicit Content Network Models for Social Media News Propagation Analysis through Large-Scale Language Models

by

Jiachen Li

A thesis submitted in total fulfillment for the
degree of Master of Computer Science

in the
Supervision by Professor Richard O. Sinnott
School of Computing and Information Systems
THE UNIVERSITY OF MELBOURNE

November 2023

THE UNIVERSITY OF MELBOURNE

Abstract

Supervision by Professor Richard O. Sinnott
School of Computing and Information Systems

Master of Computer Science

by Jiachen Li

Social media is increasingly replacing traditional news media for daily news updates. Indeed for many it has become the primary source for news and information. The fast flowing nature of social media has supported its dominant position today with news often reported in near real time. Unfortunately, this dynamic nature of social media has been a double-edged sword that allows inaccurate or, even worse, deliberately fake news to spread and potentially harm society by manipulating people's opinions and ultimately their decisions. In this context it is vital to analyse news information dispersal patterns in-depth to provide end-users with more comprehensive information on news propagation patterns thereby allowing them to think critically about how much trust they might place in a given social media news post. There has been a body of literature proposing models for tracking news dispersal across traditional news media through modelling implicit content networks. However, the domain of social media news is dramatically different from traditional news media. In this research, we propose to specialise and improve implicit content network models using Twitter corpora, now called "X". We consider the performance, efficiency, and characteristics of different text embedding methods incorporating different pre-processing pipelines. We explore large language models (LLM) for dispersal patterns since the direct/immediate sharing of the same content may not always take place. The novelty of the research is to explore Twitter data at scale and use of specialised text pre-processing techniques combined with LLM. We also consider the scalability and performance of the system on different physical hardware and associated software environments. The research serves as a solid step towards the future ultimate goals of building the cloud infrastructure for intra- and inter-social media platforms news dispersal analysis.

Declaration of Authorship

I certify that:

- This thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university, and that to the best of my knowledge and belief, it does not contain any material previously published or written by another person where due reference is not made in the text.
- Where necessary, I have received clearance for this research from the University's Ethics Committee and have submitted all required data to the School.
- The thesis is 25997 words in length (excluding text in images, tables, bibliographies and appendices).

Signed: 

Date: 30/10/2023

Acknowledgements

I would like to begin by acknowledging the traditional owners of the land on which this research was conducted, the Wurundjeri people of the Kulin Nation, and pay my respects to their Elders past, present and emerging.

I would like to express my sincere gratitude to my supervisor, Professor Richard O. Sinnott, for his professionalism and dedication. He has provided extremely valuable insights into the field of cloud computing and big data analytics. Without his tremendous effort, this research would not have been accomplished in such a comprehensive way.

I want to thank Senior Lecturer Jianzhong Qi and his PhD student Shuzhi Gong from the School of Computing and Information Systems at the University of Melbourne for providing essential guidance toward the algorithm and benchmark metric aspect of the Natural Language Processing research field.

I want to thank Luca Morandini and Qi Li from the Melbourne eResearch Group led by Richard O. Sinnott for supporting me in web data crawling and structural design of the distributed cloud computing system.

I want to thank Senior Lecturer Jey Han Lau from the School of Computing and Information Systems at the University of Melbourne for his coordination in Master of Computer Science to guide me through the administrative aspect of the research project and his COMP90042 Natural Language Processing class, which established my interest and foundation knowledge for research in the Natural Language Processing field.

Finally, I would like to thank my parents, Hong Zhao and Feng Li, for financially and mentally supporting me throughout the research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the past few years, the rise of social media has been one of the most significant global trends. Thanks to the ease of use and rapid flow of information on social media, it has become integrated into people's daily lives to connect with others and the information they receive. Social media is no longer limited to being a source of entertainment but is also used to consume news as a source of information. A Survey conducted by Pew Research Center suggests that 53% of people, which is the majority of adults in the United States, often or sometimes get news from social media in 2020 [14], while four years ago, in 2016, only 44% of adults in the United States, often or sometimes obtained their news from social media [15]. The survey illustrates that social media is rapidly but ineluctably replacing traditional web-based internet news to become the primary source of news content for most people because it can offer more personalised, diverse, and interactive news content. The same claim was made by Elon Musk [1] on the left side of the Figure 1.1. In fact, journalists who are responsible for producing traditional news based on paper or web media rely on social media for sourcing or verifying news. In the study conducted by Zhang and Li, 255 participant journalists working in Hong Kong, on average, 54.8% agree they use social media for news sourcing, and 48.8% agree they use social media for news verification [16]. This phenomenon has also been pointed out by Elon[2] shown on the right side of the Figure 1.1

However, due to the accessibility and fast information flow, social media poses various challenges and risks, including but not limited to misinformation, addiction, privacy concerns and cyber-bullying. Among all the challenges and risks, misinformation is arguably today's society's most severe challenge. Misinformation on social media can result in grievous damage to public health, democracy, and social cohesion, as people over-rely on the news and information obtained from social media to guide their daily decisions. The study conducted by Karlsen and Aalberg suggests that the existence of

FIGURE 1.1: The claims made by Elon Musk, owner of the X (Twitter) platform, regarding the superiority of social media over "legacy media" as the source of News [1][2]

misinformation on social media can influence their perceptions, resulting in people not trusting any news, including news from traditional papers or web-based media [17]. Besides the trust issues, the misinformation on social media may also manipulate emotions to polarise opinions and potentially inciting violence.

In response to hazardous misinformation challenges on social media, numerous research efforts have been conducted in the Natural Language Processing (NLP) field. One approach is rumour detection, which uses various features from social media posts and posting accounts to classify posts as rumour or truth. However, the approach has limitations, especially in outputting limited and potentially biased information. First, the model only provides binary information that may be too absolute to ask the end user to trust or disregard given that there is no ability to assess complex real-world situations such as manipulated misleading information. The limited binary output itself can be biased because the ground truth label of a statement being a rumour or not from training data itself may not be completely accurate and objective. This is especially significant because the majority of posts on social media are non-factual statements, such as the prediction of the future, that are nearly impossible to verify accurately and objectively. Therefore, the model can be easily affected by human factor. It is essential to avoid bias when deciding if a claim is true or false or a merely rumour, e.g., Gatta et al. [18] focused on false claims on Twitter during the Russia-Ukraine conflict. Their rumour detection model was trained based on verified news from news agencies and fact-checking organizations such as Politifact [19] with a clear preference for trusting English sources over other languages and Western governments from other political bodies. The paper's authors were all from Western countries, namely Italy and the United States, under the influence of a specific set of ideologies and taking a specific position in the geo-political

matters. Therefore, their academic work would inevitably be subjective and biased to some degree.

In response to this, the study of news dispersal pattern analysis has emerged. By tracking information propagation and visualising how information is spread and modified across different social media posts, the big picture of the news dissemination networks can be shown to users. With the visualised network, users can retrieve visual evidence to help them more objectively detect rumours by themselves. While there has been limited research on tracking news dispersal on social media platforms, there have been multiple approaches to tracking the information flow on web-based news blogs. One state-of-the-art implicit content network model proposed by Joshi [8] took the content diffusion approach to track information based on semantic similarity of posts. This serves as the reference model for this research.

In this thesis, there are three research objectives. The first is to improve the performance of the implicit content network model preferences and efficiency through use of novel large-scale pre-trained language models to better capture the semantic information included in text. The second is to optimise the model to the social media news post language domain. The third is to introduce a new measurement to evaluate the potential of the model to be applied to the parallel computing environments. With these goals, experiments are split into three parts. The first experiment evaluates the performance of a range of text embedding approaches using a combination of text pre-processing methods on the social media data using a semantic textual similarity task. The second experiment evaluates the efficiency of text embedding methods with optimal pre-processing pipelines on the realised reference model implementation. The third experiment compares the execution time of the model running in serial and parallel on various hardware and software environments to compute the standardised measurement of parallel computing capability based on Amdahl's [20] and Gustafson's [21] Law.

The rest of the thesis is structured as follows. The literature review in Chapter 2 provides an in-depth review of the theoretical knowledge and the historical development process of text pre-processing, text embedding, and information dispersal tracking. The review serves as the preliminary knowledge needed to identify the knowledge gap and potential improvement that can ultimately shape the research goals. In Chapter 3, the methodology and experimental setting, dataset, and model implementation are considered as well as the importance on the reliability and reproducibility of the result. The detailed experiments are presented in Chapters 4-6. The impact of the proposed model is discussed in Chapter 7. We draw conclusions on the work as a whole in Chapter 8 and identify future areas of work.

Chapter 2

Literature Review

The literature review serves as the foundation of this research to provide a comprehensive examination of existing research and knowledge that is relevant to the topic of research being investigated. In the chapter on literature review, we embark on a journey through the scholarly landscape to gain insights, identify gaps, and contextualize the research problem addressed in this study. This research is an interdisciplinary study that contains the experiment in Natual Language Processing (NLP) algorithms and models for tracking information dispersal under the setting of English social media news language domain as well as exploration of the parallel computing capability of the system in the distributed and cloud computing setting. This literature review will provide a comprehensive review through the domain of NLP to unravel the intricacies of the algorithm and models behind News dispersal pattern analysis. In this thesis, we propose to improve the performance and parallel computability of the news dispersal tracking methods proposed in previous literature by utilising the trending large-scale pre-trained language model for word embedding with corresponding customised text pre-processing techniques. Therefore, before the extensive investigation of information dispersal tracking literature, to distribute the knowledge to help readers with limited proficiency in the research domain to understand complex NLP techniques and terminologies discussed throughout the thesis, it is paramount to build a strong foundation by exploring two aspects of NLP: text pre-processing and text embedding. It is also worth noting that this thesis is taking the formatting approach of including all the non-original ideas in the literature review chapter, including the explanation of techniques and models used in the methodology chapter, which results in the literature review chapter appearing to longer than other formatting approaches while methodology appears shorter but the ratio of the actual content is identical regardless.

In the first section of this chapter, text pre-processing, the initial and foundation step of the NLP pipeline, will be explored. Text pre-processing is aimed at transforming raw textual data into structured, analysable information in order to not only improve the quality of input data but also enhance the performance of subsequent NLP algorithms. In this literature review, we will explain the diverse methods and tools employed in each step of the text pre-processing pipeline.

Word embedding as the cornerstone of NLP will be investigated in the second section of the chapter. Word embedding is the strategy for representing words as dense vectors in continuous vector space, while the more modern approach is capable of capturing semantic relationships and contextual nuances in language. It is vital to demonstrate the idea and implementation of the word embedding technique in different historical stages because the choice of word embedding method can impact the performance of NLP applications, including those related to news dispersal pattern tracking and analysis.

After navigating through the first two sections of this literature review, the readers are expected to gain a comprehensive understanding of the foundational elements of text pre-processing and word embedding. These concepts serve as essential building blocks for the subsequent exploration of news dispersal tracking algorithms in the third section of the chapter. The previous proposed News dispersal tracking method will be reviewed in chronological order with an emphasis on text pre-processing and word embedding strategies design choice with the consideration of the availability of text pre-processing and word embedding strategies at the historical stage when the original paper was published.

2.1 Text pre-processing

In the NLP pipeline, text pre-processing as the first and foundation layer has been essential for understanding and analysing the language effectively. The text data will be cleaned, formatted, and transformed to be more suitable for downstream tasks, usually machine learning models, to perform more effectively.

Before further exploring the topic, it is essential to note that text pre-processing is a general term describing a vast family of tasks and algorithms that serve the purpose of cleaning and transforming data into a more suitable format for further processing. It was defined differently in terms of inclusion, exclusion, or taxonomy of different tasks by different scholars. Therefore, to set a standard, the knowledge discussed in this section is based on the third edition draft of the book “Speech and Language Processing: An Introduction to NLP, Computational Linguistics, and Speech Recognition” by

Daniel Jurafsky and Martin [22] accompany with the review on pre-processing tools and techniques on textual data by Kathuria et al. [23]. Based on the knowledge hierarchy introduced in the book and the review, the standard pipeline for text pre-processing contains five steps, which are noisy removal, sentence segmentation, word tokenisation, word normalisation, and stopword removal.

The section is set to start by explaining the motivation and necessity of researching text pre-processing strategies, followed by a detailed explanation of five step of the text pre-processing pipeline mentioned above.

2.1.1 Motivation

There have been mainly two theories supporting the necessity of text processing in NLP tasks. The first theory suggesting the reason for having text pre-processing for NLP problems is that NLP is a subclass of machine learning problem, so text is one form of data obtained from the real world that naturally suffers from noise that needs to be pre-processed for smoother further analysis. The paper by Kathuria et al. [23] claims that text in electronic form is generally "incomplete, inconsistent, contains errors or outliers", and more than 80% of such text data are unstructured or semi-structured. The theory of the noisy nature of text data in NLP problems is also supported by the research done by Dey and Haque [24] on text mining tasks from noisy and unstructured text data collected from online communication. The noisy data challenge is also not just limited to the NLP field but encountered by all machine learning-related research, such as the challenge faced by Wu and Zhu [25] on error-aware data mining with real-world collected inaccurate data because text mining is a subcategory of data mining tasks. Therefore, to reduce the negative effect on downstream machine learning tasks by noise, data pre-processing is an unavoidable step.

The second theory suggested by Biesialska et al. [26] is on the natural language itself. Natural language, especially the language of English, which will be the scope of this research, is compositional. The letters are composited into words, and words are composited into sentences, paragraphs, documents, and eventually, corpus. Humans have been good at decomposing language naturally. Therefore, computers should do the same to decompose the language before further understanding and processing the language.

The necessity of text pre-processing in NLP tasks is not only supported by theories but also backed up by practical experiment results. There have been numerous researches suggesting the decision on text pre-processing can have a significant effect on the performance of downstream machine models for both supervised learning and non-supervised learning.

The research done by Uysal and Gunal [27] has experimented with the impact of text pre-processing on text classification tasks in multiple aspects, including but not limited to classification accuracy. The classification tasks can be solved through supervised machine-learning models. The research investigated the effect of different combinations of major text pre-processing techniques on text classification in the domain of email and news in both English and Turkish. The research suggests that instead of using all of the text-processing techniques available or not doing any pre-processing, using a suitable combination of text pre-processing techniques for a specific domain and language would enhance the performance of the downstream model significantly. This research has encouraged us to explore the suitable combination for English social media news posts language domain.

Another research done by Denny and Spirling [28] focuses on unsupervised quantitative analysis of text-as-data research in the domain of political science. To bring attention to the pre-processing techniques. Authors suggest the decision on the text pre-processing process can impact the result of real-world used models on real-world collected data profoundly. They proposed a statistical procedure along with software that experimented with different orders and combinations of text pre-processing methodologies to unveil not only that text pre-processing is essential but also that it has to be done with caution to avoid negative effects.

On account of the importance of text pre-processing in NLP, all of the research on similar topics of information dispersal analysis has included their implementation of text pre-processing in detail with the explanation of intuition backed up by experiment results and justified by theory. However, despite the tremendous effort of predecessors, this research has to reinvestigate and experiment with different combinations of text pre-processing strategies meticulously because text pre-processing is not a one-size-fits-all process and needs to be customised and tailored to specific requirements of the NLP task and adapting to the domain of research. The research aimed to accomplish the same NLP task of clustering the news on the same topic and tracking its propagation. However, different text embedding strategies are used in different approaches for calculating the text vector used for text similarity measurement. Different text embedding strategies work entirely differently and post different specific requirements for the pre-processed input text, especially in the level of noise tolerance. This is because the more sophisticated model may be capable of making use of text that is less informative and almost is completed massy. Such text would usually be pruned in the text pre-processing stage. The reason this could happen is backed up by the theory that, in essence, text pre-processing is making a compromise by treading less critical information for less noise and better model efficiency, which is dramatically different from the majority of other aspects of NLP, namely the LLMs that have consistently improved their performance

under development regardless of the language domain and other problem settings. On the other hand, this research is researching the language domain of social media posts, which is significantly different compared to the more traditional online news articles from previous research in various aspects. The differences between social media posts and traditional news articles include but are not limited to their content and topic, length and structure, language and tone, and purpose and audience. All of those differences significantly impact the text pre-processing decision in development.

2.1.2 Noisy Removal

The first step is noisy removal, which sometimes may also be referred to as unwanted formatting removal. The process involves eliminating or cleaning up formatting elements and artifacts that the downstream model is incapable of making use of to make text data more suitable for analysis and modelling. Depending on the data source, the unwanted data format can be the HTML format from web-crawled data, the URL from the blog post, non-English language form multilingual text, emoji from informal writing, and special mathematical symbols from professional writing. Because data in that unwanted format would typically be less informative or even act as noisy to confuse the model. Removing such data would not only save up storage and memory space for faster processing speed but also reduce the chance of unexpected formatted text confusing the model, resulting in a negative impact on accuracy. However, the design choice of the scope of noisy text needs to be done with caution because overdoing the noisy removal may result in potential information loss and can also reduce the model's accuracy. In a later paragraph on text normalisation, the unsupported text format handler used to maximise the retention of information will be discussed.

2.1.3 Sentence Segmentation

The second step is sentence segmentation, also called sentence tokenisation or boundary detection. As the name suggests, it is the process of breaking large chunks of text into smaller, more manageable individual sentences for further analysis. The evolution of sentence segmentation reflects the development and fundamental ideas of NLP involving three main stages: rule-based, lexicon-based, and machine learning-based. An initial approach is a rule-based approach that breaks every sentence on sentence-ending punctuation, including period, exclamation mark, and question mark. The technique is sufficient in sentence segmentation for the majority of languages; however, the research domain specified language, English, has been excluded from the exhausted list. English has the characteristic of using sentence-ending punctuation for abbreviations,

which would be falsely taken as the separation gap between two sentences. To handle such edge cases, a combination of the regular expression approach and the lexicons approach is used. The regular expression "([.?!][A-Z])" is used to ensure only breaking the sentence when the punctuation is followed by a capital letter that starts a new sentence to avoid breaking abbreviations followed by lowercase letters. However, the approach alone can not handle the abbreviation followed by entity names starting from capital letters, such as names. Therefore, the lexicon recording all the n is used to composite the flaw to avoid breaking the known abbreviation. However, despite how many benefits the lexicon has promised, the lexicon has had the fundamental drawback of requiring extensive human effort to update the dictionary in time as the language evolves, which has been extremely impractical in the era of the internet, when languages are changing faster than ever. Therefore, similarly to other NLP tasks, the ultimate solution is to exploit the machine learning approach. The modern machine learning method trains a binary classifier to classify if a sentence ending punctuation is indeed ending a sentence or part of an abbreviation based on features including but not limited to word, word shape, and part of speech tag before and after the punctuation.

2.1.4 Tokenisation

Followed by sentence segmentation, breaking down paragraphs into sentences, tokenisation is the next task to further break down the text into tokens, which are defined as single, meaningful smaller units of text to enable downstream models to understand and process the text more effectively while preserving the underlying linguistic structure. Implementation-wise, researchers such as Bird et al. [29] have provided extensive guidelines and tools like NLTK for this purpose in their book. As discussed, English has been naturally more challenging for sentence segmentation due to the existence of the abbreviation compared to other languages such as Chinese. However, on the flip side, since English has clearer word boundaries, tokens in English are mostly words, and the tokenisation process is to break words by white space. The tokenisation process does bring challenges in handling contractions, hyphenated words, and abbreviations, so dependent on the use case, subword tokenisation is the alternative approach to break words into even smaller units such as subword pieces, morphemes or even characters. It is worth mentioning one of the significant algorithms used for tokenisation, the Max Match algorithm, is used to greedily match the longest subword from the text found in the dictionary. The algorithm is broadly used for three different situations. 1: it can be used for Asian language tokenisation along with a dictionary of words in such language, as no space is used for easy breaks between words. 2: the algorithm may also used for English subword tokenisation with a data-informed dictionary generated from the corpus

by the Byte-Pair Encoding algorithm that merges frequent character pairs iteratively. 3: the algorithm can also be used for tokenisation with a given dictionary under the rare situation of English words are not separated by space correctly. For example, Twitter hashtags that properly squish the words together with no space in between could use this algorithm to break up tokens.

2.1.5 Normalisation

Word normalisation is a text pre-processing technique to transfer the token or words into their base or canonical form in order to reduce word variation by standardising the representation of similar words with the same meaning. Eventually, by merging the various forms of similar words into a single representation to reduce the vocabulary size, the word normalisation process can improve the efficiency and accuracy of downstream tasks. The normalisation process generally contains various aspects, including case lowering, spelling correction, abbreviation expansion, special text handling and morphology removal.

The first task, case lowering, is to convert every character of the token into lowercase to map the text into case-insensitive form to merge the same word with or without capital cases. Spelling correction is the second task that corrects spelling mistakes by replacing misspelled words with correct spelling that matches other instances of the intended word. To some extent, this could also include the processing of unifying the language into the specific flavour of English, usually to unify all the words in a text to American English spelling or Britain English spelling, for example, “normalise” or “normalize”. The third task, abbreviation expansion, is similar to the previous task but expands and replaces the abbreviations or contractions in English with their full form to better match the occurrence of the same entity from the rest of the text. An example of an extended abbreviation is from “NLP” to “Natural Language Processing”, and an example of an extended contraction is from “I’m” to “I am”. The fourth task, special text handling, will handle the special text that was not removed from the text as noisy in the first step of text pre-processing. In the situation of special text that cannot be understood by the downstream model but present needs to be emphasized and connected to similar present at other text positions, the special text handling strategies are used. The most commonly handled special text type is numbers, which are handled by converting digits to words like “2023” to “two thousand and twenty-three” or replacing the number with a placeholder like “NUM” in the task when actual numerical values are insignificant. URLs are also hard to be fully decoded by the machine learning model, so replacing it with a placeholder will be sufficient.

Morphology removal is the process of removing English grammatical variations known as morphology in the field of linguistics. There are two types of morphologies to be discussed in the scope of text pre-processing: inflectional morphology and derivational morphology. The process of lemmatisation and stemming is used to undo the corresponding morphology and reduce words to their base form to facilitate text normalisation. Manning and Schutze [30] have delineated the difference between the two processes, with pros and cons emphasised for each in their book, along with a detailed explanation of the processes.

2.1.5.1 Lemmatisation

Inflectional morphology is the process of modifying words with inflectional affixes to convey additional grammatical information, such as tense and plural, without creating a new word, altering the core meaning or changing the part of speech of existing words. An example of verb inflection for tense is from "research" to "researching", and an example of noun inflection for the plural is from "language" to "languages". To normalise the inflectional morphology, the technique of lemmatisation is used to remove the inflection from the word to transfer the word back to its base form, which is also known as the lemma, based on dictionaries of words. Because lemmatisation is a dictionary-based approach, it has the advantage of high transform accuracy and is even capable of handling irregular inflectional morphology such as transfer "wrote" back to "write". Besides, because the normalised lemma is the based form of the existing word that can be found in the dictionary, the text output of lemmatisation will have exceptional interpretability and human readability. Also, the only precondition condition to apply lemmatisation is to have the corresponding dictionary, which implies the technique can be generalised to any language domain as long as there is a corresponding dictionary available. Furthermore, Lemmatisation is also content-sensitive and uses context from the text to disambiguate for deciding between different lemmas of the same word in different sentences. However, the dictionary lookup actions involved in the lemmatisation process are often extremely computationally intensive. Therefore, the lemmatisation is usually too slow to be used for real-time large-scale NLP systems.

2.1.5.2 Stemming

The derivational morphology creates new words with different meanings or part-of-speech tags by adding affixes to derivate words. The frequently used affixes for derivational morphology include suffixes and prefixes. Suffixes are added at the beginning of the word and are used to change the part of the speech tag without altering the

core meaning, such as from the verb “research” to the noun “researcher”. Because the suffixes will not alter the core meaning, therefore the suffixes will be removed in most of the normalisation pipelines. Prefixes, on the other hand, are added at the end of the word and are used to the meaning of the word dramatically without altering the part of the speech tag of the word, which is exactly the opposite of the suffixes and the example of the prefixes is from adjective “significant” to adjective with exact opposite meaning “insignificant”. Because the prefixes tend to flip the word’s meaning, they will not be removed to preserve the text’s semantics. The methodology of stemming is used to remove the suffixes or even prefixes to obtain the root, also referred to as the stem of the word, by applying algorithmic rules. The major advantage of steaming is the speed so that it can be used for efficiency-sensitive task such as information retrieval because applying rule-based operation are generally significantly faster than dictionary look-up. However, stemming is also facing various challenges. The stemming rule is context insensitive, so the word in a different context will result in the same stem, resulting in reduced accuracy. The stemming rule is also a delicately designed language-specific rule that is incapable of being generalised to other language domains. Lastly, the output of stemming, the stem, is often not a real word, such as “happiness” stemmed to “happi” instead of “happy”.

2.1.6 Stop Word Removal

Stop word removal is a common text pre-processing step to remove the stop words that are common words in a language with frequent occurrence but contribute barely any meaning. An example of an implementation capable of removing such redundant words to enhance data quality is the package named “tidytext”, proposed by Silge and Robinson [31]. The high appearance frequency and low information carriage of stop words make it similar to noisy text. Upon removal, similar to noisy reduction, the model will be able to focus more on the more informative words to improve the model’s accuracy. Since the removal will be all instances of stop words, the word would be removed from the vocabulary, which can bring the benefit of dimensionality reduction to improve the efficiency of the model. The implementation of stop word removal requires the text to be normalised first and then iterate each token with a stop word list usually obtained from packages such as NLTK, spacy and Sklearn. The token is to be removed from the text if a match is found from the list. Otherwise, the token is retained. This procedure would also need caution to select not only domain language-specific but also task-specific carefully. The exclusion of stop words that contain negation words, such as “not” in sentimental analysis can result in missed negation relations. This can lead to incorrect outputs, sometimes the opposite of the correct answer. For instance, the word “not”

is included in the stop word lists of major NLP packages, NLTK, Spacy, and Sklearn. Therefore, removing this stop word can lead to wrong analysis results, as demonstrated in Figure 2.1 by Zhang et al. [3].

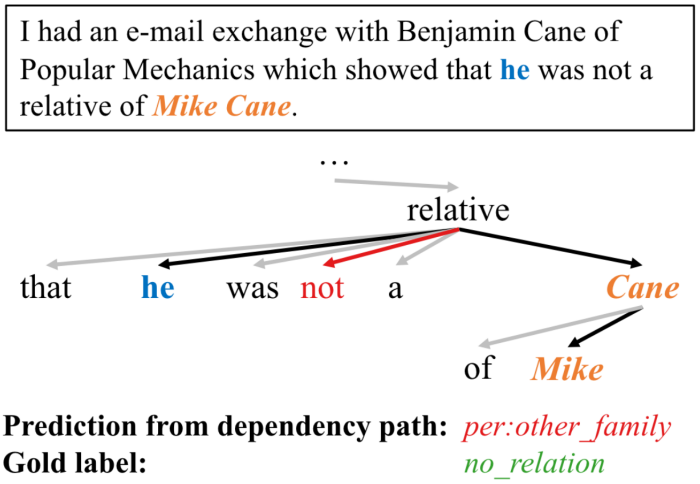

FIGURE 2.1: The example of relation extraction task on modified TAC KBP challenge corpus with miss out negation resulting opposite result cite from the paper by Zhang et al. [3]

To show the effect of text pre-processing, Kannan and Gurusamy [32] have done research on investigating the effect of text pre-processing in the information retrieval task in text mining problems. The research suggests by doing stop-word removal, the word count of the testing document can be reduced by up to 30 % to not only reduce the memory usage of the data to reduce the load on computational resources but also improve the model accuracy as removed stop words are mostly noisy with a little real meaning. Kannan and Gurusamy have improved the performance in an additional step by further applying the Stemming techniques, merging the same words represented in different forms to save up another 50% on top and merging the same words will make the system less confusing.

2.2 Text Embedding

Text embedding has been the most critical aspect of almost any NLP task over the last decade, as the spectacular advancement of text embedding strategies has dramatically changed the entire NLP research field and industry. In English, text embedding is constantly derived from the word embedding. The fundamental idea of word embedding is to represent the words as continuous vectors, capturing the syntactic and semantic intricacies of the words in vector space. The word embedding is one subclass of text representation methods that takes advantage of distributional semantics and contextual

representation. The purpose of text representation, in general, is to convert the English natural language composed from English letters into a form that the computer can understand: the numerical vector data form can be interpreted as binary to be input into the subsequent machine learning model as features. Overview of the entire history of text representation The Bag of Word method was used before the era of word embedding, and the development of word embedding can be further broken down into two stages: distributional semantics with count-based and predictive approaches and contextual semantics.

As discussed in the previous section, the taxonomy and inclusion of mainstream word embedding strategies have no universal, so this section will mainly focus on reorganising the knowledge from two surveys on word embedding. The survey by Almeida and Xexéo [13] has listed, overviewed, and referenced mainstream count-based morels and prediction-based word embedding strategies with elaborate tables. Another survey by Incitti et al. [4] has also reviewed word embedding strategies with more focus on the more modern transformer-based models to supplement what was missing in the other survey, along with meticulously made graphs on the overview and taxonomy of text representation methods classified in different approaches cited in Figure 2.2 that unfortunately this research will not expand according to the taxonomy proposed Incitti et al. [4] but will utilising the knowledge from the survey for this literature review.

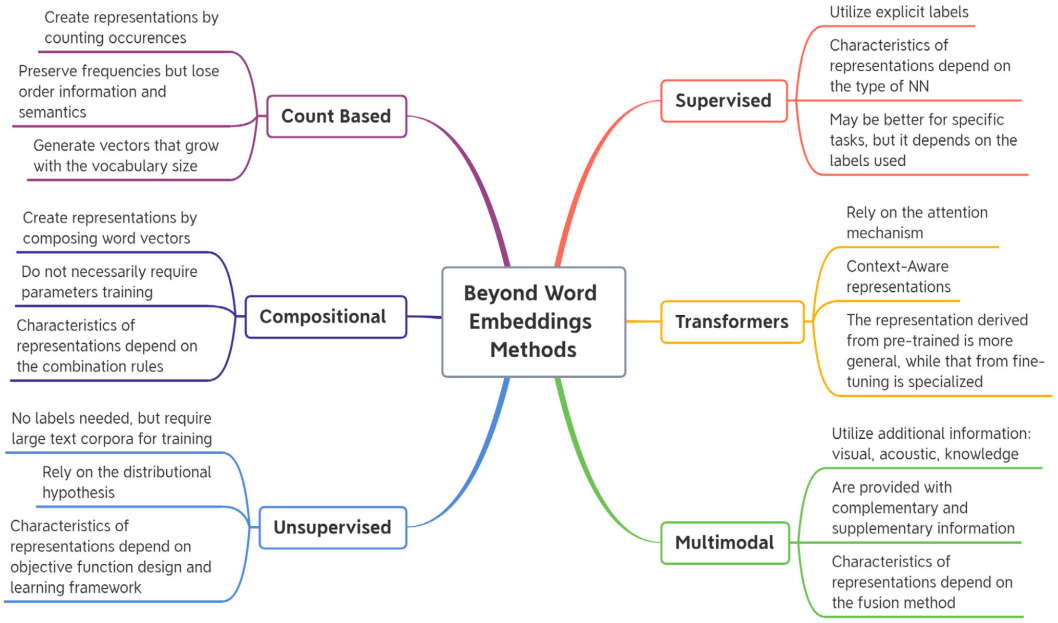

FIGURE 2.2: The overview and taxonomy of text representation methods classified in different approaches cited from the survey by Incitti et al. [4]

2.2.1 Bag of Words Method

The earliest approach to text representation, the Bag of Word method, will not be experimented with due to its unsatisfactory performance but will be discussed in contrast with word embedding to show the technique advantage of the latter approaches. The Bag of Words model was first proposed in the journal article "Distributional Structure" by Harris [33] in the year of 1954. The Bag of Words model treats the text as an unordered set of words and adds up one-hot-encoded vector of words as the text vector. Each dimension of such vector corresponds to each unique in the vocabulary, and each entry represents the count of the corresponding word in the text. Despite the simplicity of the approach, it suffers from many significant drawbacks, including loss of order information, high dimensionality, lack of semantics, limited context capture, and incapable of handling Out-of-Vocabulary. The order information is lost as Bag of Words ignored the order of words in the text, resulting in two sentences with the exact words but opposite meanings such as the sentences "I got word from home" and "I got home from work" can end up with exact vector representation. As stated before, the sentence vector has the same dimension as the vocabulary size, which can inflate dramatically with high sparsity in the situation of large corpora, resulting in inefficient computation. The most noticeable defect of the approach is its inability to capture the semantic relationship between words. As humans, we have the ability to understand the meaning of words. For example, when we think of the word "pizza", we can construct an image of its pie shape in our minds and think of the smell of it fresh from the oven. However, for non-semantic approaches like the Bag of Word method, words are merely discrete symbols without any inherent meaning or association, resulting in synonyms being treated as completely distinct words with no inherent relationship. Also, the Bag of Words model treats each word independently without capturing the context around a word. Hence, the model cannot disambiguate homonyms based on their context. Lastly, the Bag of Words model output vector with a fixed dimension corresponding to vocabulary in the training corpus. Therefore, the new words from the test corpus that are not encountered in the training corpus can not be represented by the Bag of Words model. Although there have been upgraded Bag-of-Words versions with N-gram, it can preserve very limited order and context information with trade-offs of even higher dimensions.

In conclusion, the subsequent distributional semantics and contextual representation approaches that will be thoroughly investigated and experimented with in later subsections have addressed the shortcomings mentioned for the Bag of Words approach. To solve the semantic understanding challenge, the distributional semantics methods capture the word's semantic embedding by the context of the word. This is taking advantage of the nature of words appearing in semblable contexts, which usually share similar semantic

meanings. Therefore, the synonyms will be placed at a close distance in the vector space. The dimensionality is also reduced by enforcing a fixed dimension size independent of the vocabulary size. This is achievable by having a continuous vector rather than a discrete one like before, resulting in a much improved computational efficiency with richer semantic information carried in vectors. The contextual understanding problem is solved by the contextual semantic approach by dynamically generating word embedding based on the local context, allowing polysemous words to be disambiguated.

2.2.2 Count-Based Methods

The count-based method was based on the fundamental concept from traditional computational linguistics that words appearing in similar contexts tend to have similar meanings. The concept is also referred to as the distributional properties of word that enables the count-based method to derive word vectors from the statistics of word co-occurrence when there is sufficient training corpus. Although The count-based methods have been overshadowed by predictive approaches powered by more sophisticated deep neural network models, the simplistic count-based method, as the first attempt to represent the text in semantic space, offered remarkable interpretability and computational efficiency, making it worth the investigation. Almeida and Xexéo have overviewed the mainstream count-based models for embedding with reference, proposed data, and a short note in a table that was cited and adapted in Table 2.1. Unfortunately, due to the limitation of space in this thesis, only the most iconic count-based text embedding methods relevant to this experiment on information dispersal tracking will be further investigated, which are documents as a context approach and words as a context approach.

2.2.2.1 Documents as a Context

Term Frequency-Inverse Document Frequency (TF-IDF) was first introduced by Salton and Buckley [43] in 1988. The idea was to capture the numerical statistic that evaluates the importance of a word in a document relative to the corpus of documents.

The formula for TF-IDF is typically represented as the product of two individual metrics: Term Frequency (TF) and Inverse Document Frequency (IDF). The formula is given by:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{2.1}$$

In the formula, $\text{TF}(t, d)$ is the Term Frequency of the term t in document d, calculated by counting the number of times term t appears in document d divided by the total

Article	Overview of Strategy	Notes
Deerwester et al. [34] in 1990	LSA is introduced. Singular value decomposition (SVD) is applied on a term-document matrix.	Used mostly for IR, but can be used to build word embeddings.
Lund and Burgess [35] in 1996	The HAL method is introduced. Scan the whole corpus one word at a time, with a context window around the word to collect weighted word-word co-occurrence counts, building a word-word co-occurrence matrix.	Reported an optimal context size of 8.
Rohde et al. [36] in 2006	Authors introduce the COALS method, which is an improved version of HAL, using normalization procedures to stop very common terms from overly affecting co-occurrence counts.	Optimal variant used SVD factorization. Reports gains over HAL [35], LSA [34] and other methods.
Dhillon et al. [37] in 2011	LR-MVL is introduced. Uses CCA (Canonical Correlation Analysis) between left and right contexts to induce word embeddings.	Reports gains over C&W embeddings by Collobert and Weston [38] in 2008, HLBL by Mnih and Hinton [39] in 2008 and other methods, over many NLP tasks.
Lebret and Collobert [40] in 2013 (last revised in 2017)	Applied a modified version of Principal Component Analysis (called Hellinger PCA) to the word-context matrix.	Embeddings can be *tuned* before being used in actual NLP tasks. Also reports gains over C&W embeddings, HLBL and other methods, over many NLP tasks.
Pennington et al. [41] in 2014	Introduced GloVe, a log-linear model trained to encode semantic relationships between words as vector offsets in the learned vector space, using the insight that co-occurrence ratios, rather than raw counts, are the actual conveyors of word meaning.	Reports gains over all previous count-based models and also SGNS [42], in multiple NLP tasks.

TABLE 2.1: The overview of mainstream count-based text embedding models. Table cited and adopted from the survey by Almeida and Xexéo [13]

number of terms in d. $\mathrm{IDF}(t, D)$ is the inverse document frequency of term t in the corpus D. It's typically calculated as:

$$\mathrm{IDF}(t, D) = \log\left(\frac{N}{1 + \mathrm{DF}(t, D)}\right) \tag{2.2}$$

Where N is the total number of documents in the corpus. And $\mathrm{DF}(t, D)$ is the number of documents in which term t appears. Therefore, the TF-IDF weight increases proportionally to the number of times a word appears in the document but decreases when the word also appears commonly in other documents within the same corpus. This regularisation factor aimed to give words that are unique to a particular document higher weight by preventing the model from covering up more influential words with the words

that commonly appear in the English language by nature, such as stop words discussed in the section 2.1.6.

Although TF-IDF provides a way to prioritise the distinguishing words in documents, it still suffers the challenge of computational inefficiency due to the sparsity of the TF-IDF matrix with dimension of vocabulary size times document count. Hopefully, not long after, Deerwester et al. [34] in 1990 presented the Latent Semantic Analysis (LSA) method as a tool for dimensionality reduction to improve computational efficiency and make embedding large text corpora possible. The idea of LSA is to uncover latent semantic relationships between words and their context document by applying singular value decomposition (SVD) to TF-IDF term-document matrices. Given a term-document matrix matrix A of size vocabulary size times size of documents represent by $|V| \times |D|$, SVD represents A as:

$$A = U\Sigma V^* \tag{2.3}$$

In the equation, U is the new $|V| \times m$ term matrix from the left singular vectors of A.Σ is an $m \times m$ diagonal matrix, with its diagonal entries being the singular values of A typically arranged in descending order. In practice, the hyperparameter m is usually set to be $Rank(A)$. V^* (more precisely V^T) is the transpose new $m \times |D|$ document matrix from the right singular vectors of A. Followed up by the SVD process, further truncating the new term matrix to the desirable dimension can reduce dimensionality while preserving the most semantic information of the distance between embeddings, which signifies semantic similarity by transforming the matrices from discrete counts of concurrence to continuous space.

Lastly, the TF-IDF was originally designed to capture the word's semantic embedding in the context of the document. It can also be used to capture the document semantic embedding in the context of words contained in each document by simply obtaining the new document matrices from SVD. This strategy is commonly used for topic modelling tasks in the research field of NLP.

2.2.2.2 Words as a Context

The Co-occurrence Matrix is a simple word as a context idea to build a large matrix capturing the frequency of each word co-occurs with every other word in vocabulary within a specific window of words in the corpus. The matrix can capture both direct and indirect semantic relationships between words. The major advantage of this words as a context approach is the capability of capturing the word semantic without reliance

on the document information compared to the document as a context approach just discussed. Similar to using TF without IDF, the raw co-occurrence count suffers from the problem of common word dominance. The solution to such a challenge is Pointwise Mutual Information (PMI), rooted in information theory, which is a statistical measure used to quantify the strength and significance of association between discrete variables word co-occurrence in the context of NLP. The PMI has the foumlar of:

$$\text{PMI}(x, y) = \log_2 \left(\frac{p(x, y)}{p(x) \times p(y)} \right) \tag{2.4}$$

Where $p(x, y)$ is the joint probability of word x and word y occurring together and $p(x)$ and $p(y)$ are the marginal probabilities of word x and word y in the corpora, respectively. The core idea behind PMI is to evaluate the probability of two words appearing together compared to what would be expected if they were independent.

The PMI may yield negative values for less frequent word co-occurrences that can cause major interpretability issues in the situation of word embedding. Therefore, the variation of PMI, Positive Pointwise Mutual Information (PPMI), is introduced to address the problem by setting all negative values to zero to ensure a more intuitive representation of the association between words. The PPMI is computed by:

$$\text{PPMI}(x, y) = \max\left(0, \text{PMI}(x, y)\right) \tag{2.5}$$

This ensures that all negative values of PMI are clipped to zero. The PPMI method has been around in the information theory field for a while, but the one influential paper highlighting the utility of PPMI for semantic word embedding was published by Liu Bing et al. [44] in 2002.

In comparison to TF-IDF, in most situations, the vocabulary size would be much larger than the document size, so the PPMI matrix having a dimension of vocabulary size square tends to be uncritical to use as the semantic embedding before any subsequent factorise or dimensionality reduction producing dense word vectors. Hopefully, it can also be applied with the SVD technique demonstrated in Equation 2.3 for dimensionality reduction. The research by Levy and Goldberg [45] in 2014 has pointed out that PMI shares the same characteristics of outstanding computational efficiency compared to predictive methods.

2.2.3 Predictive Methods

The predictive methods that benefit from the high performance of deep-learning techniques have been a giant step forward from the initial attempt of capturing the word embedding using the word-based method. Differing from count-based methods, the predictive methods aim to use parameterised machine learning models, typically neural network work models, for a context-related task, for example, predicting the word based on the surrounding context or the other way around. In the process of training models to perform corresponding prediction, dense and semantic meaningful vector embeddings for words are learned as a by-product. Similar to the previous section on count-based methods, the survey by Almeida and Xexéo [13] has included a table reviewing the milestone research on the topic of prediction-based models for embeddings with emphasis on the iterative performance advance throughout the development which is cited and adopted in Table 2.2.

2.2.3.1 Word2Vec

Word2Vec has been a milestone that reshaped the entire research field of NLP by transitioning the text embedding from sparse, high-dimensional representations produced by bag-of-words and TF-IDF strategies discussed in previous sections to dense, continuous vector spaces. Its ability to encapsulate semantic and syntactic word relationships in fixed-size vectors transformed how researchers approached language modelling, laying the groundwork for NLP's current deep learning era. The state-of-the-art information propagation tracking model proposed by Joshi and Sinnott [8] that formed the basis of this research was utilising the Word2Vec-based model for text embedding. Therefore, it is essential to explore the Word2Vec technique in detail before further discussing the model in later sections. The Word2Vec was first proposed by Mikolov et al. [5] in their paper "Efficient Estimation of Word Representations in Vector Space" in the year 2013, with the core ideas still very much relevant in even the most advanced NLP models today. Word2Vec is based on a distributional hypothesis, assuming that words in similar contexts share semantics. By embedding words into a high-dimensional space, Word2Vec captures these contextual relationships in the form of dense vectors, making words with similar meanings closer in this vector space. The Word2Vec employ a shallow neural network to iteratively adjust the network's weights, which essentially are the word embedding vectors, during the training stage to minimise the prediction error. However, different in prediction task setup and model architecture, there are two different implementations of Word2Vec: Skip-gram and Continuous Bag of Words (CBOW) with the model architecture shown in Figure 2.3. The Skip-gram architecture

is set to perform the task of predicting the surrounding context within a certain predefined window from a given word. This setting endorsed Skip-gram being particularly effective with a smaller text dataset and capable of representing rarer words or phrases. The CBOW, on the other hand, does the opposite by predicting the target word from its surrounding context. CBOW also has the opposite characteristic of being generally faster and showing slightly better accuracy for frequent words than Skip-gram.

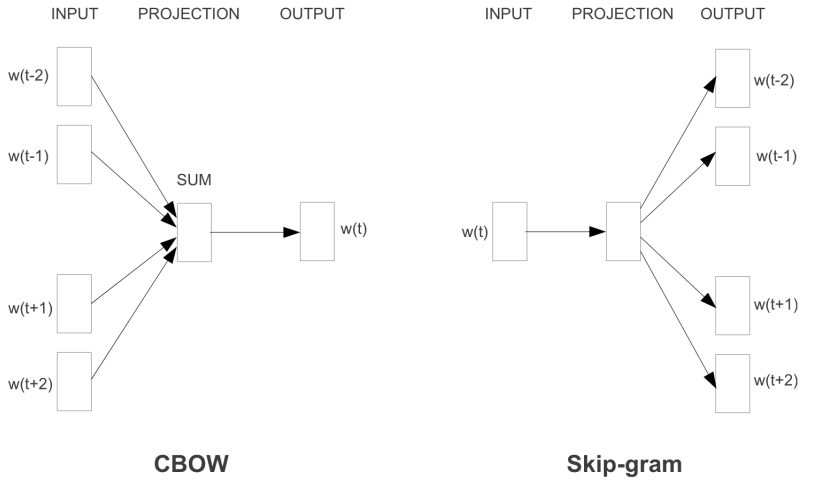

FIGURE 2.3: Architecture of the CBOW and Skip-gram respectively cited from the paper by Mikolov et al. [5] that originally proposed Word2Vec technique.

Empowered by the powerful neural network machine learning model, the initial version of Word2Vec has already demonstrated its powerful performance but posted serious challenges in computational efficiency in contrast to count-based models. It also learns word vectors more efficiently than most other previous approaches of prediction-based models using neural network architectures for not involving dense matrix multiplications. However, in the same year of the proposition of Word2Vec, 2013, Mikolov et al. [42] published a subsequent paper focusing on further optimising Word2Vec to improve efficiency and performance in several approaches. The paper focused on the Skip-gram architecture among the Word2Vec because it performs better on capture word embedding, especially when training on a large corpus, but suffered the most performance drop when the size of the corpus increases. As stated in the paper, the Skip-gram model is aimed at predicting the surrounding words of a given word by maximise the average log probability to find the word embedding, which can be mathematically represented as:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \tag{2.6}$$

Where the c is the predefined context window indicating how far from the target word the words within can be considered as context words. A bigger c could lead to better training performance as a wider context is captured when building the word vector but with more training time as a trade-off. The log probabilitylog $p(w_{t+j}|w_t)$ can be further expanded into:

$$p(w_o|w_t) = \frac{\exp({v'_{w_o}}^\top \cdot v_{w_t})}{\sum_{w=1}^{W} \exp({v'_w}^\top \cdot v_{w_t})} \tag{2.7}$$

In the equation, v_w and v'_w are the initial input word vector representation and final output semantic word embedding of the word w, respectively. The equation suggests the computation cost for computing log probability is proportional to the vocabulary size W, which tends to be extremely large in real practice. Therefore, optimising the performance of the Skip-gram model is necessary and urgent.

The first approach discussed in the paper [42] is to use the Hierarchical Softmax first proposed by Morin and Bengio [48] reducing computational complexity from $O(W)$ to $O(log_2(W))$ for each training step. The Hierarchical Softmax technique represents the vocabulary with a binary tree data structure, with the leaf node saving all the words and other nodes saving the relative probabilities of its child nodes. When computing the probability $p(w_o|w_t)$, the Hierarchical Softmax algorithm only computes the relative probability of the word among all the words lying along the path from the root to the target leaf nodes in the binary tree instead of among all the words in the vocabulary. The idea was converted into a formula by Mikolov et al. [42] as:

$$p(w_O|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left(\llbracket n(w, j+1) = ch(n(w, j)) \rrbracket \cdot {v'_{n(w,j)}}^\top v_{w_I}\right) \tag{2.8}$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{2.9}$$

Where the $n(w, j)$ stands for the j-th node on the path with length $L(w)$ from the root node to the leaf node containing the word w. The $ch(n)$ represent a child of n and $\llbracket x \rrbracket$ be 1 if x is true and -1 otherwise. The formula suggests the computational cost for calculating the $p(w_o|w_t)$ is proportional to $L(w_O)$, which on average is less or equal to $log(W)$ due to the natural of the binary tree. By reducing the complexity of the prediction task, Hierarchical Softmax allows Word2Vec to be trained more efficiently on large vocabularies, making it feasible to produce high-quality word embeddings even on massive datasets.

Noise Contrastive Estimation (NCE), first introduced by Gutmann and Hyvarinen [51] and first applied to language modelling by Mnih and Teh [52], is an alternative way to reduce the computational cost alternative from Hierarchical Softmax. The traditional

approach of the Word2Vec method to predict the correct word out of the entire vocabulary requires computing the probability of the target word across the entire vocabulary for the softmax activation function to adjust the embeddings to assign high probabilities to the actual context words and lower probabilities to all other words in the vocabulary. The NCE originates from the idea of estimating parameters of unnormalised statistical models by introducing artificial noise data to distinguish between true data and noise. The strategies can be used to simplify such a task into a more tractable binary classification problem of given a word-context pair and predict if it's a real context from the corpus or made-up noise for a significant reduction in the number of weight updates and computations for each training step. The noise samples were drawn from the predefined noise distribution describing the non-context words in the context in which the complexity of the noise distribution affects the efficiency and performance of NCE directly. The Negative sampling (NEG) strategies proposed by Mikolov et al. [42] is a specific and streamlined version of NCE tailored for the Word2Vec model for further simplifying and improving the efficiency of Word2Vec strategies. NEG does not attempt to model the noise distribution explicitly like the original NCE but further simplifies it by focusing only on the sampled negatives to distinguish between the observed samples (positive) and a few randomly sampled negatives. NEG uses a simpler distribution, usually a unigram distribution, to randomly draw negative samples. The less negative sample also avoids overwhelming the positive sample to preserve the semantic and syntactic information in the actual context. Therefore, NEG is conceptually simpler and more computationally efficient than the full NCE approach.

The Word2Vec approach with the discussed optimisation has been the optimal strategy to capture semantic word embedding at the time with excellent performance and effects, but it still has one last challenge to overcome to be utilised in the news dispersal tracking system, and the challenge is how to represent larger text such as sentence or paragraph. To encounter such a challenge, Le and Mikolov [53] in 2014, the subsequent year after Mikolov's team proposed Word2Vec, introduced a methodology termed Doc2Vec or alternatively named Paragraph Vector in some context building upon the success of Word2Vec in representing words as vectors but extend the idea to entire documents or paragraphs. While Word2Vec operates on the premise of representing individual words in a dense vector space, Doc2Vec aims to represent larger chunks of text, such as sentences, paragraphs, or entire documents, as vectors. The model does this by associating a unique vector with each document and training this vector in tandem with word vectors, leading to a consolidated representation that captures the semantics of the entire text.

Word2Vec has two different approaches, CBOW and Skip-gram. Likewise, Doc2Vec also have two different algorithms: Distributed Memory (DM) and Distributed Bag of

Words (DBOW). In the DM model, paragraph vectors are captured as the by-product of a neural network trained on the task of inferring a centre word based on context words along with a paragraph identifier. The paragraph vectors are averaged with context word vectors, resulting in a vector to predict the centre word in the training stage. In comparison to Word2Vec, DM is similar to the CBOW model but with an added paragraph token. Because the DM model has considered the context in the training phase, it represents the order of words better and is typically more suited for tasks that emphasise the sequence and semantics of the word, like sentiment analysis. On the other hand, DBOW is based on the Skip-gram model in Word2Vec to train the neural network to predict words in a paragraph based on the input paragraph vector among randomly sampled words, with the main difference of preserving only paragraph vectors but not the word vectors. Although DBOW trains faster than DM for not focusing on word order, it still outperforms DM on certain tasks when the overall topic of the paragraph is more critical than the specific word order or semantics. It is worth noting that hybrid models exist in the practice by concatenating the result paragraph vector from DM and DBOW to take advantage of both models. Also, there have been other document representation methods, namely the naive approach of averaging Word2Vec vectors of all words of the paragraph as the paragraph vector. But Doc2Vec has the advantage over other methods in the ability to capture semantic meanings of longer texts more effectively. The unique vector assigned to each document or paragraph can capture latent topics and abstract concepts present in the text, making Doc2Vec particularly valuable for tasks like document similarity, clustering, and information retrieval.

However, like Word2Vec, the quality of Doc2Vec embeddings relies heavily on the amount and quality of training data and the parameter tuning. Recent advancements in NLP have introduced transformer-based models like BERT, classified as the contextual embeddings to be introduced in the next subsection, which often outperform Doc2Vec in various tasks. Nevertheless, Doc2Vec remains a pivotal step in the evolution of NLP and still finds utility in applications where large pre-trained models are not feasible.

2.2.4 Contextual Embeddings

In more recent years, contextual embedding methods have surged in popularity within the NLP research field, marking a significant shift from traditional word embedding techniques introduced in previous sections. Unlike previously discussed text representation methods, such as the prediction-based Word2Vec approach, which are all classified as static embedding methods that assign a fixed vector to each word type, the contextual embedding methods assign vectors based on the word's context to offer a dynamic representation. This dynamic nature promises advantages over traditional models like

Word2Vec. The introduction of contextual embeddings is relatively recent, most notably through models to be explained in detail in the following sections such as ELMo introduced by Peters et al. [54] in 2018, BERT proposed by Devlin et al. [55] in 2019, and GPT presented by Radford et al. [56] in 2018, which represents a paradigm shift in the world of text representations. These models leverage vast amounts of data and deep learning architectures, notably Long Short-Term Memory (LSTM) and transformer architectures, to produce word vectors.

The ability to dynamically generate distinct vectors for a word based on its context marks the primary advantage of contextual embedding. In comparison, static embedding generated by traditional approaches like Word2Vec struggles with polysemy, which are words with multiple meanings, as they can only bind a word to a single representation. For example, the word "bank" would have the same representation in "river bank" and "bank account" with static embedding generated by Word2Vec. However, with contextual embeddings, "bank" would have different representations in these contexts, capturing its semantic nuances. Dynamic embedding can also better understand more intricate linguistic phenomena such as negation, sarcasm, and even coreference resolution, as inferred in the paper by Asudani et al. [57].

Contextual models, particularly BERT, also introduce a two-step process: pre-training on a large corpus and fine-tuning on specific tasks. This procedure allows them to transfer knowledge from vast datasets to specific tasks, even with limited data. Or used completely as an unsupervised model when no annotated data is available. This characteristic makes them more versatile than models like Word2Vec, which lack this fine-tuning capability. Despite their strengths, contextual embeddings are computationally expensive, both in terms of training and inference. Moreover, they require significant memory storage, making them less accessible for edge devices or environments with limited resources. Therefore, the choice of word embedding strategy should be based on the nature of the specific task with the compromise between performance and efficiency.

2.2.4.1 ELMo

Embeddings from Language Models (ELMo) proposed by Peters et al. [54] in 2018 is the first contextual embedding model representing the breakthrough transition of representation of the word in the natural language process from traditional static word embeddings to context-dependent dynamic semantics embedding. In the paper, the author Peters et al. demonstrated that the ELMo often significantly outperformed traditional statistic embedding modes in a range of NLP tasks, especially in tasks requiring a deeper semantic understanding, such as sentiment analysis or named entity recognition, when

incorporated into existing NLP models. It was shown that when incorporating ELMo into other NLP architectures, even the simple concatenation of ELMo embeddings with traditional embeddings can drastically enhance the model performance across a range of tasks which its modularity and ease of integration drove the widespread adoption of ELMo in the NLP research field.

To understand the brilliant architectural design that empowered ELMo, it is essential to unfold and explore the implementation of the ELMo model in detail. ELMo combines two newly proposed word-changing neural network structures, Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTMs), to derive its deep contextualized word representations. The CNN model is involved because ELMo generates word representations at its foundation by first creating embeddings for words based on their individual characters, which this architecture was cited from the survey by Chiu and Baker [6] to illustrated in Figure 2.4. This differs from many traditional embedding techniques, such as BOW, that treat words as indivisible units. ELMo begins by converting each word into a series of character embeddings in the actual implementation. For the example shown in Figure 2.4 with the word "The", ELMo would first represent it by embeddings for "T", "h", and "e". These character embeddings are then passed through a convolutional layer. The CNN helps capture local information about character sequences, such as the patterns and structures that might hint at morphology or meaning, which have been discussed in detail in section 2.1.5. For example, the suffix "-ing" or prefix "un-" have consistent semantic and syntactic properties across various words. After the convolutional layer, a max-pooling operation is applied to capture the most salient features from the convolutional filters for each word. The outcome of this process is a fixed-size vector for each word, which encapsulates information about its characters and their arrangements.

With the character-based token representations in hand, ELMo further refines these embeddings by considering the broader context in which each word appears. Therefore, each token representation is fed into a bidirectional LSTM. The "forward" LSTM reads the text from left to right, and the "backward" LSTM reads it from right to left. This bidirectional approach ensures that the representation of each word is influenced by all other words in its sentence, regardless of their position. Additionally, ELMo uses multiple layers of bidirectional LSTMs to capture information at varying levels of abstraction illustrated in Figure 2.5. For instance, lower layers might capture syntactic information, while higher layers capture more semantic content. Finally, when generating the final embedding for a word, ELMo combines the outputs from all layers of the LSTM using a weighted average, ensuring that the resulting representation is rich and captures a wide range of linguistic information. An illustration of the

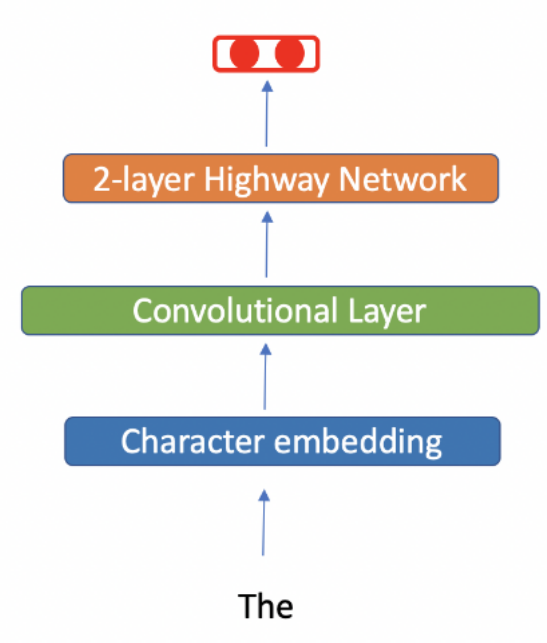

FIGURE 2.4: An illustration of the word 'The' being embedded using CNN at the character level. The figure was cited from the survey by Chiu and Baker [6]

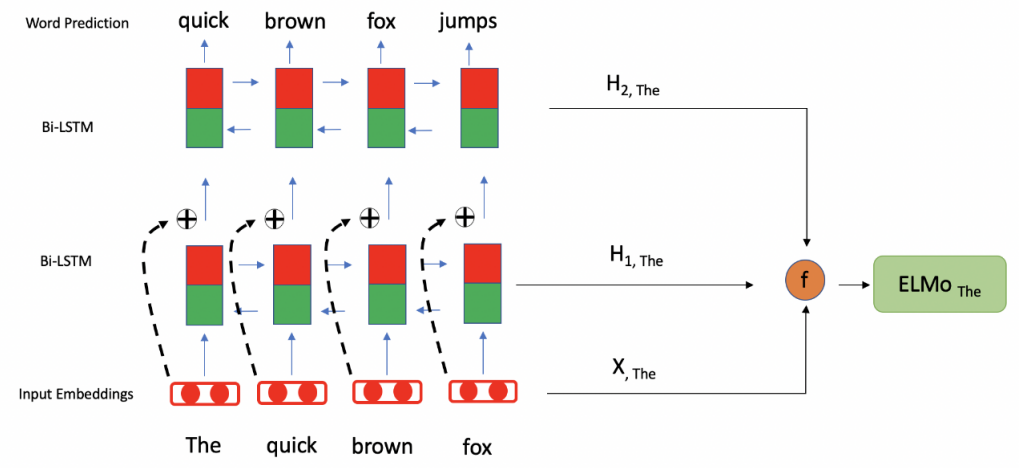

FIGURE 2.5: An illustration of ELMo The figure was cited from the survey by Chiu and Baker [6]

In conclusion, ELMo marked a significant advancement in the field of NLP by highlighting the importance of contextualised word representations. Its success across various benchmarks demonstrated the value of considering context in word embeddings. As the NLP field continues to evolve, ELMo remains a foundational piece of work, representing a bridge between static embeddings and the more advanced transformer-based models to be introduced in the coming section that dominates the landscape today.

2.2.4.2 BERT

The Bidirectional Encoder Representations from Transformers (BERT) is a state-of-the-art NLP (NLP) model introduced by researchers Devlin et al. [55] at Google AI in 2019. Its architecture is based on the Transformer, a deep-learning model proposed by Vaswani et al. [58] in 2017. BERT revolutionized the NLP landscape due to its ability to pre-train vast amounts of text and fine-tune specific tasks, achieving record-breaking results on numerous benchmarks, as suggested in the review by Koroteev et al. [59].

BERT's training comprises two main phases that make it known for large-scale and pre-trained. The first phase involves unsupervised pre-training on two enormous corpora: BooksCorpus(800M words) [60] and English Wikipedia (2,500M words), where the model learns to predict masked words in a sentence, a strategy termed the "masked language model". The second phase involves fine-tuning the pre-trained model on specific tasks such as question answering, named entity recognition, and sentiment analysis using a smaller labelled dataset. This adaptability has led to its widespread adoption in the NLP community. BERT set new standards on eleven NLP tasks upon its introduction, sparking a wave of research into Transformer-based models. Its influence extends beyond academic research; many industry applications now leverage BERT and its variants for tasks ranging from search engine optimization to chatbot development.

BERT primarily employs the encoder mechanism of the Transformer architecture. While the original Transformer model proposed by Vaswani et al. [58] in 2017 had both encoders and decoders, BERT uses just the encoder stacks. The heart of the Transformer, and by extension BERT, is the self-attention mechanism. This mechanism allows each token in the input sequence to focus on other tokens, giving the model the capability to understand context over varying distances within the input data. This is particularly important for languages where meaning can be determined by words far apart in a sentence. Since the Transformer architecture doesn't have a sense of the order of tokens inherently, positional encodings are added to the embeddings at the bottom of the encoder stack to provide a sense of token position within a sequence. BERT uses multiple layers of these encoders where the exact number depends on the specific BERT variant of BERT-base or BERT-large. The stacking of layers enables the capture of complex relationships in the data. Unlike some models that read input data sequentially, which is either left-to-right or right-to-left, BERT processes input data in both directions simultaneously. This bi-directionality is inherent in the Transformer's design and gives BERT its name. It allows BERT to capture context from both before and after each token. Each encoder layer in BERT contains a fully connected feed-forward neural network and a normalization step in addition to the self-attention mechanism.

Following the success of the original BERT model, several variants were introduced. The RoBERTa was introduced by Liu et al. [61] in 2019. It modifies key hyperparameters in BERT, removes the next-sentence pre-training objective, and trains with much more data. These optimisations led to performance gains across various NLP benchmarks. The DistilBERT was introduced by Sanh et al. [62] in 2020 as a distilled version of BERT, aiming to retain most of BERT's capabilities while being smaller and faster by training DistilBERT to mimic BERT's behaviour it offers a more efficient alternative for resource-constrained environments. The BERTweet proposed by Nguyen et al. [63] in 2020 is a variant of the BERT specifically fine-tuned for the analysis of English tweets due to the unique linguistic characteristics of tweets that include the use of slang, abbreviations, hashtags, and emojis. It was based on RoBERTa's pre-training procedure.

However, the above models are all for word embedding rather than sentence or paragraph embedding. Therefore, SentenceBERT was introduced by Reimers and Gurevych [64] in 2019 and has successfully bridged the gap between the power of transformer models and the computational efficiency required for large-scale applications. By producing semantically rich, fixed-size sentence embeddings, SBERT has not only accelerated tasks like similarity measurement and clustering but has also set the foundation for future research on sentence-level embeddings in the era of deep learning. The continued interest in SBERT underscores its importance in the evolving landscape of NLP.

2.2.4.3 GPT

Generative Pre-trained Transformer (GPT) is a language model first introduced by Radford et al. [56] from OpenAI in 2018. It's built upon the transformer architecture and follows a two-step process: pre-training on a vast corpus of text and fine-tuning on specific tasks. The model is designed to predict the next word in a sequence, enabling it to generate coherent and contextually relevant sentences. In the process, the dense vector encoding semantic information is learned.

The GPT architecture has seen several iterations, each improving upon the previous in terms of model size, training data, and capabilities. Starting with GPT, which laid the foundation, OpenAI then introduced GPT-2, boasting a staggering 1.5 billion parameters. GPT-2 was initially deemed too powerful for public release due to concerns about potential misuse. However, its capabilities to generate coherent paragraphs of text from prompts made it a significant subject of research and discussion in the NLP community. Subsequently, OpenAI released GPT-3, an even larger model with 175 billion parameters. Its capabilities extend beyond text generation to tasks like translation, question-answering, and simple programming tasks without task-specific training.

One notable application of the GPT architecture is ChatGPT. Built on top of GPT-3, ChatGPT was designed to converse with users naturally and coherently. Its diverse training data, coupled with the massive model size, allows it to understand and generate responses across various topics, from casual conversations to more complex queries.

OpenAI released APIs for developers to integrate GPT models into their applications, opening up many opportunities. From chatbots to content-generation educational tools to programming helpers, the GPT API has empowered various industries to leverage the power of state-of-the-art language models in their products. This API also provides a function specifically for text embedding that can be used in this research.

2.3 News Dispersal Tracking

This chapter will explore the most representative research works in the field of news dispersal pattern tracks in chronological order. The development in news dispersal patterns can be seen as a condensed history of the development of NLP, as each approach would take advantage of the state-of-the-art NLP techniques at the time of publishing. The literature review approach will demonstrate the temporal and logical relation between the different approaches to solving the discussed problem but also hint at the limitation of technology availability at certain historical stages to illustrate the intuition behind each approach.

To our knowledge, there has not been previous news dispersal pattern analysis research conducted on social media news, potentially because the rise of social media as a news source was so fast and recent. There has been a lot of research on news dispersal pattern analysis on traditional paper-based and web-based news. Therefore, to maintain the relevance of our experiment, only the content-based news dispersal analysis models rather than meta-data-based ones will be explored for their platform-independent capability and interoperability. We do acknowledge the existence of other approaches, such as the entity tracing-based model shown in the paper by Minard et al. [65] or the knowledge graph modelling-based approach by Rospocher et al. [66] but they are not the focus of this very experiment. Macroscopically speaking, the development of content-level propagation analysis can be divided into two major stages: the link diffusion method and the content diffusion method.

2.3.1 Link Diffusion

The paper "Tracking Information Epidemics in Blogspace" by Adar and Adamic [7] in 2005 provides a comprehensive study on utilising URLs to track the spread of information in the online blog space. The research was conducted under the historical background when the newly recently emerged weblogs reshaped the World Wide Web. People are excited about this new invitation and sharing their thoughts, experiences and comments on others' posts on blogs. The blogs form networks and become conduits for information propagation within online communities, which makes them a perfect target for studying the pattern and dynamics of information dispersal. However, because blogs are a relatively new technology, the previous academic research on it is very limited and was mainly on exploring the size, activity, and dynamics of blog communities rather than specifically the spread of information. Inspired by the study of epidemics, the authors of the paper track the appearance and propagation of well-defined links to web pages external to the blog network, known as URLs, and consider any blogs containing a specific URL as "infected". The experiment was conducted on the comprised daily Blogpulse crawls for May 2003, which included 37,153 blogs with 175,712 URLs that "infected" more than two blogs. The experiment found that the source of infection can be determined given any URL-infected blog. Therefore, propagation routes of the information through the blog network can be uncovered and visualised with an example shown in Figure 2.6.

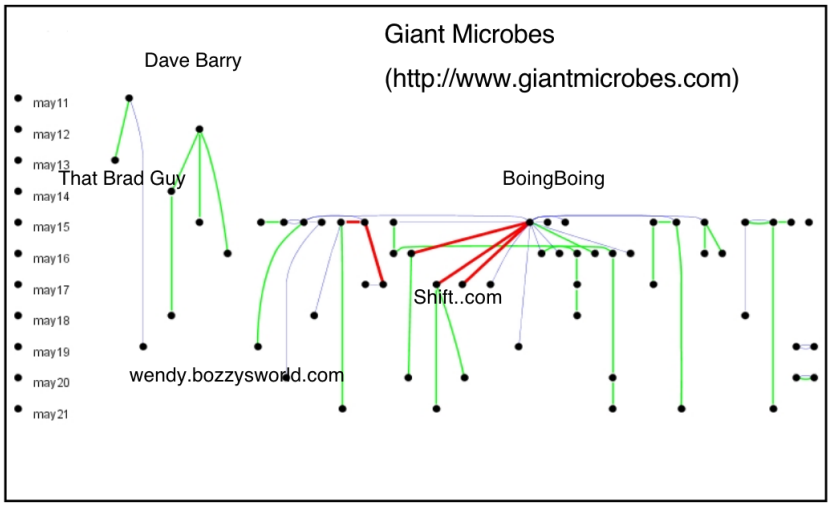

FIGURE 2.6: Infection tree of Giant Microbes with red, blue, and green links representing "via" links, explicit links, and inferred links, respectively. Cited from the paper by Adar and Adamic [7]

The textual similarity between blogs was determined using a cosine similarity metric on a TF-IDF weighted vector representing the textual content of the blog discussed in section 2.2.2.1. The similarity metric showed that unlinked pairs of blogs were less textually

similar on average than linked ones. Although the link diffusion approach may not function well today, as only an extremely limited portion of tweets contains URLs, the research builds the foundational theory of dispersed information is semantically similar.

2.3.2 Content Diffusion

With the advance of the NLP techniques, the content diffusion approach was introduced to overcome the limitation of the link diffusion model that only utilises links, which only make up a small portion of the news content. The early content diffusion approach proposed by Yang and Leskovec [67] in 2010 exactly matches the conserved content, mainly quotes or Twitter hashtags between news, to track the news propagation. On the other hand, the MemeTracker System proposed by Leskovec et al. [68] in 2009 allows quotes, or "memes" in this setting, to be slightly different in each propagation. The approach was further improved by Sjuen et al. [69] in 2013 with their NIFTY system introducing a clustering algorithm to track the slight change of quotes over time. However, at this moment, the approach still only relies on quotes, the short text enclosed in quotes that also only make up a limited portion of full text, similar to URLs. In 2015, Colavizza et al. [70] used similarity measures and local text alignment to use the full news text entirely. On the one hand, the proposed approach has finally fully utilised the entire content of the news article for the first time; on the other hand, the approach had a new limitation of being unable to match semantically similar content as the similarity measure, string kernel in this approach, only captures the appearance of text. Thereafter, a more advanced approach to address this new issue has been proposed by Vakulenko et al. [71] in 2016, which approaches the problem by phrasing art articles into "n-gram-like" grammatical relations at the sentence level to track the dispersal of information. While it has overcome the shortcomings of previous studies, the main limitations of this model are its inability to monitor more complicated content relationships and the dependency on the specific language grammatical parsers and the synsets in Wordnet [72].

2.3.3 Implicit Content Networks

Finally, in 2018, Joshi and Sinnott [8] proposed to track the information dispersal by modelling the implicit content network based on text embedding to capture the semantic similarities between information. Their model has summed up the experience of predecessors, standing on the shoulders of giants and becoming the state-of-the-art model. Their model also served as the reference model for this research.

The dataset used for training the model was The Single Media 1-Million Article Training Dataset [73] because it is a standardised tremendous dataset obtained from authentic online sources that include duplication, noisy data, incorrect language articles and unparsable text that best simulated the real word situation.

For the text pre-processing pipeline, the model first removed duplicated articles to avoid the output graph being too dense with similar nodes. And then tokenisation of the text using the Stanford Cor NLP package introduced in the paper by Manning et al. [74]. The benefit of tokenisation has been discussed in section 2.1.4.

The model chooses to embed the text with the Doc2Vec method. The method was one of the best-performing text embedding methods available at the time of writing. As discussed in section 2.2.3.1, the Doc2vec method proposed by Le and Mikolov [53] in 2014 was based on the word embedding method Word2Vec proposed by Mikolov et al. [5] in 2013. The text embedding approach vectorises documents into a high-dimensional vector as the distributed representation of the paragraph with semantic meanings encoded. An innovative approach taken by Joshi and Richard is that they reused the optimal hyperparameter found in the experiment by Lau and Baldwin [75] trained on two massive datasets, Wikipedia and AP News, and also intersected the word embeddings from larger models trained by Lau and Baldwin into theirs. The approach is coincidently similar to the large-scale pre-trained transfer learning idea of BERT discussed in section 2.2.4.2, which was only proposed a year later.

Due to the nature of the information dispersal problem, there are no easily accessible labelled data to evaluate the performance of model tracking information dispersal. Therefore, Joshi and Richard choose to evaluate the performance of the embedding method as an indicative performance benchmark of the entire model. The vectorisation model was then evaluated by the English Semantic Textual Similarity (STS) task from *SEM [76] and SemEval workshops [77–80] to test the performance of semantic models in different language domains to measure the pairwise similarity of sentences to evaluating the capital of text embedding method in capturing semantic information from the text. The result of evaluation using Pearson Correlation on SemEval 2016 STS Task 1 [80] have been shown in Figure 2.7. A finding that is worth mentioning to avoid confusion is that in the original paper where Joshi and Richard proposed the reference model [8], the SemEval 2016 dataset has been incorrectly referenced with SemEval 2016 task 2 dataset [81], wherein the actual experiment they used the dataset from SemEval 2016 task 1 [80]. This claim is supported by the fact that task 2 was on text alignment evaluated using an F1 score instead of a Pearson correlation. Also, it has been unfortunate they have not provided the exact number of the Pearson correlation but only show the result

in the graph with sparse Y-axis labels so that we have to approximate their PC in future comparisons.

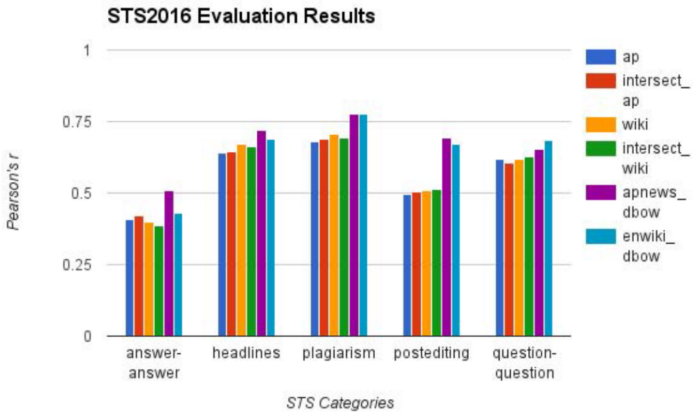

FIGURE 2.7: The PC of different models on SemEval 2016 STS Task 1 from paper by Joshi and Richard [8].

After vectorising the sentence, the cosine similarity [82] of sentence vectors was used as a distance for the k-nearest neighbour (KNN) [83] algorithm in conjunction with the hierarchical agglomerative clustering (HAC) [84] algorithm to create the propagation graph.

The kNN is employed as an efficient way to generate lists of neighbours. It creates undirected content exploration graphs by connecting semantically similar content. It is particularly useful for content exploration tasks because they don't aim to find isolated content clusters but rather connect related content. The paper found that when combined with HAC clusters as connection cluster seeds, they could select sections of the content graph for examination. By ordering these graphs over time, they could generate approximate content propagation graphs.

HAC acts complementary to kNN and allows for selecting well-defined and disjoint content clusters rather than a continuous range of overlapping content. The use of HAC was found to be critical for graph analysis applications, especially for selecting content clusters for investigation and for removing a significant number of connections that make graphs difficult to read at higher levels of kNN. HAC starts by assigning each vector to a singleton cluster and then iteratively links together any two clusters that exhibit the greatest similarity until a single global cluster exists. The paper used HAC with k-NNs to generate semantically similar content clusters. These clusters were then used as seeds for generating a content propagation graph.

Lastly, the author evaluated the efficiency of the model by recording the execution time of the model in datasets of different sizes. And provide a discussion of the applications

of the model. It involves two major categories: exploratory analysis and graph analysis. The exploratory analysis includes related content consumption, in-context comparison, and comparative analysis. The graph analysis includes content propagation graph visualization, news source analysis, and generative art.

Article	Overview of Strategy	Architecture	Notes
Bengio et al. [46] in 2003	Embeddings are derived as a by-product of training a neural network language model.	Neural Net	Commonly referred to as the first neural network language model.
Bengio and Senecal [47] in 2003	Makes improvements on the previous paper by using a Monte Carlo method to estimate gradients, bypassing the calculation of costly partition functions.	Neural Net	Decreased training times by a factor of 19 with respect to previous study [46].
Morin and Bengio [48] in 2005	Full softmax prediction is replaced by a more efficient binary tree approach, where only binary decisions at each node leading to the target word are needed.	Neural Net, Hierarchical Softmax	Report a speed up with respect to previous study [47] (over three times as fast during training and 100 times as fast during testing), but at a slightly lower score (perplexity).
Mnih and Hinton [49] in 2007	Among other models, the log-bilinear model is introduced here. Log-bilinear models are neural networks with a single, linear, hidden layer [39].	Log-linear Model	First appearance of the log-linear model, which is a simpler model, much faster and slightly outscores the model from Bengio et al. (2003).
Mnih and Hinton [39] in 2008	Authors train the log-bilinear model using hierarchical softmax, as suggested in the paper by Morin and Bengio [48], but the word tree is learned rather than obtained from external sources.	Log-linear Model, Hierarchical Softmax	Reports being 200 times as fast as previous log-bilinear models.
Collobert and Weston [38] in 2008	A multi-task neural net is trained using unsupervised data and supervised data such as SRL and POS annotations. The model jointly optimizes all of these tasks, but the target was to learn embeddings.	Deep Neural Net, Negative Sampling	First time a model was built primarily to output just embeddings. Semi-supervised model (language model + NLP tasks).
Mikolov et al. [5] in 2013	Introduces two new models, namely CBOW and SG. Both are log-linear models using the two-step training procedure. CBOW predicts the target word given a context. SG predicts each context word given a target word.	Log-linear Model, Hierarchical Softmax	Trained on DistBelief, which is the precursor to TensorFlow [50].
Mikolov et al. [42] in 2013	Improvements to CBOW and SG, including negative sampling instead of hierarchical softmax and subsampling of frequent words.	Log-linear Model, Negative Sampling	SGNS (skip-gram with negative sampling), the best-performing variant of Word2Vec, was introduced here.
Bojanowski et al. 2016	Embeddings are trained at the n-gram level to help generalization for unseen data, especially for languages where morphology plays an important role.	Log-linear Model, Hierarchical Softmax	Reports better results than SGNS. Embeddings are also reported to be good for composing into sentences document embeddings.

TABLE 2.2: The overview of mainstream prediction-based text embedding models cited from the survey by Almeida and Xexéo [13].

Chapter 3

Methodology

In the preceding chapter of the literature review, we first symmetrically reviewed the theory, implementation and compromise of the different text pre-processing steps in the pipeline form the bedrock of NLP, then illustrated the development and advancement of text embedding techniques as the foundation for computational linguistic procedure with a focus on the detailed architecture and design of the models, and lastly explored the history of approaches on solving information dispersal tracking problem bounded by the limitation of available text representation methods at certain historical period. In this chapter, by standing on the shoulder of the giant to identify the knowledge gap and potential research opportunities in the past literature, we will first clarify the research objective of the thesis, then demonstrate the experiments dedicated to achieving research goals and finally illustrate the experimental setup and dataset before the forthcoming chapters on detailed discussion for separate experiments.

3.1 Research Objectives

This section clarifies this thesis's research objectives, which are derived by identifying the knowledge gap and potential research opportunities in the past literature. At the very end of the literature review in section 2.3.3, we meticulously inspected the methodologies of the stat-of-the-art approach for information dispersal tracking proposed by Joshi and Richard [8] to track information propagation by modelling implicit content networks. The Doc2Vec semantic text embedding model empowers the paper's content network. The Doc2Vec is a variant of the Doc2Vec method using the prediction-based approach to produce static text embedding discussed in section 2.2.3.1. Josh and Richard chose such an approach because Doc2Vec was the state-of-the-art text representation model

at the time when they published their report in 2018. However, at the time of conducting this research, the newly proposed contextual embedding introduced in section 2.2.4, taking advantage of transfer learning pre-trained using large corpus and advanced neural network structure of CNN, LSTM, and transformer to dynamically capture the syntactic and semantic meaning into the text embedding from the context has surpassed traditional approaches in performance dramatically.

Therefore, the first research goal of this thesis is to investigate modelling implicit content networks using large pre-trained contextual embedding models. Changing the text representation model would also require reinvestigating and experimenting with the text pre-processing pipeline because different embedding methods have different characteristics and preferences for input text.

The second research objective is to optimise the model for the specific social media news post language domain. This objective is also related to the change from 2018. As the introduction demonstrates, the mainstream medium for news consumption has shifted from web news to social media platforms. The social media post language domain significantly differs from the web news with shorter text, less formal language, emoji, hashtags and more. The change would also require explicit effort in investigating the most suitable text preprocess pipeline and text embedding model.

The third research goal is also related to the advancement of the times. Cloud computing, epitomized by platforms such as the popular Amazon Web Services (AWS), has ushered in a transformative era in technology and business. Before its advent, companies had to invest heavily in physical infrastructure, which was both capital-intensive and lacked scalability. Today, public clouds such as AWS and its contemporaries have democratized access to powerful computing resources, allowing startups and large enterprises to scale operations on demand, paying only for the resources they use. This paradigm shift has accelerated innovation, as businesses can now experiment, deploy, and iterate applications at unprecedented speeds. Moreover, with the global reach of cloud services, businesses have been empowered to operate and serve customers across borders seamlessly. Cloud computing has redefined the technological landscape and reshaped the fabric of global commerce, making digital solutions more accessible, efficient, and cost-effective, where investigating the parallel performance on the cloud has been unavoidable for developing our proposed model with future industrial applications planned.

3.2 Experimental Design

This section describes the experiment's methodology for the research goals listed above. The problem, by its nature, requires a complex system with a well-planned pipeline of strategies. Based on one of the most foundational ideas of computing, this thesis aims to use the idea of the divine and conquer to split the ultimate goal of research into three incremental separate experiments. The implemented experiments will be explained by theories, critically, quantitatively, and practically evaluate the performance in the forthcoming chapters.

The first experiment in chapter 4 focuses on exploring the text embedding and text pre-processing techniques demonstrated in the literature review and benchmarking their performance in the language domain of social media news. The second experiment in chapter 5 aims to recreate the reference paper's implicit content networks model and test its compatibility with the new text embedding pipelines found to perform the best in the previous experiment and benchmark them on their efficiency. The third experiment in chapter 6 is designed to introduce a new metric for evaluating the model: the parallel computing capability based on Amdahl's and Gustafson's Law using running time measurement on various platforms.

3.3 Experiment Setup

A comprehensive and detailed description of the experiment setup, both software and hardware, is essential to improve the repeatability of the experiment and ultimately improve the trustworthiness of the entire thesis. While the software information is vital as different implementation versions of a model can result in different performance measurements, the hardware information is equally essential because the quantitative analysis of models' efficiency relies on measuring execution time, which the hardware setting could heavily impact.

3.3.1 Software Setup

Software-wise, the experiment only requires one programming language, Python. This is not solely because Python has also been chosen as the only programming language to use by the reference model [8], but also because it has been essential for many other NLP or, generally, any machine learning-related research and projects. The Python programming language provided an extensive ecosystem of libraries, especially libraries that facilitate text processing, modelling and analysis, such as NLTK and transformers, which will be

discussed in detail in chapters for specific experiment setups, making it the perfect selection for avoiding reimplementing many essential NLP algorithms. The study conducted by Zehra et al. [85] compared Python with C++. As a high-level object-oriented interpreted language, Python runs slower than C++ as a low-level object-oriented compiled language. However, the research projects, by their nature of running on a small scale with a small user count, are generally not speed-sensitive. Therefore, Pythan has been the most popular programming language because its readability, simplicity, and consistent syntax make it accessible for novice researchers and seasoned practitioners, ensuring that codebases are maintainable and easily shared across the scientific community.

The reference model used two different versions of Python: Python 2.7 for training and generating vectors and Python 3.5 for parsing and manipulating articles. Running two versions of Python on a single system is only made possible by the Anaconda scientific platform. This research will also be utilising Anaconda for package management. The research by Maji et al. [86] introduced Anaconda, emphasising the enormous impact when dealing with Python packages in an HPC environment. In short, Anaconda is a powerful open-source distribution platform designed specifically for Python and R, two of the most widely used data science and machine learning languages. It is built on creating consistent and isolated environments, which is crucial when conducting experiments or developing projects. By using virtual environments, Anaconda ensures that different experiments can be run with their specific dependencies without any interference, thereby maintaining the accuracy and reproducibility of results. This is achieved through Anaconda's environment manager, Conda, which allows users to create, manage, and switch between distinct project environments, each with its tools and libraries. This approach prevents software conflicts and streamlines the development workflow, making our research more efficient and effective.

In the Anaconda environment, 3.9.18 is selected as the version of Python for the experiment. The model numbers have three components to be interpreted. The first component in version number, such as "3" in "3.9.18", is the major version of the software and typically signifies significant changes, especially ones that might break backward compatibility. For Python, two major versions are widely recognized: Python 2 and Python 3, which are not fully compatible. The "9" is the minor version number. It represents feature updates and improvements that are backward-compatible. New functionalities may be added in a minor version without breaking existing functionalities. The "18" is the micro version number, sometimes called the patch version number. Patch versions are typically released to fix security vulnerabilities, bugs, and other issues that don't introduce new features.

With this preliminary knowledge, the reason for selecting 3.9.18 can be easily explained. First of all, all the NLP package introduced was only on Python 3, as Python 2 was already deprecated on 1 Jan 2020 [87]. The earlier functional model is intended to be selected because it can support more retro packages used by the reference model and because the tried-and-truth early model has been involved in more experiments from published papers that make the performance comparison more accessible for the reader of this thesis where newer functional models may involving performance optimisation of internal functions making performance comparison invalid. Python 3.9 is the earliest active maintained version that runs on all the experiment hardware, especially on Apply Silicon chips, to be introduced in the next section. On the other hand, for a given functional version, a newer patch version is better for providing a more secure and robust platform without affecting the experimental result.

The last software tool the thesis takes advantage of before going into more detail about package use and algorithm implementation in chapters for specific experiments is the Package Installer for Python (PIP). The PIP can not only install and manage additional libraries and dependencies that are not part of the Python standard library but also can be used to ensure the package version is consistent across experiment platforms. To ensure the package version consistency, on the initial set-up machine, the command `pip freeze > requirements.txt` saves the version data of the installed package into the `requirements.txt` file. Running the command `pip install -r requirements.txt` on other machines after files are synced to install the exact version of packages the initial machine installed.

3.3.2 Hardware Setup

Due to experiments 2 and 3 involving benchmark efficiency of the model using running time measurements, it is essential to clearly state the hardware specifications of the machine conducting the experiments to ensure the credibility and reproducibility of the result. The experiments employed three physical and one virtual machine with three Central Processing Unit (CPU) platforms and running four different operating systems to validate the result across different computation platforms. This is designed to cover all the potential use cases for the comprehension experiment and test the robustness of the model. The detailed specifications of the hardware have been stated in Table 3.1 where the Intel Xeon is the x86 workstation platform, Intel Core is the x86 personal computer platform, and Apple M1 is the new Apple Silicon ARM architecture processor for Mac. For Mac system, the GPU acceleration hardware `mps` have its driver ready to be enabled by deep learning package such as `PyTorch` to take advantage of. Win11 system was also equipped with Nvidia RT3060, so CUDA toolkit version 12.2 was set up

to enable CUDA GPU acceleration. For the rest of the experimental platforms, namely Win10 and Linux, GPU support was deliberately avoided for the comparison purpose of the experiment.

Apart from the physical hardware, it is equally important to conduct the experiment on different operating systems because the DS-based Windows 10 and 11 and UNIX-based MacOS and Linux are dramatically different in underlying kernel design, which would affect the multi-threading implementation in experiment 3.

Reference Name	**Mac**	**Win10**	**Win11**	**Linux**
Format	Laptop	Desktop	Desktop	Virtual Machine
Operating System	MacOS 14.0	Windows 10 Home 22H2	Windows 11 Pro 22H2	Ubuntu 22.04.3 LTS
CPU	Apple M1 Pro	Intel Xeon E5-1650v3	Intel Core i5-10400F	Intel Core i5-10400F
Physical Cores	8	8	8	2
Logical Cores	8	16	16	4
Memory	16GB	32GB	16GB	4GB
GPU	M1 Pro 14-Core-GPU	GTX 1070	RTX 3060	LLVM

TABLE 3.1: Detailed configuration of the machines participating in the experiment.

3.4 Dataset

The experiment involves six different datasets from two categories of data serving different purposes. There are labelled Semantic Textual Similarity (STS) datasets from Semantic Evaluation (SemEval) for the performance evaluation of text embedding methods in experiment 1, and unlabelled Twitter rumour detection datasets for training unsupervised models, and efficiency and parallel computing capability evaluation in experiments 2 and 3.

The STS method remains the most renowned evaluation method for text embedding models, as suggested in the survey on word embedding evaluation methods by Bakarov [88]. To understand the reason for using STS datasets, it is essential first to introduce the STS task, an NLP task that aims to measure the semantic similarity between two pieces of text. Given two sentences, the objective is to rate their similarity on a scale, often from 0 to 5, where a score of 0 indicates that the sentences are semantically unrelated, and a score of 5 indicates that they are semantically equivalent, even if they are lexically different. The STS task was chosen over the binary tasks, such as paraphrase

identification because STS provides a graded similarity measure for more accurate performance evaluation in capturing slight semantic differences. The datasets used in the experiments are benchmark datasets from the SemEval challenges with a ground truth Pearson correlation (PC) value assigned to each sentence pair created to facilitate the training and evaluation of models on the challenges. The SemEval is a series of workshops and competitions organised annually in computational semantics to promote developing and assessing computational models for evolving semantic-related tasks reflecting the changing priorities and interests of the NLP community. It provides standardised datasets and evaluation metrics, enabling researchers and practitioners worldwide to benchmark their models against others in the community. Results are published at the end of each SemEval challenge, providing insights into the methodologies used by the top-performing teams promoting knowledge sharing and further research. The specific SemEval STS dataset used in the research are listed below:

- The "tweet-news" and "headlines" dataset from the SemEval 2014 task 10: "Multilingual Semantic Textual Similarity" [78] each having 750 entries of sentence pair and gold standard PC.
- The dataset from the SemEval 2015 task 1: "Paraphrase and Semantic Textual Similarity (PIT)" [89] with 972 entries of sentence pair and gold standard PC.
- The "headlines" dataset from the SemEval 2016 task 1: "Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation" [80] with 1498 sentence pairs but only 249 of the parish have corresponding

To experiment on the Twitter social media post language domain, not only all the Twitter datasets from the history of SemEval have been selected, but also the headlines datasets, which are very similar to the length and format of Twitter, were put into the experiment.

Although STS labelled datasets are perfect for evaluating the model's performance, the data annotation tasks time and human effort, so limited available labelled data are insufficient for benchmarking the model's efficiency. Also, the golden rule of machine learning or statistics in general is to infer population from a sample, which requires the sample to capture as much of the characteristic and distribution of the the population data as possible. Therefore, unlabeled Twitter15 and Twitter16 data with 1490 and 818 entries of real-world Twitter posts are also used to benchmark the model's efficiency. This data is essential because it is as noisy and messy as the realistic social media post-text data, mainly since it includes domain-specific languages, emojis, and hashtags. Twitter 15 was collected for developing the rumour detection model in the study by Liu

et al. [90]. Similarly, Twitter 16 was also collected for rumour detection research by MA et al. [91]. The two datasets are named based on the years the datasets were collected, 2015 and 2016, respectively and are commonly used in research, especially in studies related to rumour detection on Twitter, which is not exactly the focus of this research. Still, they contain extensive collections of actual tweets containing noise and particular domain-specific format, where most are related to news as news is the main subject of study on rumour detection.

Chapter 4

Experiment 1: Text Embedding Performance Evaluation

The reference model by Joshi and Richard [8] on modelling implicit content networks for information dispersal tracking includes a series of steps in the order of text processing, text embedding, clustering, and content graph generating. The first experiment will mainly focus on the first two sections, text pre-processing and text embedding. The first one is because text pre-processing requires domain-specific configuration for the downstream model to perform well, and this research is in the domain of news-related posts from social media platforms, which is dramatically different from previous works on blog posts that need to be thoroughly investigated. The second one, text embedding, is the area of study response for bringing the impression of rapid development of the NLP field as the rise of the large-scale pre-trained language models introduced in the section 2.2.4 like BERT and GPT powered by transformer and transfer learning. The quantitative changes of these emerging technologies have led to qualitative changes. For example, ChatGPT, developed by OpenAI, which emerged based on GPT LLM, has gained phenomenal public attention. Therefore, experiments on novel text embedding techniques would have the most significant potential for performance improvement over the previous solutions. This chapter on the first experiment will be divided into two parts. We will first apply different text embedding strategies to all available labelled datasets without further text pre-processing and benchmark using Pearson Correlation (PC) to have a general idea of the characteristics of each dataset and embedding methods. Then, we will further optimise each embedding method on the most representative dataset to understand the effectiveness of each text pre-processing strategy on the Twitter post domain and each embedding approach.

4.1 Embedding Methods

In this experiment, we implement major text embedding methods introduced in the section 2.2 to apply on testing labelled SemEval STS dataset mentioned in the section 3.4 and evaluate the methods with the PC metric. For embedding models requiring training, the Twitter 15 and Twitter 16 unlabeled dataset is used for unsupervised learning. We specifically included "headlines" datasets from SemEval 2016 to establish a direct comparison to the reference model, as it was also involved in the reference model with the claimed performance cited in Figure 2.7. The implemented embedding methods for this experiment are listed below:

- **TF-IDF**: The `TfidfVectorizer` from the Scikit-learn package [92] was used to initilise the model with only parameter of `stop_words="english"` specified and rest parameters remain default. Before transforming the sentence pairs, the model was fitted and trained on the combined Twitter 15 and Twitter 16 datasets.
- **Doc2Vec**: The DBOW Doc2Vec implementation from Gensim package [93], the same Python package used in the reference model, was used for this experiment. Joshi and Richard [8] have not provided the exact value of the hyperparameters they used but pointed to their reference model from the paper by Lau and Baldwin [75] instead. Therefore, in our experiment, we used the same best hyperparameter found by Lau and Baldwin for the STS task using DBOW, which is `vector_size=300, window=15, min_count=1, negative = 5, workers=4, epochs=400`. Like the TF-IDF, the Doc2Vec model was also trained on the combined Twitter 15 and 16 datasets.
- **BERTweet**: The `AutoTokenizer and AutoModel` from the `transformer` package was used to load the pretrained model `vinai/bertweet-base` proposed by Nguyen et al. [63] from huggingface. The embedding for each word is obtained from `last_hidden_state` and averaged the word embedding into the sentence embedding.
- **SBert**: The `sentence_transformers` package proposed by Reimers and Gurevych [64] was used to invoke the pre-trained model `all-MiniLM-L6-v2` to encode the sentence.
- **OpenAI**: The OpenAI [94] package is used to call the `text-embedding-ada-002` text embedding API. The `tenacity` package is used to wait and retry the API call to avoid exceeding the rate limit. Local caching is established to reduce repetitive API calls as it is a paid service priced at 0.0001 USD per 1K tokens.

The above text embedding models output the dense continuous vector captured the semantic information of the post. No additional text pre-processing techniques are applied for the first subsection of the experiment, as it will be the topic for the next subsection. To quantitative compute the semantic similarity across sentence pairs to perform the STS task, the cosine similarity of two sentence embedding vectors is computed with output ranging from 0 to 1, indicating how similar two sentences are semantically, with 1 suggesting two sentences are semantically identical. The `cosine_similarity` from `sklearn.metrics.pairwise` package is used to implement the cosine similarity algorithm shown in equation 4.1.

$$\text{cosine similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}||_2 \times ||\mathbf{B}||_2} \tag{4.1}$$

The cosine similarity computed based on the vector output from text embedding models is then set to compare with the human-labelled gold standard similarity score of sentence pairs to evaluate the capability of text representation models capturing semantic information from the text. The evaluation statistical measurement for the STS task used in the SemEval challenge, PC, quantifies the strength and direction of a linear relationship between two variables and is used for such comparison with its formula shown in Equation 4.2. Capturing the linear relationship avoids the extra step to normalise the cosine similarity range from 0 to 1 and the gold standard range from 0 to 5 for comparison.

$$PC_{XY} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2 \sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{4.2}$$

With the experiment's methodology explained, the experiment's result PC has been listed in Table 4.1 and visualised in Figure 4.1. Since Doc2Vec was a prediction model powered by a neural network that initialises randomly at the training stage, as discussed in Section 2.2, it will result in stochastic text embedding and PC at each run. Therefore, for the Doc2Vec, we recorded the median PC over five runs for each dataset.

Embedding	**2014 twitter-news**	**2014 headlines**	**2015 PIT**	**2016 headlines**
TF-IDF	0.6240	0.5364	0.1602	0.6075
Doc2Vec	0.0836	0.0647	0.1654	-0.0286
BERTweet	0.2552	0.3169	0.1980	0.3040
SBert	0.8071	0.7916	0.6297	0.7960
OpenAI	0.8413	0.8303	0.7246	0.8457

TABLE 4.1: Table of PC of embedding methods on SemEval STS datasets with no additional text pre-processing technique employed.

From the result, multiple findings can be discussed. First of all, performance-wise, we first compare our model with the reference model. The PC evaluation result of

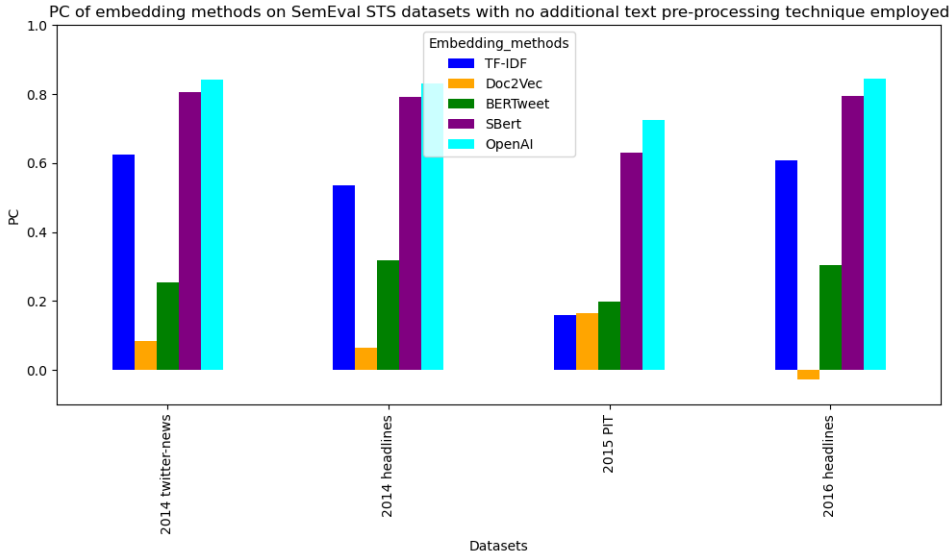

FIGURE 4.1: Graph of PC of embedding methods on SemEval STS datasets with no additional text pre-processing technique employed

different Doc2Vec models on various tasks has been shown in Figure 2.7 cited from the reference paper by Joshi and Richard [8]. As discussed in Section 2.3.3 that, we can only approximate the PC value from the figure because it is unfortunate Joshi and Richard have not included the exact values in the paper but shown results in the graph with sparse Y-axis labels instead. The "headlines" dataset SemEval 2016 task 1 [80] have been involved in both our and reference model's experiment. The overlapped dataset built a bridge to establish the direct comparison of models across two experiments. In Figure 2.7, the top two performing models are Doc2Vec models "ap_dbow" and "enwiki_dbow" trained on huge English Wikipedia and AP News datasets first proposed by Lau and Baldwin [75]. Then there are four Doc2Vec models trained by Joshi and Richard [8] on relatively smaller *The Signal Media 1-Million Article Training Dataset* [73]. With all six models considered, it is evident in the figure that no models, even the best-performing model, have a PC exceeding the line for 0.75. However, in our experiment, `SBert` and `OpenAI`, two models utilising novel pre-trained large-scale language model text embedding methods, have achieved 0.796 and 0.8457 PC, respectively. This astonishing result shows that the experiment has been aimed in the correct direction to fulfil the research objective of improving the model performance. In theory, the performance of the pre-trained model can be further improved by fine-tuning due to the transfer learning capability. However, for this specific research, we choose not to do so because the STS is just an intermedia task to evaluate the embedding model, and the ultimate goal of the research is still to build the implicit content network with the most suitable embedding approach. Over-optimising the model for the STS dataset may result in adverse effects such as overfitting.

Besides the performance, we found that TF-IDF trained on Twitter 15 and 16 datasets best performed on the 2014 "twitter-news" dataset, among other SemEval datasets. This phenomenon does align with the expectation because the "twitter-news" is the dataset most similar to the training Twitter 15 and 16 datasets, which have been well accepted as the representative datasets reflecting the actual Twitter post in terms of language domain. By inspecting the datasets, 2014 is the only dataset among all STS datasets with hashtags and emoji to reflect the noisy Twitter post language. Therefore, the 2014 "twitter-news" dataset will be the only dataset to use in the follow-up text pre-processing experiment.

Lastly, this experiment has validated various theories relating to text embedding learnt from past literature. As discussed in the previous paragraph, TF-IDF will perform better when the test corpus and train corpus are in similar language domains. The same rule also applied for Doc2Vec, but because the neural network has more parameters to capture more complex semantic relations, it would require an extensive training set to perform well. This explains why our Doc2Vec implementation performances are much worse than the reference model trained on the much larger *The Signal Media 1-Million Article Training Dataset* [73]. This is one reason, among many other reasons, including the lack of text pre-processing, which will be explored in the next section. The BERTweet, purposely optimised for the Tweet post language domain, has been performing surprisingly unsatisfactorily. By investing in the related research using the same model, we found the model is meant to connect additional neural networks after the transformers to perform higher-level tasks such as text classification rather than directly computing text embedding. Besides, in our use case, the word embedding taken out was averaged to obtain the sentence embedding, which suffered the same issue of losing sentence order and structure information similar to BOW. In contrast, the Sentence BERT model for the general language domain leveraged Siamese and triplet network structures to fine-tune BERT for producing semantically meaningful sentence embeddings, which performed much better than BERTweet. The STS task was involved in the initial training stage of the Sentern BERT, which is not surprising given its superior performance in this experiment. The OpenAI has proven to be the best-performing model across all datasets because it has been the largest model driven from GPT in terms of both training size and parameter size. Unfortunately, the model is neither open-source nor free to use to investigate further. This result indeed suggests that the study of NLP in the foreseeable future will rely heavily on LLM powered by solid computing capability and extensive training sets.

4.2 Text pre-processing

The experiment did not come to an end after the research goal had been achieved by outperforming the reference model on the STS task in the last section. The indispensability of text pre-processing has been expressed in the literature review. Therefore, this experiment section aims to investigate the optimised text pre-processing pipeline for specific text embedding methods in the specific language domain of Twitter news posts. This reflects the theory of two significant factors affecting the performance of the text pre-processing pipeline. Specific embedding models are sensitive to different aspects of the data, which require different text pre-processing pipelines. Also, unique language domains require a particular text pre-processing pipeline before feeding into general purposed NLP models. This experiment will also enrich the previous experiment by unleashing the full potential of each embedding model for a more impartial comparison. This experiment is further divided into two subsections. The first subsection will implement and experiment with the text-pre-processing approach taken by the reference paper. The second subsection will conduct an exhaustive search to find the optimised pipeline for each text embedding method from all available text pre-processing strategies.

4.2.1 Reference Model

In the reference paper by Joshi and Richard [8], the text pre-processing strategies of duplication removal, tokenisation, and lower casing were employed. However, the duplication removal will be skipped for this experiment as no duplicate sentence exists in the specific testing set. The reference paper used the Stanford CoreNLP package [74] for tokenisation and lower casing, which has been updated and replaced by the new stanza package [95] with the same functionality but improved efficiency. For models requiring training, namely TF-IDF and Doc2Vec, the same pre-processing pipeline is also applied to the training set as applied to the testing set. The experiment result of PC and percentage improvement of text embedding methods with reference model's text pre-processing pipeline over models without them is shown in Figure 4.2 and visualised in Figure 4.2.

The result suggests that the reference text pre-processing pipeline only improved the performance of the more traditional non-pre-trained models. This result shows the text pre-processing technique might not consistently have a positive effect on model performance, which aligns with the claim learned from the literature review that the text pre-processing is compromising to cut off less informative data at the risk of reducing the model performance and need to be applied with caution. The second finding was that

Embedding	Original PC	Reference Model PC	% Improve
TF-IDF	0.6240	0.6267	0.43
Doc2Vec	0.0836	0.1064	27.27
BERTweet	0.2552	0.2392	-6.26
SBert	0.8071	0.8067	-0.04
OpenAI	0.8413	0.8385	-0.33

TABLE 4.2: Table of PC and percentage improvement of text embedding methods with reference model text pre-processing pipeline compared with no pre-processing.

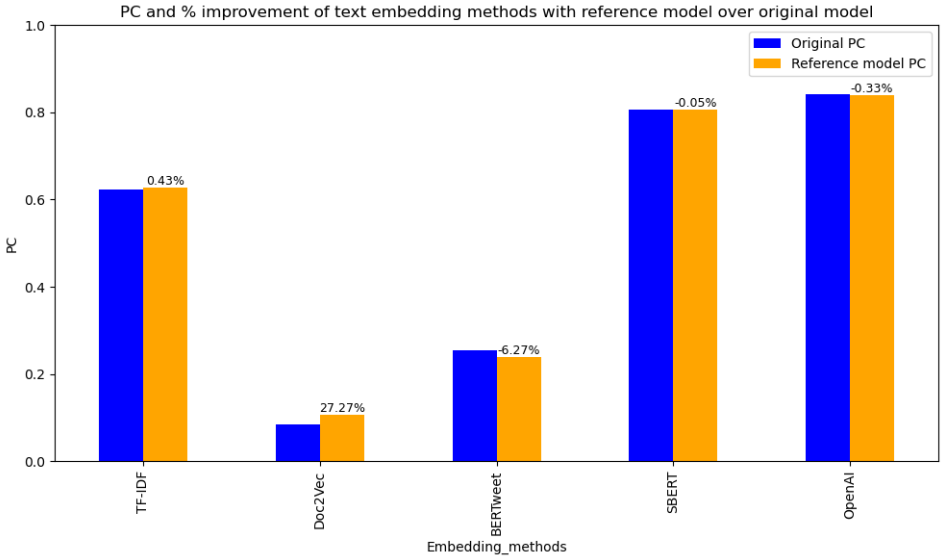

FIGURE 4.2: Graph of PC and percentage improvement of text embedding methods with reference model over original model.

Doc2Vec experienced the most significant performance improvement after applying the reference text pre-processing pipeline. This is reasonable because the reference text pre-processing pipeline was purposely developed to optimise performance for the Doc2Vec model in the reference paper. This conclusion directly inspired the next stage of the experiment of finding the optimal pipeline for each embedding method.

In this subsection, we recreated the reference STS model with the exact Python package implementation and Text pre-processing process as mentioned in the paper by Joshi and Richard [8]. However, we stopped here and chose not to further revivificate the reference model by training the model on the same *The Signal Media 1-Million Article Training Dataset* [73] for two reasons. The first reason is the language domain being investigated in this thesis is the social media news posts, which are dramatically different from the long-form online article from the dataset. The second reason is that while 1 million data entries are indeed big enough to provide good theoretical training results. But a larger dataset also makes the pre-processing significantly more computationally heavy that, even on our best CUDA-enabled Win11 platform, it would take around three years to tokenise the text to train the Doc2Vec model.

4.2.2 Exhaustive Search

As concluded from the previous experiment, with the correct pipeline of text pre-processing strategies optimised for specific text embedding strategies, the model performance can be improved dramatically. Therefore, this subsection will implement and experiment with numerous text pre-processing techniques discussed in the literature review section 2.1 and two pre-processing strategies, emoji/emoticons to text and hashtag expansion, are specially proposed for the Twitter post language domain. The reference name, implementation, and explanation of the involved text pre-processing strategies are shown below:

- **EMO, Emoji and emoticons to text**: The `demojize` function from the `emoji` package is used to transform the union code emoji symbol to the corresponding text to be recognised by downstream models. A custom-collected dictionary of emoticons handles emotions and replaces them with text. For example, ":)" is replaced by "smile". The approach transfers the traditionally unsupported data formats to generally accepted text format, which is a better alternative approach than noisy removal discussed in section 2.1.2.
- **HAS, Hashtag expansion**: There are two logics involved in this approach. First, if capital letters are inside the hashtag, the text will be interpreted as camel cases and split based on capitalisation. Otherwise, in most cases where hashtags are composed of all lower letters, the `wordsegment` is used to split the words from the hashtag. This is similar to the tokenisation process discussed in the section 2.1.4.
- **PUN, Punctuation removal**: The Python `string.translate()` function is used to translate all `string.punctuation` into blank spaces. This is classified as the noisy removal process discussed in section 2.1.2.
- **LOW, Lowercasing text**: The Python `string.lower()` is used to lower all the alphabet from the text to lowercase. This process is part of the normalisation process to interpret identical words despite the capitalisation with the same vector discussed in section 2.1.5.
- **LEM, Lemmatisation text**: The text is first tokenised by `word_tokenize()` function from `nltk` package to obtain tokens from the text. Then, to ensure accurate context-aware lemmatisation, the part of speech tag of each token is obtained by `pos_tag()` function also from `nltk` package. Lastly, the lemma of each word is found in the dictionary using `lemmatise` function in `WordNetLemmatizer` class from `nltk.stem` package. The theoretical knowledge about lemmatisation has been explored in section 2.1.5.1.

- **STE, Stemming text**: The `stem()` function from the `PorterStemmer` class in `nltk.stem` package is used to stem the words in the text. The theoretical knowledge was discussed in section 2.1.5.2.
- **STO, Stopword removal**: The `stopwords` set of English stopwords from `nltk.corpus` is downloaded and used to compare and remove the stop words from the text. The literature review in section 2.1.6 discusses the process in detail.

The condition of this experiment is that the text pre-processing technique can only applied in pipelines strictly in the order they were introduced in the above list. This is the measure to reduce the computation complexity of the research while not likely to miss out on a good pipeline backed up by the theoretical pipeline order of the text pre-processing discussed in section 2.1. For example, hashtag expansion must be done before punctuation removal because otherwise, the hashtag without "#" will not be captured by the hashtag expansion algorithm. With the condition set, the number of iterations that each embedding method needs to run by testing all the possible combinations of the listed eight text pre-processing techniques can be computed by n choosing k for k in the range of 0 to n. The formulas are shown in Equation 4.3.

$$\sum_{k=0}^{n} \binom{n}{k} = \sum_{k=0}^{n} \frac{n!}{k!(n-k!)} \tag{4.3}$$

Therefore, for eight text pre-processing techniques, there are 128 combinations for each embedding method, which is a total of 640 embedding to be done for the exhaustive search. Putting such a long list on this page would make it too bloated so that the complete list of experiment results will be included in the Appendix A. For our experiment, we will first observe the best-performing text pre-processing pipeline for future experiments. We will also analyse the improvement in PC over unprocessed situations to unearth the characteristics of different text embedding and text pre-processing strategies. The experimental results have been listed in Table 4.3 and visualised in Figure 4.3.

Embedding	**Origin PC**	**Best PC**	**% Improve**	**Best Pipeline**
TF-IDF	0.6240	0.6684	7.12	LEM-STE
Doc2Vec	0.0836	0.1366	63.40	EMO-HAS-LEM-STE
BERTweet	0.2552	0.3379	32.41	HAS-PUN-LEM
SBert	0.8071	0.8105	0.42	HAS
OpenAI	0.8413	0.8425	0.14	HAS

TABLE 4.3: Table of PC, percentage improvement, and pipeline of best text pre-processing pipeline for each text embedding method compared with no pre-processing.

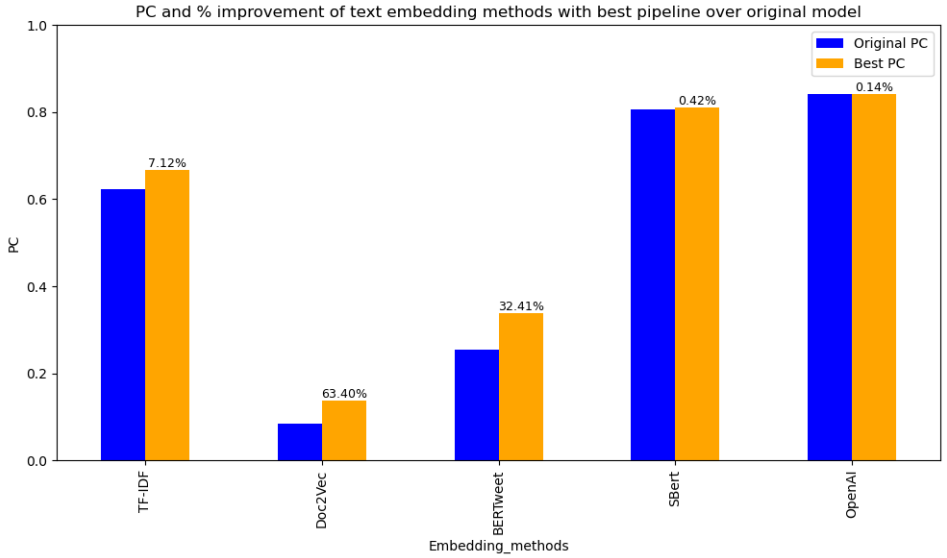

FIGURE 4.3: Graph of PC and percentage improvement of text embedding methods with the best pipeline over the original model.

Overall, all the embedding models have been improved by applying text pre-processing techniques at various degrees through different pipelines. Both TF-IDF and Doc2Vec have gained performance improvement through the normalisation process of lemmatisation and stemming. This does not conform to common practical, which would only choose one technique between lemmatisation and stemming rather than both. The hypothesis to explain the phenomenon is that two embedding models were underperforming mainly due to overfitting issues caused by minimal training data. Therefore, by actively overdoing the normalisation to reduce the vocabulary size, the model will be regularised and perform better. The language domain specified preprocess techniques, emoji-to-text and hashtag expansion, causing positive effects on Doc2Vec but not TF-IDF. The observation could be interpreted that TF-IDF is not as sensitive to non-textual data compared to Doc2Vec. The BERTweet benefits from hashtag expansion, punctuation removal, and lemmatisation mainly due to the reduced noise in text. This indirectly suggests that similar pre-processing has been done on the training set when the models were initially pre-trained. Lastly, Sentence BERT and OpenAI models performed well before the pre-processing, so the improvements are less significant than those of other models. However, the models can still benefit from the language domain-specific hashtag expansion process as the hashtag only appears in Twitter posts. Conversely, emojis can appear elsewhere and may have been included in the pre-train phase. It was also noticed that case lowing does not change performance, suggesting that the text pre-processing technique has been included in the embedding process. This reflects the technical advance of the more recent pre-trained model encapsulated with the necessary pre-processing pipeline for easier usage. Another important finding is that the word normalisation technique will reduce the performance of the pre-trained large-scale language

models. As claimed before, all text pre-processing techniques must be used cautiously because they improve the model performance by removing the less informative "noisy" data to avoid confusing the model. However, the large-scale part of the Sentence BERT and OpenAI API model suggests that the model has a large set of parameters and was pre-trained on large training data. This empowers the model to handle and mine the tiny bit of information from the "noisy" data without overfitting.

Chapter 5

Experiment 2: Implicit Content Network Graph Generation Efficiency Evaluation

In the previous experiment, a range of text embedding methods with corresponding text pre-processing pipelines optimised for news posts from social media platforms have been benchmarked for performance. To our knowledge, there are no labelled social media news benchmark datasets for information dispersal tracking tasks. Therefore, we have to take the performance measure of the major component of the system, the embedding method, as the indirect inference of the system performance. In this experiment, the aim is first to recreate the algorithm from the reference paper and then evaluate the efficiency of the realised model with proposed text embedding approaches following the same evaluation process in the reference paper by Joshi and Richard [8].

5.1 Model Realisation

This section is the first part of experiment 2 to realise and implement Python code to model the implicit content network for information propagation analysis proposed in the reference paper by Joshi and Richard [8] due to the unfortunate fact that reference papers have only described the approach without publishing any pseudocode or Python code. This effort is to validate the implementation-wise suitability of the embedding methods to build the network. The experiment also aimed to compare the model's efficiency with different embedding methods based on their training and running times. The literature review in section 2.3.3 discusses the detailed methodology of the reference

paper's algorithm that we are trying to realise. Our Python implementation of the network generation and visualisation have been shown below:

```
def generate_network(X, model_name, k=50, n=100):
    knn = NearestNeighbors(n_neighbors=k).fit(X)
    distances, indices = knn.kneighbors(X)
    clustering = AgglomerativeClustering(n_clusters=n).fit(X)
    G = nx.Graph()
    for idx, label in enumerate(clustering.labels_):
        G.add_node(idx, cluster=label)
    for idx, neighbors in enumerate(indices):
        for neighbor in neighbors:
            if idx != neighbor:
                G.add_edge(idx, neighbor)
    colors = [node[1]['cluster'] for node in G.nodes(data=True)]
    plt.figure(figsize=(10, 10))
    pos = nx.spring_layout(G)
    nx.draw(G, pos, with_labels=False, node_color=colors, cmap=plt
    .cm.rainbow)
    plt.savefig(f"images/twitter-{model_name}-{k}-{n_clusters}.png
    ")
```

Listing 5.1: Code realisation for generate implicit content network graph

In the code, a list of Twitter news posts embedded in a dense vector matrix is inputted into the function along with two hyperparameters: k and n. The `kneighbors()` function in `NearestNeighbors` class from `sklearn.neighbors` package is used to train the k-Nearest Neighbour (kNN) model for locating the semantic neighbours of each data point representing a Twitter news post. The `AgglomerativeClustering` class from `sklearn.cluster` package is used to train the HAC model for clustering posts into news event clusters. Finally, the graph is visualised through `networkx` and `matplotlib.pyplot` packages. Each node represents a social media news post and is coloured based on its general topic found in the HAC cluster. Edges are added between nodes that are nearest neighbours from the KNN model. The hyperparameter k is similar to the basic reproduction ratio (R0) in epidemiology mentioned in section 2.3.1. In the code, it controls the number of neighbours in the KNN model, which is reflected in the density of edges in the graph. It can also be interpreted as the expected dispersal ability of the posts and needs to be identified based on specific use cases. The hyperparameter n controls the number of clusters in the HAC model, which is reflected in the number of colours in the graph. It is interpreted as the expected number of news events within the data set that also needs to be predefined based on the prior analysis of the data set. Two unsupervised clustering models, KNN and HAC, are included in the system with different clustering

hyperparameters to empower the system to capture the news dispersal of minor aspects of news within the big news event. For example, tracking the dispersal and development of news, "Israeli Commander Nimrod Aloni has been captured by Hamas" in the big news event of "Israel Hamas War" originally sent by Mario Nawfal [9] on Twitter shown in Figure 5.1.

FIGURE 5.1: Screenshot of News of "Israeli Commander Nimrod Aloni has been captured by Hamas" on Twitter sent by Mario Nawfal [9]

In this experiment, we applied the model to visualise the news dispersal pattern on the combined Twitter 15 and Twitter 16 datasets with 2308 entries in total. The hyperparameter n is set to 100, and k is set to 50. The two number was found in the prior experiment as a reasonable pair for the data set. Each graph is generated via different embedding methods with corresponding optimal text pre-processing pipeline found in the previous experiment shown in Figure 4.3. We test the compatibility of all the embedding methods with the network model despite their PC performance to have an overall idea of the efficiency difference between methods. The output of graphs from each embedding method has been listed below in Figure 5.2.

In Figure 5.2, the embedding methods in the first row from left to right are TF-IDF, Doc2Vec, and BERTweet, respectively, and in the second row are Sentence BERT and

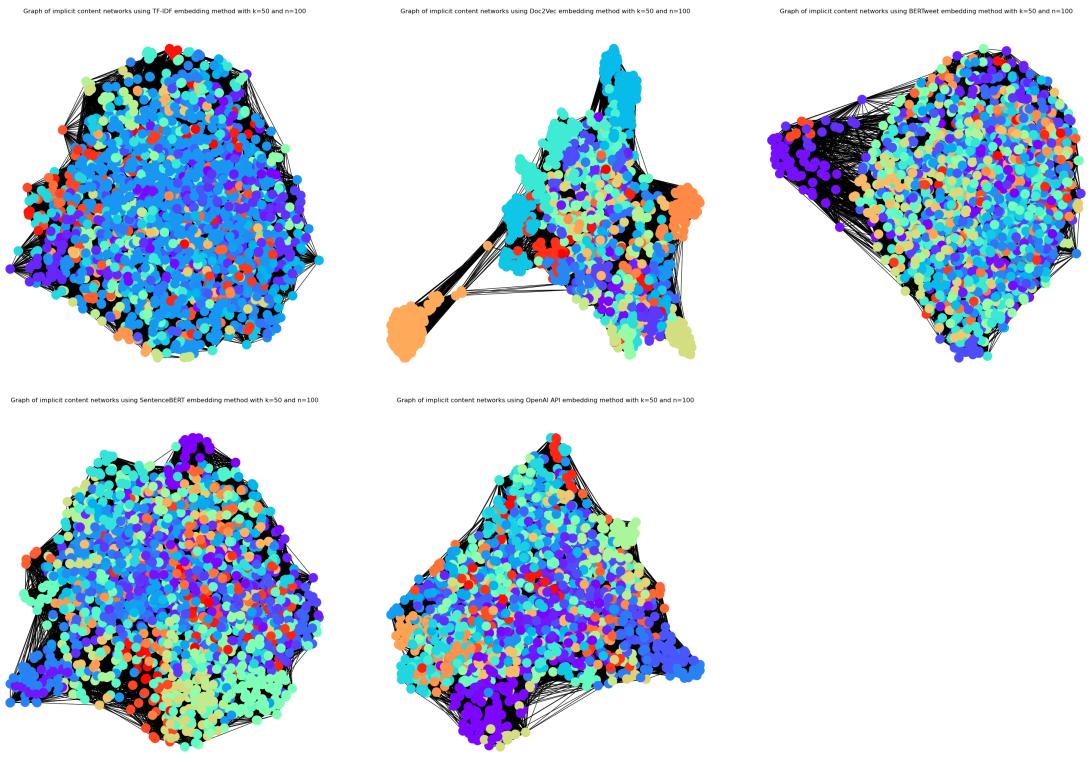

FIGURE 5.2: The graph of implicit networks using different text embedding methods with Twitter 15 and Twitter 16 dataset and hyperparameter n=100,k=50.

OpenAI API. As for implementation, only the dense vector outputted from the TF-IDF function must be converted to the array before feeding to the HAC model training method. With this minor fix, all the embedding models have demonstrated compatibility with the cluster, tracking, and visualising implementations. However, due to the space limitations of this paper, visualising the network containing 2308 notes would be too congested. To better aid the purpose of discussing the findings, the experiment on only the first 100 posts from the data set with hyperparameter n = 10 and k = 5 is conducted with the result shown in Figure 5.3 with the positioning of embedding methods.

The graphs represent content networks, where nodes represent individual pieces of Twitter news posts and edges represent dispersal relationships based on the embedding similarities between them. Different colours signify different clusters of topics within the news event. These clusters represent different narratives or facets of a particular news event. Different embedding methods (TF-IDF, Doc2Vec, BERTweet, SentenceBERT, OpenAI) have been employed to create these networks. The structure and clustering of the networks vary significantly across the embedding methods. This showcases the differences in the embedding methods' ability to capture content similarity and relationships. The TF-IDF embedding method provides a well-connected graph. However, the clustering seems more uniform, suggesting that it might be capturing more general or widespread similarities in the content. As for the Doc2Vec embedding method, the

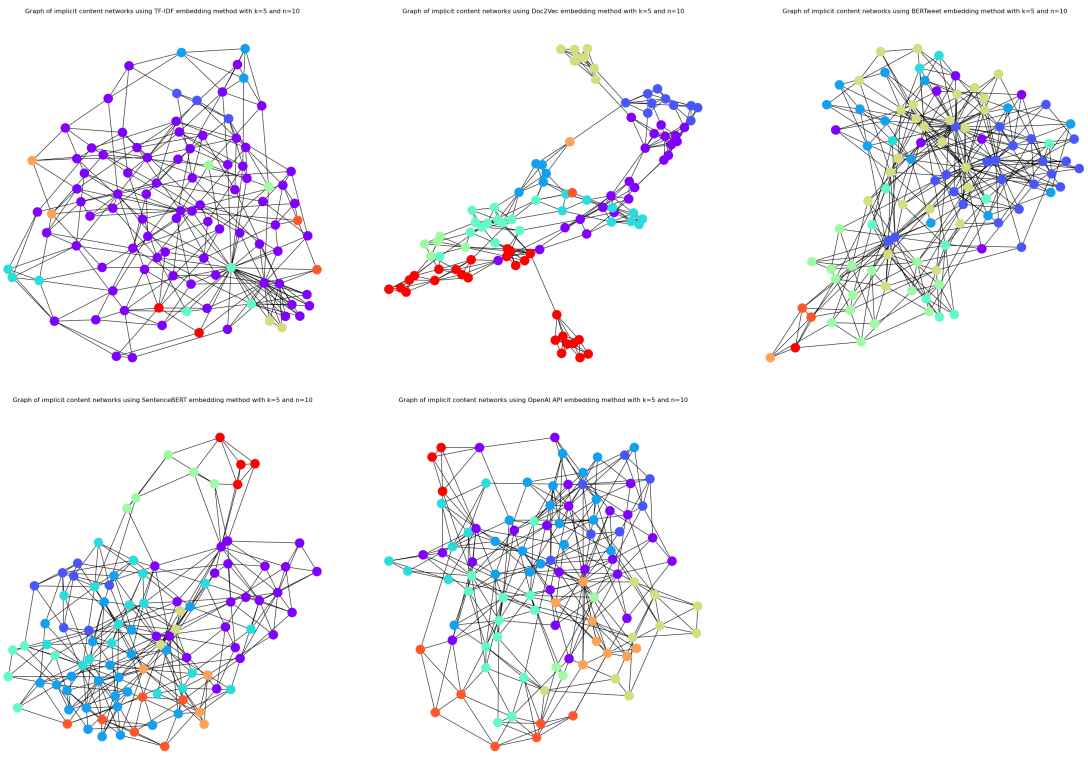

FIGURE 5.3: The graph of implicit networks using different text embedding methods with first 100 posts Twitter 15 and Twitter 16 dataset and hyperparameter n=10,k=5.

graph showcases more pronounced clustering compared to TF-IDF. There are more distinct colour-coded groups, suggesting that Doc2Vec may capture specific thematic or topic-based similarities more effectively. The graph generated by BERTweet embedding shows several distinct clusters with fewer inter-cluster connections. Since BERTweet is tuned explicitly for Twitter content, it might capture nuances and contexts unique to social media content, leading to more precise clusters. The SentenceBERT embedding network is well-connected but also has discernible clusters. This might indicate a balance between capturing general content similarities and specific topic-based nuances. Lastly, the OpenAI embedding method produces a well-connected graph similar to the TF-IDF approach but with slightly more defined clusters. It might be a more advanced embedding capturing both generalized and specific content similarities.

In conclusion, the above subjective interpretation of the network graphs shows that different embedding methods capture varying levels of content relationships and similarities reflected from previous experiments, resulting in different network structures and clusters. Depending on the usage goal being broad or specific topic tracking, one might prefer one embedding method over the others. The choice of embedding can significantly impact the analysis results, so careful consideration of qualitative evaluation of performance and efficiency is essential when analyzing news events and information propagation.

5.2 Efficiency Evaluation

With the algorithm explained, and the interpretation of the result discussed. This experiment will be conducted to benchmark the efficiency of each text embedding approach integrated with the optimised text pre-processing pipeline obtained from the previous experiment. The experiment was conducted on the Mac platform mentioned in Table 3.1 with training time, if applicable and running time recorded. The results have been listed below in Table 5.1 and visualised in Figure 5.4.

Embedding	Train Time (s)	Run Time(s)
TF-IDF	1.42	14.66
Doc2Vec	42.39	20.60
BERTweet	N/A	57.57
SBERT	N/A	14.24
OpenAI	620.62	11.22

TABLE 5.1: Train and run times in seconds for different embedding methods with optimal text pre-processing pipeline.

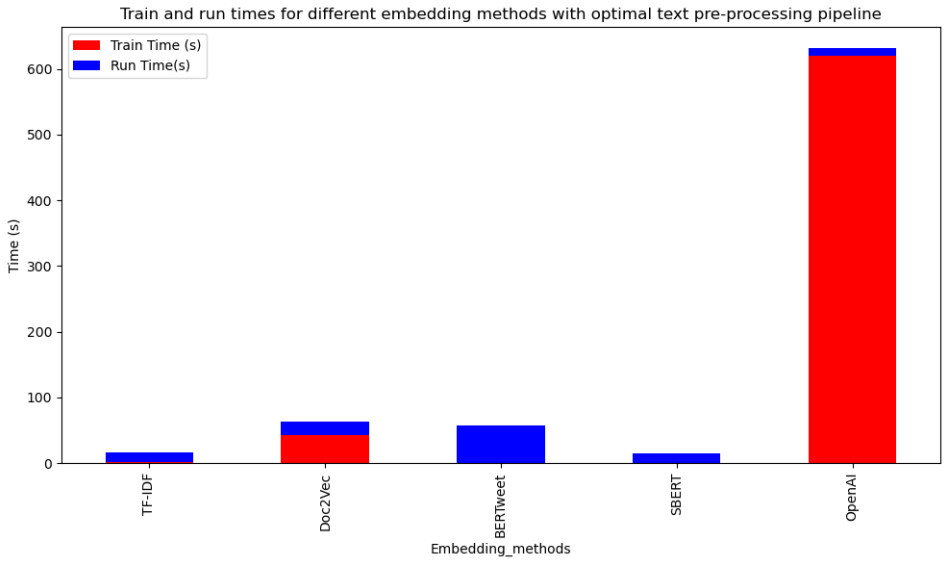

FIGURE 5.4: Graph of Train and run times for different embedding methods with optimal text pre-processing pipeline.

The training time for OpenAI API doesn't involve the time taken by actual pre-training of the model. Instead, it represents the time taken to request and obtain embeddings via the API calls without any local caching. This time also accounts for waiting periods to ensure that the rate limits of the API aren't exceeded. The run time denotes the efficiency post caching, meaning after all the required embeddings have been fetched and stored locally.

Based on the result from the table, the embedding process takes up the majority of the run-time, leaving the clustering and visualising process time negligible. TF-IDF is quick

to train but takes a moderate amount of time to run. Doc2Vec requires considerably more time to train compared to TF-IDF. Its run time is also greater than TF-IDF. BERTweet, despite having no training time in this context, has the highest run time amongst all the methods listed due to backtracking the parameters in the last hidden layer and averaging them. SBERT has the most competitive run time, slightly faster than TF-IDF and significantly faster than BERTweet. The OpenAI API takes the longest time for training or fetching embeddings, as clarified before. However, once the embeddings are cached, it offers the fastest run time.

In conclusion, when considering both training and running times, the OpenAI API method seems to be the most time-intensive for initial setup to fetching embeddings but offers the quickest execution thereafter. BERTweet, while not requiring training in this experiment, takes a significantly long time to run, which may be a consideration for real-time or rapid applications. TF-IDF and SBERT offer a balance between setup and execution times, making them viable options for various applications. The choice of best embedding method considering the efficiency as the compensation of their performance evaluation conducted in the previous experiment shown in Figure 4.3 is Sentence BERT. Sentence BERT appears to be the most suitable embedding technique with balanced good performance and efficiency. Although OpenAI API is incomparable in performance, the fact that it's a slow, online-based, paid service makes it incapable of scaling up for an actual application.

Chapter 6

Experiment 3: Parallelism Evaluation

The Sentence BERT embedding method with hashtag expansion text pre-processing technique has been concluded as the most suitable approach for modelling implicit content networks to track news dispersal on social media platforms in experiment 2. The past literature has all been concluded at this stage, but for this thesis, we aim to investigate further if the approach is ready for real-world use cases. As the computation hardware development is reaching the limit of physics and the end of Moore's law, suggested by Theis and Wong [96], vertical computational scaling to further improve single-core and single-machine performance has become physically and economically impractical. Therefore, the world has to take the alternative approach of horizontal computational scaling of adding more cores to processors and connecting distributed clusters for computers to form into the cloud. This choice has been reflected in the recent trend of cloud computing platforms such as AWS introduced earlier. To ensure that our proposed model is the capital of parallel computing to run on the cloud, this experiment is designed to test the running time difference between running on one core and four cores on different platforms to benchmark the parallel computing capability of the proposed model in different hardware and software environment.

6.1 Single-Core vs. Quad-Core Process Time

This experiment uses the exact implementation for generating an implicit content network with Sentence BERT and the same Twitter 15 and Twitter 16 combined data set. An important action taken to ensure the validity of the experiment by freeing it from the potential effect of the implicit parallelisation by the `sentence_transformers` package

performing text embedding is to set the system environment `TOKENIZERS_PARALLELISM` to `false` to disable the unintended parallelism. The Python class `ProcessPoolExecutor` from the `concurrent.futures` package with the parameter `max_workers` set to 4 is used to parallelise the program running with four cores. As such implementation runs on Windows and Mac, it does not work on Linux initially. It was found that on Linux, the default start method for multiprocessing is 'fork', while on Windows, it's 'spawn'. The 'fork' method caused issues with libraries not designed to be used post-fork. Therefore, the code only works on Linux by forcing the start method to 'spawn'. Also, to ensure the code only runs when the script is executed directly and not when it's imported as a module, the main execution guard of `if __name__ == '__main__'` is introduced to follow the common practice of using multiprocessing in Python. Running time recorded from the experiment with discussed implementation are shown below in Table 6.1 and visualised in Figure 6.1.

Platform	#Cores	Attempt 0	Attempt 1	Attempt 2	Attempt 3
MAC	One	18.22	17.53	17.33	17.35
	Four	6.31	6.32	6.28	6.23
WIN10	One	52.75	32.93	33.54	32.93
	Four	14.00	13.91	13.91	13.69
WIN11	One	40.06	39.67	39.83	38.30
	Four	18.48	16.28	17.61	16.23
Linux	One	40.39	46.37	44.89	40.65
	Four	54.73	49.50	45.89	46.04

TABLE 6.1: Execution Time of proposed model across different platforms and cores in seconds.

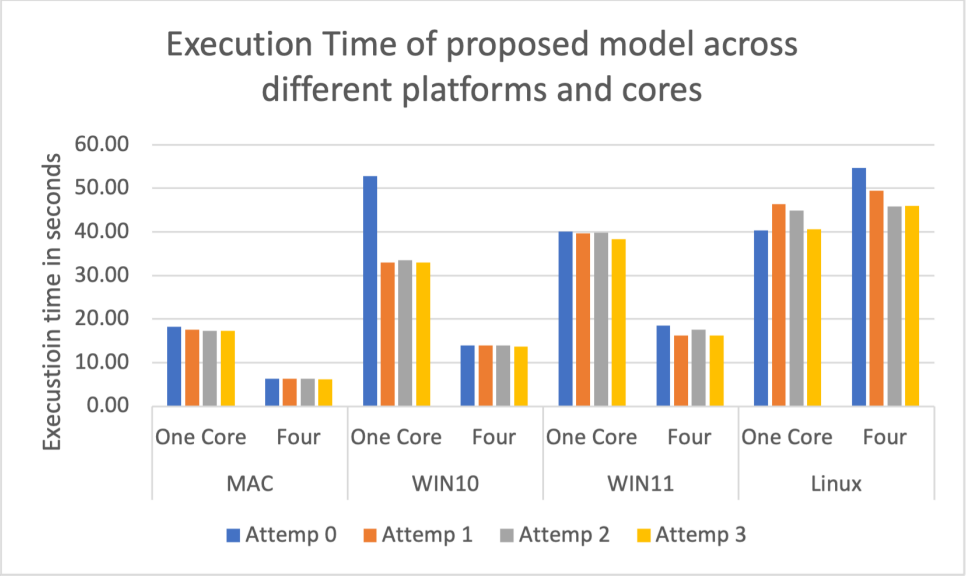

FIGURE 6.1: Graph of the execution time of the proposed model across different platforms and cores.

It was noticed that the first run on each platform tends to be slower than the following runs. This is suspected due to the initial loading of data from the hard drive to memory and configuring the Python environment. Therefore, to eliminate distractions, the result of the first run is discarded, and the analysis is based only on attempts one to three. For most platforms, utilizing four cores significantly speeds up the computation compared to using a single core. This validates the idea of horizontal scaling for improving performance. However, an anomaly on the Linux platform is spotted despite the extra effort put in for the model to run on. The increased runtime on Linux when using four cores is unusual and suggests there might be other factors or configurations affecting performance on this platform. After a careful investigation, it was found that the deadlock between thread situations has happened constantly, making multi-core situations worse. Therefore, at the current moment, the model is not ready to be put in a production environment with a Linux system.

6.2 Quantitative Evaluation

Although comparison of execution time can provide insight into how much the model can benefit from parallel computing. To standardised the evaluation measurement for the reader to compare our model with others quantitatively, we proposed to calculate the α, the fraction of running time spent on non-parallel parts of a computation, through both Amdahl's Law [20] and Gustafson's Law [21]. The two laws are foundational principles in the field of parallel computing and offer insight into the potential speedup one can achieve by parallelising a computational task across multiple processors, as stated in the journal paper by Sun and Ni [97].

Amdahl's Law was named after Gene Amdahl [20]. This law is used to determine the maximum potential speedup for a task when only a portion of that task can be parallelised. The law can be expressed as the formula shown below in Equation 6.1.

$$S = \frac{1}{\alpha + \frac{(1-\alpha)}{n}} \tag{6.1}$$

Wheres in the equation, the variables represent:

- S is the potential speedup of the parallelised operation.
- α represents the proportion of the execution time of the whole task that is serial or non-parallelisable. This is the portion of the program that must execute serially and cannot benefit from multiple processors

- n represents the number of processors.

The law overly simplifies the real world and assumes a fixed problem size, suggesting no overhead for parallelisation. It brought the insight that no matter how many processors we have, if a significant portion of the task (α) is sequential, the speedup is limited. As

$$n$$

tends to infinity, the maximum speedup will approach $\frac{1}{\alpha}$.

On the other hand, John Gustafson [21] introduced this law to counter a pessimistic view of Amdahl's Law. While Amdahl's Law keeps the problem size constant, Gustafson's Law suggests that as we add more processors, we will likely tackle more extensive problems as parallelisation overhead is involved. Hence, the parallelisable portion of the task can increase with the problem size. The formula is shown below in Equation 6.2 with the same variables from the previous equation except α now the proportion of the execution time of the whole task on a single processor that is serial. To further explain this, α refers to the serial portion of the computation in both laws. However, the interpretation and emphasis differ between the two laws: Amdahl focuses on the limitations imposed by the serial portion, while Gustafson focuses on the scalability of the parallel portion as the problem size increases.

$$S = \alpha + n(1 - \alpha) \tag{6.2}$$

The formula suggests that as we add more processors and aim to solve more significant problems, the parallel portion can dominate, leading to near-linear speedup. The sequential fraction becomes less significant as n increases.

In comparison, Amdahl's Law emphasizes the limitation on speedup due to the non-parallelisable portion of a task. It can be seen as a more conservative or pessimistic view of parallelism. On the other hand, Gustafson's Law counters the pessimism by suggesting that, in reality, as computational resources grow, we'll attempt larger problems, thus benefiting more from parallelism. It presents a more optimistic view.

Both laws serve as critical tools when evaluating and designing parallel algorithms and systems, helping engineers and researchers understand the potential gains and limitations of parallelisation. Therefore, we intend to compute the α of our model to help readers make comparisons. First, the speedup variable, S, can be calculated as the ratio of sequential execution time to parallel execution time. The number of processors, n, in

this experiment, is 4. Then, we rewrite both laws to solve for α with Equation 6.3 for Amdahl's Law and Equation 6.4 for Gustafson's Law.

$$\alpha = \frac{n - S}{n(1 - S)} \tag{6.3}$$

$$\alpha = \frac{S - n}{n - 1} \tag{6.4}$$

The results of the computation have been shown below in Table 6.2, and the comparison between computed α is visualised in Figure 6.2.

Platform	**Average Sequential Execution Time**	**Average Parallel Execution Time**	S	α_A	α_G
MAC	17.40	6.28	2.7727	0.6045	0.8058
WIN10	33.13	13.84	2.3946	0.4177	0.5911
WIN11	39.27	16.71	2.3504	0.3501	0.5174
Linux	43.97	47.14	0.9327	0.0749	-0.9040

TABLE 6.2: The average execution time for sequential and parallel, speedup, and fraction of sequential process computed from Amdahl's Law and Gustafson's Law.

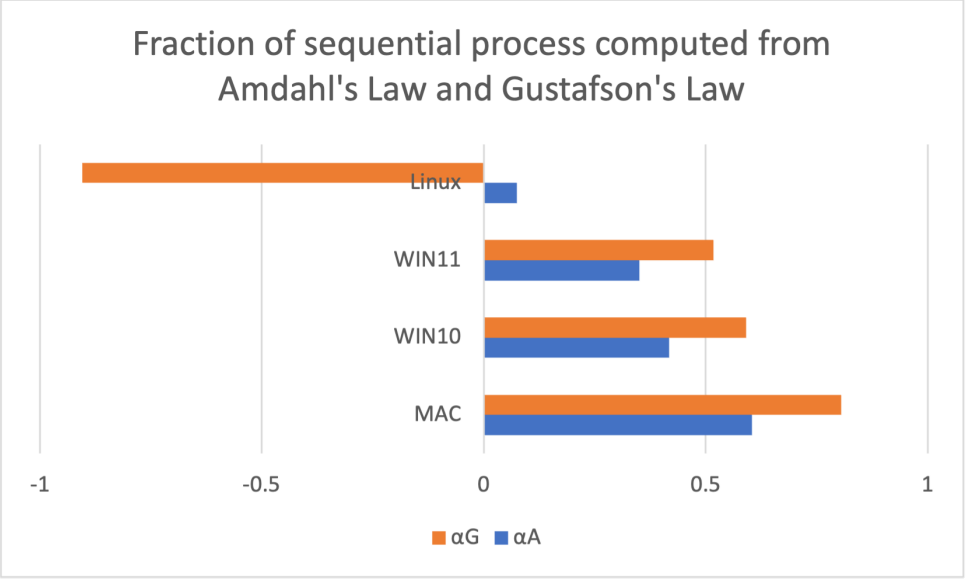

FIGURE 6.2: Graph of fraction of sequential process computed from Amdahl's Law and Gustafson's Law.

The parallel execution on MAC yields a substantial speedup ($S = 2.7727$), indicating the efficiency of parallel processing. Only about 60.45% or 80.58% of the computation is inherently sequential according to Amdahl's Law and Gustafson's Law, respectively, which means a significant portion can be parallelised. Higher α suggests more computations have been parallelised and take less time than running in series. Parallel execution

on WIN10 offers a better speedup ($S = 2.3946$). The α_A value of 0.4177 indicates that less than half of the computation is inherently sequential, offering room for parallelism. The speedup on WIN11 ($S = 2.3501$) is comparable to that on WIN10. With an α_A value of 0.3501, WIN11 also has a good potential for parallelism.

Lastly, the anomaly Linux can be interpreted differently based on Amdahl's or Gustafson's Law. Based on Amdahl's Law, the Linux $\alpha_A = 0.0749$ indicates that a small fraction of the program is sequential, which is somewhat unexpected given the speedup is less than 1 (0.9327). This might be due to various other factors, such as overheads introduced by parallelism or the specific configuration of the Linux system. On the other hand, according to Gustafson's Law, the negative value of α_G = -0.904 for Linux is atypical. Negative values suggest that the parallel portion takes more time in the four-core setting than the single-core. It can be a result of overheads or inefficiencies in the parallel execution.

In summary, parallelisation offers significant performance benefits on the MAC, WIN10, and WIN11 platforms. On the Linux platform, the benefits are not evident from the given data. Still, there's potential for improvement, especially if the problem is larger or the code is optimised parallelisation for the Linux platform. The data demonstrates the importance of considering both hardware and software aspects when evaluating parallel performance and suggesting the model scales for parallel best on the Mac OS platform.

Chapter 7

Discussion

After archiving the research goal of improving the implicit content networks model for social media news propagation analysis with large-scale pre-trained language models in three experiments demonstrated in the previous chapters, this chapter aims to discuss the potential applications of the model.

7.1 Potential Application

The output of our proposed model, the implicit content network for tracking news dispersal through social media platforms, was visualised at the end of the experiment. This demonstrated the network has been used standalone to provide users and researchers with a more comprehensive understanding of how news is propagated through social media posts. However, this is not the full potential of the network. It can also serve as an extra informative feature to aid other more popular NLP tasks that improve performance or efficiency. In this section, the potential applications of the network to be discussed are rumour detection and knowledge-based news event analysis and forecasting.

7.1.1 Rumour Detection

Although rumour detection has been criticised in the introduction for its bias, it is still unavoidable to be a much more popular research field with more significant market potential for providing users with short and easy-to-understand binary answers to trust the post or not. We brought it up again because the outputted network of our proposed model has the potential to help in identifying anomalies or irregular patterns in news dispersal, which could be indicative of fake news or misinformation campaigns. Because of the limited time of the study, we did not have the opportunity to conduct detailed

experiments. Still, we recorded our reasoning and ideas of potential integration between implicit content networks and rumour detection techniques. Before diving deep into the integration, it is essential to first overview the development of the rumour detection research.

In the NLP field, social media rumour detection is an important research topic to identify social media posts containing misinformation to avoid negative influence. The rumour detection field was pioneered by Castillo et al. [98] in 2011, who proposed an automatic method for assessing the credibility of a given set of tweets. The research was conducted with data from Twitter on newsworthy topics that were getting a spike in popularity in a short time detected by Twitter Monitor [99]. J48 decision tree machine learning model was selected, shown in Figure 7.1, with the output of "A" for truth and "B" for the rumour. The input features were from three different aspects: user, topic, and propagation. In contrast, the message-based feature is ineffective for the task. Benefiting from the excellent interpretability of the decision tree machine learning model, we learnt that the core idea was to identify if the tweet was sent from a user in connection with a network of credible users that are more likely to produce trustworthy tweet posts. The fine-tuned model claimed to achieve an astonishing 0.787 F1 score, considering it was constituted by a non-neural network model completed in the early stage of the research field. We observed that the paper has suggested the URLs in tweets were traded as an important feature under the propagation aspect feature set. This has coincided with the idea of URL-based information propagation tracking that was reviewed in section 2.3.1. This coincidence indirectly presents the commonalities between the two studies and boosts our confidence to improve rumour detection performance with the implicit content network.

After Castillo et al.'s pioneered paper, abundant research has been conducted to improve the performance of the automatic rumour classifier by tweaking different parts of the model's pipeline or specialised the model to perform in a specific setting.

One such research conducted by Sicilia et al. [11] focuses specifically on the social media post data from Twitter in the health domain for rumour detection. The main innovation point of the paper is that Sicilia et al. proposed a novel feature design method to improve accuracy across various classification machine learning models. Because of the limitation of social media post data from only a single domain, the topic-based features mentioned in the previous study were unavailable; therefore, Sicilia et al. have proposed a range of new features and evaluated them with other remaining available features across different machine learning classification models. As a result, the finding has been shown in Figure 7.2 cited from the original paper [11]. In the figure, the newly proposed influence potential-based features are marked with a dagger (†) and

FIGURE 7.1: Decision tree model built for the credibility classification by Castillo et al. [10]

graph theory-inspired features are marked with a diamond (♢). The histogram can be interpreted so that more informative features would have a shorter stacked histogram bar, and the red dotted vertical line stands for the median. For example, the most informative feature, the sentiment score, suggests that tweets with neutral tones tend to be more trustworthy. The paper concluded that the novel proposed features enable models to "detect about 90% of rumours, with acceptable levels of precision" [11]. From the result, we observed that features related to the information dispersal pattern tend to be more informative than those related to user information. URL information also played an essential role in the task, such as the feature of the likelihood of a URL being shared (Purl shown in Figure 7.2).

Besides improving the performance by tweaking the features, there have also been research experiments on rumour detection with more advanced machine learning models, especially in the current era where the rising neural network-based deep learning strategies have outperformed traditional machine learning models in the vast majority of aspects. The research conducted by Tian et al. [100] focused on the early detection of

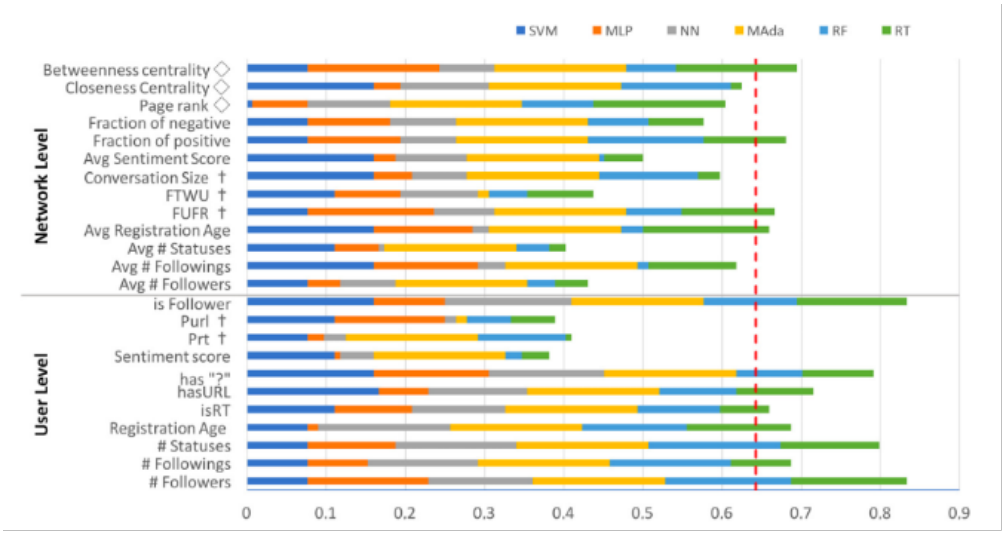

FIGURE 7.2: Stacked histogram of the rank analysis by Sicilia et al. [11].

rumours on Twitter. Rumour detection in the early stage is essential for the platform to contain the spread of misinformation promptly to reduce the negative influence as much as possible. However, at the early stage, before the information starts to spread, the available features are minimal. To tackle such a challenge, the early rumour detection model proposed by Tian et al. only relied on features from tweet contents and their immediate user comments. The model consists of two procedures. The first step is to learn the attitude representation for stance prediction from user comments, and Tian et al. proposed to use CNN and BERT-based deep neural models to solve such tasks via transfer learning. The follow-up step was to classify the rumours based on the attitude representation obtained from the previous step and the content representation of the tweets with their comments. For this step, the model proposed by Tian et al. was based on CNN-BiLSTM and BERT neural models. The experiment was conducted on two public Twitter datasets inspired by Ma et al. [101], namely, Twitter15 [90], and Twitter 16 [91] that we have used for efficiency measurement of our model. Tian et al. claimed that their proposed BERT-based model had consistently outperformed state-of-the-art baselines significantly. The research demonstrated that the content of user comments is informative and contains early signals for rumour detection which could benefit from the network of semantically similar posts to provide extra information when there is no other information available at an early stage.

The last study to be discussed was conducted by P et al. [102] on automatically detecting rumoured tweets and finding their origin. The research was conducted on the data set containing Twitter posts relating to the 2011 London Riots with the fact check done by The Guardian. Although P et al. have used the same J48 decision tree machine learning model as Castillo et al. [98], P et al. have innovatively conducted a feature reduction based on the experiment results to improve the classification accuracy. It was found that

user-propagated content-based features are considerably more important than the user's identity-based features. This was explained by the observation illustrated by Gupta et al. [103] that the genuine user has an equal chance of retweeting and propagating tweets containing misinformation as the rogue users intend to spread rumours during the crisis situation. To better understand the potential of user-based features, P et al. proposed an algorithm for locating the origin rumour posting user for user-based features analysis to help the situation with limited tweet-based information, similar to the early stage rumour detection setting in Tian et al.'s research [100]. We believe our semantic similarity-based content network will provide the model with more user-propagated content-based information that was not traditionally found by only tracking repost actions.

7.1.2 Knowledge-Based News Event Analysis And Forecasting

Apart from feeding the implicit content network into a machine learning model for rumour detection, it can also be used to analyse and forecast news events. The news event analysis and forecasting toolkit based on the knowledge graph proposed by Hassanzadeh et al. would be helpful for such task [104]. The knowledge base was a class of strategies for storing information, knowledge, and truth in the computer system, which knowledge graph (KG) has been the most widely accepted intense of knowledge bases in the question-answering field of NLP research. The KG is constructed by Resource Description Framework (RDF) triple in format ¡subject, predicate, object¿. The toolkit proposed by Hassanzadeh et al. has three main functions: extracting the news post into KG, using KG for causal analysis and forecasting, and supplementing KG with causal knowledge from external sources. The distinctive point of Hassanzadeh et al.'s research is that their toolkit enables building a human-in-the-loop explainable solution for downstream analysis, which has met our initial accessibility goal. However, the toolkit could only retrieve news headlines from Wikinews and EventRegistry [105] with no support for social media news post input from content networks.

To utilise the toolkit, the first task that needs to be completed is to extract the social media news post from the implicit content network to KG. Retrieving text into KG can be classified as a relation extraction task within the NLP field. To solve such a class of problems, Trisedya et al. [106] have proposed a novel neural-based end-to-end-relation extraction model for information extraction. Their approach has addressed the error propagation drawback caused by Named Entity Disambiguation (NED), which was heavily relied on for mapping RDF to knowledge base space, by replacing NED with a novel neural-based end-to-end model. The paper also improved the performance

from other aspects. An n-gram-based attention model was proposed to avoid breaking multi-word entity names from news posts apart during the extraction.

At this stage, the toolkit has sufficient support to be used with social media news post input for news event analysis and forecasting. However, an additional challenge is posed by the difference in language domain. Compared to news headlines, social media news posts contain less standardised language that could potentially refer to the same entity with multiple different names. By merging different names of the same entity, the performance of analysis and forecast news events from social media content networks using the toolkit can be significantly improved. The task of merging different representations of the same real-world entities is called Entity Alignment (EA). It is essential for creating more precise KG or connecting multiple KGs together to expand the knowledge base. To address the issue of merging similar but distinct entities using a traditional embedding similarity-based greedy search algorithm, a modified beam search along with a triple classification algorithm was proposed by Trisedyl et al. [106] to merge the same real-world entities with different names in KG. To further improve the performance, Trisedyl et al. [107] have published a separate paper on the EA problem that first generates attribute embeddings from the attribute triples as the seed alignment to shift the entity embedding of multiple KG into the same vector space. Then, the number of attributes of an entity is further enriched by the method transitively to better identify the similarity between different entity embeddings.

Chapter 8

Conclusions

In this thesis, the background of the research is explained that social media is replacing traditional paper of web-based news media as the primary source of news consumption. In response to harmful misinformation on social media, information dispersal analysis is studied. The past literature regarding text pre-processing, text semantic embedding, and information dispersal tracking was reviewed to form the basis of the research. In the first experiment, the research goal of utilising large-scale pre-trained language models to improve the performance and efficiency of the implicit content network model proposed by Joshi and Richard [8] specifically for the social media language domain has been accomplished. Our model captures the semantics of content text using Sentence BERT and OpenAI API, which are empowered by large-scale pre-trained language models BERT and GPT, resulting in a PC of 0.7960 and 0.8457 respectively, on the SemEval 2016 headlines dataset without any pre-processing techniques. Both approaches have outperformed all the models, which are based on Doc2Vec in the reference paper, and claimed to reach PC below 0.75 on the same benchmark dataset. It was also found in the experiment that a more refined modern embedding approach has integrated the required pre-processing stage, and overdoing the pre-processing can lead to performance loss. In the second experiment, the Sentence BERT has also been successfully integrated into the realised implementation of the implicit content network model. It has shown better efficiency than all models with other text embedding approaches for only taking 14.24 seconds in execution time, including Doc2Vec used by the reference model, which takes 20.6 seconds, excluding training time. Therefore, the implicit content network with Sentence BERT incorporated with hashtag expansion has been selected as the best model considering the balance of both performance and efficiency that have outperformed the reference model. In the third experiment, by experimenting with the proposed model in series and parallel on a range of hardware and software platforms, we conclude our current implementation of the model can parallelised well on Windows and Mac OS but

not quite on Linux. We also calculate the portion of the code that can be parallelised on each platform by Amdahl's and Gustafson's Law.

8.1 Contribution

To summarise the contribution of the research. We improve the state-of-the-art implicit content network model for tracking information dispersal in both performance and efficiency. We further optimised the model specifically on the English Twitter news post domain with a customised text pre-processing pipeline. Lastly, we introduced the novel measurement of parallel computing capability based on the parallelisable portion of the system through Amdahl's and Gustafson's Law and computed it for our best model under different physical and software platforms.

8.2 Future Improvement

The research was planned and proposed with ambition but faced the reality of research time limitations and other objective constraints. In this section, we aim to share the direction of subsequent efforts that can be done moving forward from the current research stage. There are two directions to further the study. One is the industrial direction to implement the cloud infrastructure for intra- and inter-social media platforms' news dispersal pattern analysis as a product. The other one is the academic direction, which optimises the model further to work with trending short video platforms.

8.2.1 Cloud Infrastructure

At the early planning stage of the research, the experiment was planned to not stop at evaluating the parallel computing capabilities of the proposed model but to implement further the cloud infrastructure for intra- and inter-social media platforms news dispersal pattern analysis.

8.2.1.1 ADO

Due to the fact such a proposed system would provide great aid to sociological research, a related system has been proposed, implemented and deployed. The Australian Data Observatory (ADO) [108], funded by the Australian Research Data Collection, was built to collect, store, and analyse social media posts, as proposed by Morandini [12].

The ADO system shared various similarities compared to the proposed cloud system, especially because it also visualised the propagation of social media posts, except ADO captured the relation of posts based on topic modelling using BERTopic [109] instead of semantic similar as the model proposed by us. The topic modelling approach would enable the model to better capture the tweet cluster on a higher level general topic while our approach performs better at capturing minor steam of information flowing among the general topic as shown in Figure 5.1. Apart from the difference in the core approach for tracking information dispersal, the ADO has many implementation and deployment design choices we can use for reference. For example, Morandini et al. [12] proposed a novel topographic visualization method to include the "popularity" or the importance of a specific node based on its influence in the final visualisation stage with an additional dimension as shown in Figure 8.1.

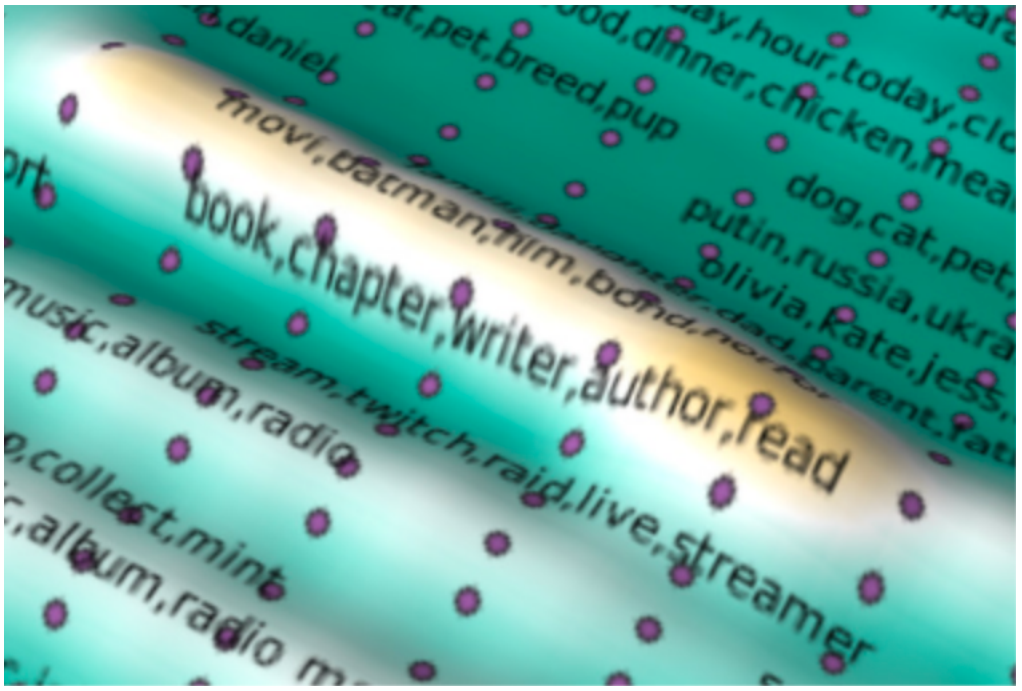

FIGURE 8.1: Topographic visualisation method on Batman Peak visualisation by Morandini et al. [12].

8.2.1.2 Real Time Data Streaming

The ADO is capable of steaming real-time social media posts using API and storing them in a database. We proposed to include the same capability for our cloud infrastructure as motivated by the assumption that the proposed model can be instrumental in real-time tracking of how news spreads across different social media platforms. By observing the patterns of information flow, users and researchers can get a clearer picture of which platforms act as primary sources and which ones serve as amplifiers. According to

the plan, the developer API for Twitter [110], Reddit, and Mastodon will be used to stream news-related posts continuously to the buffer that keeps posts for seven days to avoid data overflow. When trending news is detected, the system can also use API to search history posts with keywords for up to seven days due to the limitation of API. The collected data will be stored in the database system of choice, which in this research is proposed to be CouchDB [111]. Two main benefits brought by CouchDB are clustering for fault tolerance and MapReduce for efficient data pre-processing [112]. By running a CouchDB cluster across different nodes, the replication of the database will store the same document on different nodes to reduce the chance of data loss and improve fault tolerance. The in-build MapReduce query logic can be applied for content extraction from raw data in the pre-processing stage or statistics summarisation in an easy-to-manage and efficient way as the algorithm will be running in parallel across clustered CouchDB deployed on different nodes. However, the proposal ended up being impossible mainly due to the inability to stream data through their APIs. Since Elon Much took over Twitter, the rate limit for free-tier researcher API has been reduced to an unacceptable 1500 posts per month. At the same time, the paid enterprise API costs an unreasonable $42,000 per month [113]. A similar situation also happened on Reddit as the free API also started requiring payment, and our application for free researcher API has come to nothing today, half a year later. Fortunately, while Maston still has their free researcher API available, their limited number of users and content can not justify the scaling research goal. We suspect the tightening of API restrictions stems from concerns about the source of its training data after the rise of ChatGPT. The company does not want to provide free training data for it and the next chatGPT.

8.2.1.3 MRC

An intuition supporting the adoption of the cloud technique is that the proposed model is both computationally expensive and requires unprecedented amounts of data. We propose to use cloud technologies to address the above issue with the extra benefit of broad network access, rapid elasticity and other essential characteristics of the cloud proposed by Mell and Grance [114]. For that purpose, the Melbourne Research Cloud (MRC), hosting deployment of ADO, is a great fit be the cloud platform to deploy the model for social media news dispersal pattern analysis. The MRC was managed by the University of Melbourne, which grew out of the Nectar Research Cloud [115], Australia's national research cloud specifically designed for research computing [116]. Under the cloud taxonomy discussed in the Journal by Mell and Grance [114], the MRC is classified as an Infrastructure as a Service (IaaS) service model and a private cloud

deployment model. The MRC system is built based on the standardised open-stack framework, allowing access to different modules for extra functionality.

The first benefit of using MRC is its availability. As a cloud-based system, MRC instances are capable of running day and night continuously and thus enable us to run the social media post collection and analysis backstage continuously. Secondly, with allocated resources for this research so far from MRC illustrated in Table 8.1, sufficient computational resources have been provided to complete this research project, especially since sufficient RAM is essential for large-scale NLP models. Even though MRC provides great on-demand dynamic scalability, extra resources can always be applied and allocated in the event of a burst in the news. In the security aspect, MRC, as a private cloud, has implemented stringent security. The security group function avoids unauthorised access, and the snapshot function helps the system recover from a fatal crash.

Resource	Quality
Compute Resources - Virtual Cores	32 VCPUs
Compute Resources - Instances	6 servers
Compute Resources - RAM	Unlimited
Volume Storage	2000 GiB
Advanced Networking - Networks	3 Networks
Advanced Networking - Routers	3 Routers
Advanced Networking - Floating Ips	2 Floating IPs
Advanced Networking - Load Balancers	2 Load Balancers

TABLE 8.1: MRC resource allocation for this research

System design-wise, benefiting from the fact the MRC is classified as an IaaS delivery model according to the NIST definition of cloud computing written by Mell and Grance [114], there is sufficient room to play in system design. In the IaaS model, the cloud service providers manage the networking, storage, servers, and virtualisation to offer users a cloud infrastructure to deploy and run arbitrary software, including cooperation systems and applications. The user is responsible for managing and controlling the operating system, middleware, runtime, data, and application. IaaS is optimal for conducting this research as we have absolute control and freedom over the software side while the hardware has been encapsulated. Therefore, we have designed a system architecture specifically for this research to utilise the full potential of the cloud computational power while being robust. The system's fundamental design is based on the master-worker architecture with a worker pool design to manage multiple instances of a distributed and parallel computing system. The master is opened to the public internet to receive news topics of interest for analysis from researchers or users through a ReSTful HTTP request from the researcher or user. The ReSTful design of the master

node enables standardised access to input requests and output results for third parties to develop an open-source frontend quickly. After receiving the request, the master will decompose the task into smaller subtasks and add them to the task queue stored in the database system. The worker instance periodically checks the task queue and take task when available. This design can bring two significant benefits: scalability and fault tolerance. Each worker is running identical code capable of processing any task, enabling the system to scale horizontally by adding or taking additional worker nodes with no extra configuration needed while the system runs and functions normally. Fault tolerance-wise, by adding the time-out logic that when work fails to return the result of a particular task due to a potential node crash or network outage, the master node will add the assigned task back to the top of the task queue.

8.2.2 Short Video Platforms

As shown in the introduction, social media have taken over the market share of traditional blog platforms for news consumption faster than people realise. This taking over is not ending at the short text social media platforms such as Twitter but will keep evolving. In fact, short video platforms such as TikTok are taking over the internet with inexorable momentum, similar to how traditional internet-based news media was replaced by social media. It was feasible to apply speech-to-text and image-to-text algorithms to feed the outputted text as the input to the unsupervised learning model discussed previously in this paper. However, unfortunately, at the time of this paper being written, the leading company of the short video industry, TikTok, confirmed by its CEO, Shou Chew, that it has 150 million monthly active users in the United States, which is making up nearly half of its countries population, has not been opening their researcher API access to a researcher outside of United States [117]. Future follow-up research on including TikTok short video to the news dispersal pattern analysis will be conducted once the TikTok developer API is available globally in the near future with technologies such as Whisper API from OpenAI to convert the video into text description to feed in our proposed model.

Appendix A

Text Preprocessing Exhaustive Search Experiment Full Result

This is the Appendix for the full experiment result for the text preprocessing section in the first experiment, which is discussed in section 4.2.2. The tables are the result of the text embedding method in the order of: "TF-IDF", "Doc2Vec", "BERTweet", "Sentence BERT", and "OpenAI". For each row, `"O"` indicates the preprocessing step of the column is applied, whereas `"X"` is the opposite. The reference abbreviation name for each text preprocessing strategy has been discussed in detail in section 4.2.2, and the shorter version has been listed below:

- **EMO**: Emoji and emoticons to text
- **HAS**: Hashtag expansion
- **PUN**: Punctuation removal
- **LOW**: Lowercasing text
- **LEM**: Lemmatisation text
- **STE**: Stemming text
- **STO**: Stopword removal

Table A.1: TF-IDF Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	X	X	X	X	X	X	0.6240
O	X	X	X	X	X	X	0.6239

Table A.1: TF-IDF Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	O	X	X	X	X	X	0.6132
X	X	O	X	X	X	X	0.6037
X	X	X	O	X	X	X	0.6240
X	X	X	X	O	X	X	0.5985
X	X	X	X	X	O	X	0.6483
X	X	X	X	X	X	O	0.6240
O	O	X	X	X	X	X	0.6130
O	X	O	X	X	X	X	0.6026
O	X	X	O	X	X	X	0.6239
O	X	X	X	O	X	X	0.5986
O	X	X	X	X	O	X	0.6481
O	X	X	X	X	X	O	0.6238
X	O	O	X	X	X	X	0.5964
X	O	X	O	X	X	X	0.6132
X	O	X	X	O	X	X	0.5831
X	O	X	X	X	O	X	0.6379
X	O	X	X	X	X	O	0.6131
X	X	O	O	X	X	X	0.6037
X	X	O	X	O	X	X	0.5786
X	X	O	X	X	O	X	0.6449
X	X	O	X	X	X	O	0.6038
X	X	X	O	O	X	X	0.6420
X	X	X	O	X	O	X	0.6484
X	X	X	O	X	X	O	0.6232
X	X	X	X	O	O	X	0.6684
X	X	X	X	O	X	O	0.5970
X	X	X	X	X	O	O	0.6468
O	O	O	X	X	X	X	0.5952
O	O	X	O	X	X	X	0.6130
O	O	X	X	O	X	X	0.5833
O	O	X	X	X	O	X	0.6378
O	O	X	X	X	X	O	0.6129
O	X	O	O	X	X	X	0.6026
O	X	O	X	O	X	X	0.5788
O	X	O	X	X	O	X	0.6448
O	X	O	X	X	X	O	0.6027

Table A.1: TF-IDF Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
O	X	X	O	O	X	X	0.6421
O	X	X	O	X	O	X	0.6482
O	X	X	O	X	X	O	0.6230
O	X	X	X	O	O	X	0.6683
O	X	X	X	O	X	O	0.5972
O	X	X	X	X	O	O	0.6467
X	O	O	O	X	X	X	0.5964
X	O	O	X	O	X	X	0.5666
X	O	O	X	X	O	X	0.6380
X	O	O	X	X	X	O	0.5962
X	O	X	O	O	X	X	0.6277
X	O	X	O	X	O	X	0.6380
X	O	X	O	X	X	O	0.6122
X	O	X	X	O	O	X	0.6582
X	O	X	X	O	X	O	0.5812
X	O	X	X	X	O	O	0.6365
X	X	O	O	O	X	X	0.6170
X	X	O	O	X	O	X	0.6450
X	X	O	O	X	X	O	0.6038
X	X	O	X	O	O	X	0.6456
X	X	O	X	O	X	O	0.5790
X	X	O	X	X	O	O	0.6445
X	X	X	O	O	O	X	0.6638
X	X	X	O	O	X	O	0.6399
X	X	X	O	X	O	O	0.6469
X	X	X	X	O	O	O	0.6662
O	O	O	O	X	X	X	0.5952
O	O	O	X	O	X	X	0.5668
O	O	O	X	X	O	X	0.6380
O	O	O	X	X	X	O	0.5950
O	O	X	O	O	X	X	0.6278
O	O	X	O	X	O	X	0.6378
O	O	X	O	X	X	O	0.6120
O	O	X	X	O	O	X	0.6582
O	O	X	X	O	X	O	0.5814
O	O	X	X	X	O	O	0.6363

Table A.1: TF-IDF Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
O	X	O	O	O	X	X	0.6172
O	X	O	O	X	O	X	0.6449
O	X	O	O	X	X	O	0.6027
O	X	O	X	O	O	X	0.6456
O	X	O	X	O	X	O	0.5792
O	X	O	X	X	O	O	0.6444
O	X	X	O	O	O	X	0.6638
O	X	X	O	O	X	O	0.6400
O	X	X	O	X	O	O	0.6467
O	X	X	X	O	O	O	0.6662
X	O	O	O	O	X	X	0.6082
X	O	O	O	X	O	X	0.6381
X	O	O	O	X	X	O	0.5962
X	O	O	X	O	O	X	0.6394
X	O	O	X	O	X	O	0.5668
X	O	O	X	X	O	O	0.6377
X	O	X	O	O	O	X	0.6537
X	O	X	O	O	X	O	0.6253
X	O	X	O	X	O	O	0.6365
X	O	X	X	O	O	O	0.6560
X	X	O	O	O	O	X	0.6430
X	X	O	O	O	X	O	0.6175
X	X	O	O	X	O	O	0.6446
X	X	O	X	O	O	O	0.6459
X	X	X	O	O	O	O	0.6616
O	O	O	O	O	X	X	0.6084
O	O	O	O	X	O	X	0.6380
O	O	O	O	X	X	O	0.5950
O	O	O	X	O	O	X	0.6394
O	O	O	X	O	X	O	0.5670
O	O	O	X	X	O	O	0.6376
O	O	X	O	O	O	X	0.6536
O	O	X	O	O	X	O	0.6254
O	O	X	O	X	O	O	0.6363
O	O	X	X	O	O	O	0.6560
O	X	O	O	O	O	X	0.6431

Table A.1: TF-IDF Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
O	X	O	O	O	X	O	0.6177
O	X	O	O	X	O	O	0.6445
O	X	O	X	O	O	O	0.6459
O	X	X	O	O	O	O	0.6616
X	O	O	O	O	O	X	0.6369
X	O	O	O	O	X	O	0.6085
X	O	O	O	X	O	O	0.6377
X	O	O	X	O	O	O	0.6397
X	O	X	O	O	O	O	0.6514
X	X	O	O	O	O	O	0.6433
O	O	O	O	O	O	X	0.6369
O	O	O	O	O	X	O	0.6087
O	O	O	O	X	O	O	0.6376
O	O	O	X	O	O	O	0.6397
O	O	X	O	O	O	O	0.6514
O	X	O	O	O	O	O	0.6434
X	O	O	O	O	O	O	0.6372
O	O	O	O	O	O	O	0.6372

Table A.2: Doc2Vec Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	X	X	X	X	X	X	0.0836
O	X	X	X	X	X	X	0.0827
X	O	X	X	X	X	X	0.0807
X	X	O	X	X	X	X	0.1112
X	X	X	O	X	X	X	0.0596
X	X	X	X	O	X	X	0.1214
X	X	X	X	X	O	X	0.0552
X	X	X	X	X	X	O	0.0446
O	O	X	X	X	X	X	0.0939
O	X	O	X	X	X	X	0.1054
O	X	X	O	X	X	X	0.0464
O	X	X	X	O	X	X	0.1170
O	X	X	X	X	O	X	0.0514
O	X	X	X	X	X	O	0.0146

Table A.2: Doc2Vec Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	O	O	X	X	X	X	0.1130
X	O	X	O	X	X	X	0.0648
X	O	X	X	O	X	X	0.1218
X	O	X	X	X	O	X	0.0664
X	O	X	X	X	X	O	0.0420
X	X	O	O	X	X	X	0.0959
X	X	O	X	O	X	X	0.1118
X	X	O	X	X	O	X	0.0953
X	X	O	X	X	X	O	0.0828
X	X	X	O	O	X	X	0.1204
X	X	X	O	X	O	X	0.0617
X	X	X	O	X	X	O	0.0218
X	X	X	X	O	O	X	0.1182
X	X	X	X	O	X	O	0.0980
X	X	X	X	X	O	O	0.0402
O	O	O	X	X	X	X	0.1266
O	O	X	O	X	X	X	0.0560
O	O	X	X	O	X	X	0.1252
O	O	X	X	X	O	X	0.0647
O	O	X	X	X	X	O	0.0351
O	X	O	O	X	X	X	0.0971
O	X	O	X	O	X	X	0.1173
O	X	O	X	X	O	X	0.0842
O	X	O	X	X	X	O	0.0945
O	X	X	O	O	X	X	0.1114
O	X	X	O	X	O	X	0.0673
O	X	X	O	X	X	O	0.0288
O	X	X	X	O	O	X	0.1194
O	X	X	X	O	X	O	0.1153
O	X	X	X	X	O	O	0.0391
X	O	O	O	X	X	X	0.0876
X	O	O	X	O	X	X	0.1145
X	O	O	X	X	O	X	0.0988
X	O	O	X	X	X	O	0.0978
X	O	X	O	O	X	X	0.1137
X	O	X	O	X	O	X	0.0687

Table A.2: Doc2Vec Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	O	X	O	X	X	O	0.0415
X	O	X	X	O	O	X	0.1257
X	O	X	X	O	X	O	0.1145
X	O	X	X	X	O	O	0.0198
X	X	O	O	O	X	X	0.1157
X	X	O	O	X	O	X	0.0918
X	X	O	O	X	X	O	0.0979
X	X	O	X	O	O	X	0.0935
X	X	O	X	O	X	O	0.1035
X	X	O	X	X	O	O	0.0727
X	X	X	O	O	O	X	0.1166
X	X	X	O	O	X	O	0.1001
X	X	X	O	X	O	O	0.0299
X	X	X	X	O	O	O	0.0938
O	O	O	O	X	X	X	0.0873
O	O	O	X	O	X	X	0.1217
O	O	O	X	X	O	X	0.0888
O	O	O	X	X	X	O	0.0990
O	O	X	O	O	X	X	0.1275
O	O	X	O	X	O	X	0.0652
O	O	X	O	X	X	O	0.0113
O	O	X	X	O	O	X	0.1366
O	O	X	X	O	X	O	0.1156
O	O	X	X	X	O	O	0.0226
O	X	O	O	O	X	X	0.1122
O	X	O	O	X	O	X	0.0756
O	X	O	O	X	X	O	0.1023
O	X	O	X	O	O	X	0.1004
O	X	O	X	O	X	O	0.1111
O	X	O	X	X	O	O	0.0722
O	X	X	O	O	O	X	0.1160
O	X	X	O	O	X	O	0.1087
O	X	X	O	X	O	O	0.0490
O	X	X	X	O	O	O	0.1042
X	O	O	O	O	X	X	0.1180
X	O	O	O	X	O	X	0.0925

Table A.2: Doc2Vec Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	O	O	O	X	X	O	0.0923
X	O	O	X	O	O	X	0.0939
X	O	O	X	O	X	O	0.1020
X	O	O	X	X	O	O	0.0779
X	O	X	O	O	O	X	0.1305
X	O	X	O	O	X	O	0.1123
X	O	X	O	X	O	O	0.0430
X	O	X	X	O	O	O	0.1092
X	X	O	O	O	O	X	0.1061
X	X	O	O	O	X	O	0.1010
X	X	O	O	X	O	O	0.0681
X	X	O	X	O	O	O	0.0991
X	X	X	O	O	O	O	0.1077
O	O	O	O	O	X	X	0.1114
O	O	O	O	X	O	X	0.0834
O	O	O	O	X	X	O	0.0937
O	O	O	X	O	O	X	0.0914
O	O	O	X	O	X	O	0.0986
O	O	O	X	X	O	O	0.0728
O	O	X	O	O	O	X	0.1330
O	O	X	O	O	X	O	0.1163
O	O	X	O	X	O	O	0.0430
O	O	X	X	O	O	O	0.1238
O	X	O	O	O	O	X	0.1065
O	X	O	O	O	X	O	0.0985
O	X	O	O	X	O	O	0.0792
O	X	O	X	O	O	O	0.0940
O	X	X	O	O	O	O	0.1099
X	O	O	O	O	O	X	0.1092
X	O	O	O	O	X	O	0.0970
X	O	O	O	X	O	O	0.0713
X	O	O	X	O	O	O	0.0972
X	O	X	O	O	O	O	0.1188
X	X	O	O	O	O	O	0.1082
O	O	O	O	O	O	X	0.1077
O	O	O	O	O	X	O	0.1071

Table A.2: Doc2Vec Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
O	O	O	O	X	O	O	0.0807
O	O	O	X	O	O	O	0.0891
O	O	X	O	O	O	O	0.1227
O	X	O	O	O	O	O	0.1138
X	O	O	O	O	O	O	0.1076
O	O	O	O	O	O	O	0.1079

Table A.3: BERTweet Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	X	X	X	X	X	X	0.2552
O	X	X	X	X	X	X	0.2549
X	O	X	X	X	X	X	0.2638
X	X	O	X	X	X	X	0.3300
X	X	X	O	X	X	X	0.2661
X	X	X	X	O	X	X	0.2646
X	X	X	X	X	O	X	0.2767
X	X	X	X	X	X	O	0.1971
O	O	X	X	X	X	X	0.2636
O	X	O	X	X	X	X	0.3301
O	X	X	O	X	X	X	0.2659
O	X	X	X	O	X	X	0.2641
O	X	X	X	X	O	X	0.2765
O	X	X	X	X	X	O	0.1967
X	O	O	X	X	X	X	0.3320
X	O	X	O	X	X	X	0.2724
X	O	X	X	O	X	X	0.2941
X	O	X	X	X	O	X	0.2836
X	O	X	X	X	X	O	0.2081
X	X	O	O	X	X	X	0.2893
X	X	O	X	O	X	X	0.3353
X	X	O	X	X	O	X	0.3029
X	X	O	X	X	X	O	0.2370
X	X	X	O	O	X	X	0.2633
X	X	X	O	X	O	X	0.2769
X	X	X	O	X	X	O	0.1848

Table A.3: BERTweet Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	X	X	X	O	O	X	0.2770
X	X	X	X	O	X	O	0.1834
X	X	X	X	X	O	O	0.2055
O	O	O	X	X	X	X	0.3321
O	O	X	O	X	X	X	0.2723
O	O	X	X	O	X	X	0.2935
O	O	X	X	X	O	X	0.2835
O	O	X	X	X	X	O	0.2077
O	X	O	O	X	X	X	0.2892
O	X	O	X	O	X	X	0.3354
O	X	O	X	X	O	X	0.3029
O	X	O	X	X	X	O	0.2365
O	X	X	O	O	X	X	0.2628
O	X	X	O	X	O	X	0.2766
O	X	X	O	X	X	O	0.1843
O	X	X	X	O	O	X	0.2768
O	X	X	X	O	X	O	0.1820
O	X	X	X	X	O	O	0.2051
X	O	O	O	X	X	X	0.2896
X	O	O	X	O	X	X	0.3379
X	O	O	X	X	O	X	0.3075
X	O	O	X	X	X	O	0.2381
X	O	X	O	O	X	X	0.2916
X	O	X	O	X	O	X	0.2837
X	O	X	O	X	X	O	0.1945
X	O	X	X	O	O	X	0.3052
X	O	X	X	O	X	O	0.2120
X	O	X	X	X	O	O	0.2125
X	X	O	O	O	X	X	0.3058
X	X	O	O	X	O	X	0.3026
X	X	O	O	X	X	O	0.2255
X	X	O	X	O	O	X	0.3085
X	X	O	X	O	X	O	0.2409
X	X	O	X	X	O	O	0.2459
X	X	X	O	O	O	X	0.2827
X	X	X	O	O	X	O	0.1569

Table A.3: BERTweet Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	X	X	O	X	O	O	0.2058
X	X	X	X	O	O	O	0.1725
O	O	O	O	X	X	X	0.2895
O	O	O	X	O	X	X	0.3379
O	O	O	X	X	O	X	0.3076
O	O	O	X	X	X	O	0.2376
O	O	X	O	O	X	X	0.2911
O	O	X	O	X	O	X	0.2837
O	O	X	O	X	X	O	0.1940
O	O	X	X	O	O	X	0.3048
O	O	X	X	O	X	O	0.2103
O	O	X	X	X	O	O	0.2121
O	X	O	O	O	X	X	0.3058
O	X	O	O	X	O	X	0.3026
O	X	O	O	X	X	O	0.2253
O	X	O	X	O	O	X	0.3085
O	X	O	X	O	X	O	0.2407
O	X	O	X	X	O	O	0.2453
O	X	X	O	O	O	X	0.2826
O	X	X	O	O	X	O	0.1554
O	X	X	O	X	O	O	0.2054
O	X	X	X	O	O	O	0.1712
X	O	O	O	O	X	X	0.3068
X	O	O	O	X	O	X	0.3072
X	O	O	O	X	X	O	0.2287
X	O	O	X	O	O	X	0.3134
X	O	O	X	O	X	O	0.2418
X	O	O	X	X	O	O	0.2538
X	O	X	O	O	O	X	0.3113
X	O	X	O	O	X	O	0.1843
X	O	X	O	X	O	O	0.2128
X	O	X	X	O	O	O	0.1953
X	X	O	O	O	O	X	0.3169
X	X	O	O	O	X	O	0.2200
X	X	O	O	X	O	O	0.2459
X	X	O	X	O	O	O	0.2522

Table A.3: BERTweet Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	X	X	O	O	O	O	0.1842
O	O	O	O	O	X	X	0.3067
O	O	O	O	X	O	X	0.3073
O	O	O	O	X	X	O	0.2285
O	O	O	X	O	O	X	0.3136
O	O	O	X	O	X	O	0.2416
O	O	O	X	X	O	O	0.2532
O	O	X	O	O	O	X	0.3110
O	O	X	O	O	X	O	0.1823
O	O	X	O	X	O	O	0.2124
O	O	X	X	O	O	O	0.1938
O	X	O	O	O	O	X	0.3170
O	X	O	O	O	X	O	0.2200
O	X	O	O	X	O	O	0.2453
O	X	O	X	O	O	O	0.2515
O	X	X	O	O	O	O	0.1829
X	O	O	O	O	O	X	0.3217
X	O	O	O	O	X	O	0.2230
X	O	O	O	X	O	O	0.2538
X	O	O	X	O	O	O	0.2598
X	O	X	O	O	O	O	0.2073
X	X	O	O	O	O	O	0.2611
O	O	O	O	O	O	X	0.3219
O	O	O	O	O	X	O	0.2227
O	O	O	O	X	O	O	0.2532
O	O	O	X	O	O	O	0.2591
O	O	X	O	O	O	O	0.2057
O	X	O	O	O	O	O	0.2605
X	O	O	O	O	O	O	0.2690
O	O	O	O	O	O	O	0.2684

Table A.4: Sentence BERT Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	X	X	X	X	X	X	0.8071
O	X	X	X	X	X	X	0.8070

Table A.4: Sentence BERT Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	O	X	X	X	X	X	0.8105
X	X	O	X	X	X	X	0.8055
X	X	X	O	X	X	X	0.8071
X	X	X	X	O	X	X	0.8017
X	X	X	X	X	O	X	0.7572
X	X	X	X	X	X	O	0.8029
O	O	X	X	X	X	X	0.8102
O	X	O	X	X	X	X	0.8056
O	X	X	O	X	X	X	0.8070
O	X	X	X	O	X	X	0.8017
O	X	X	X	X	O	X	0.7572
O	X	X	X	X	X	O	0.8033
X	O	O	X	X	X	X	0.8084
X	O	X	O	X	X	X	0.8105
X	O	X	X	O	X	X	0.8049
X	O	X	X	X	O	X	0.7637
X	O	X	X	X	X	O	0.8069
X	X	O	O	X	X	X	0.8055
X	X	O	X	O	X	X	0.8028
X	X	O	X	X	O	X	0.7736
X	X	O	X	X	X	O	0.8074
X	X	X	O	O	X	X	0.7964
X	X	X	O	X	O	X	0.7576
X	X	X	O	X	X	O	0.7995
X	X	X	X	O	O	X	0.7675
X	X	X	X	O	X	O	0.7973
X	X	X	X	X	O	O	0.7438
O	O	O	X	X	X	X	0.8085
O	O	X	O	X	X	X	0.8102
O	O	X	X	O	X	X	0.8048
O	O	X	X	X	O	X	0.7636
O	O	X	X	X	X	O	0.8070
O	X	O	O	X	X	X	0.8056
O	X	O	X	O	X	X	0.8030
O	X	O	X	X	O	X	0.7737
O	X	O	X	X	X	O	0.8075

Table A.4: Sentence BERT Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
O	X	X	O	O	X	X	0.7965
O	X	X	O	X	O	X	0.7575
O	X	X	O	X	X	O	0.8000
O	X	X	X	O	O	X	0.7674
O	X	X	X	O	X	O	0.7977
O	X	X	X	X	O	O	0.7439
X	O	O	O	X	X	X	0.8084
X	O	O	X	O	X	X	0.8054
X	O	O	X	X	O	X	0.7784
X	O	O	X	X	X	O	0.8096
X	O	X	O	O	X	X	0.7981
X	O	X	O	X	O	X	0.7640
X	O	X	O	X	X	O	0.8037
X	O	X	X	O	O	X	0.7727
X	O	X	X	O	X	O	0.8013
X	O	X	X	X	O	O	0.7480
X	X	O	O	O	X	X	0.7980
X	X	O	O	X	O	X	0.7739
X	X	O	O	X	X	O	0.8015
X	X	O	X	O	O	X	0.7734
X	X	O	X	O	X	O	0.8050
X	X	O	X	X	O	O	0.7552
X	X	X	O	O	O	X	0.7641
X	X	X	O	O	X	O	0.7840
X	X	X	O	X	O	O	0.7442
X	X	X	X	O	O	O	0.7513
O	O	O	O	X	X	X	0.8085
O	O	O	X	O	X	X	0.8055
O	O	O	X	X	O	X	0.7785
O	O	O	X	X	X	O	0.8097
O	O	X	O	O	X	X	0.7981
O	O	X	O	X	O	X	0.7638
O	O	X	O	X	X	O	0.8038
O	O	X	X	O	O	X	0.7724
O	O	X	X	O	X	O	0.8014
O	O	X	X	X	O	O	0.7480

Table A.4: Sentence BERT Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
O	X	O	O	O	X	X	0.7983
O	X	O	O	X	O	X	0.7740
O	X	O	O	X	X	O	0.8016
O	X	O	X	O	O	X	0.7736
O	X	O	X	O	X	O	0.8055
O	X	O	X	X	O	O	0.7554
O	X	X	O	O	O	X	0.7640
O	X	X	O	O	X	O	0.7844
O	X	X	O	X	O	O	0.7444
O	X	X	X	O	O	O	0.7514
X	O	O	O	O	X	X	0.7997
X	O	O	O	X	O	X	0.7787
X	O	O	O	X	X	O	0.8017
X	O	O	X	O	O	X	0.7781
X	O	O	X	O	X	O	0.8076
X	O	O	X	X	O	O	0.7580
X	O	X	O	O	O	X	0.7693
X	O	X	O	O	X	O	0.7860
X	O	X	O	X	O	O	0.7484
X	O	X	X	O	O	O	0.7519
X	X	O	O	O	O	X	0.7697
X	X	O	O	O	X	O	0.7864
X	X	O	O	X	O	O	0.7556
X	X	O	X	O	O	O	0.7561
X	X	X	O	O	O	O	0.7492
O	O	O	O	O	X	X	0.8000
O	O	O	O	X	O	X	0.7788
O	O	O	O	X	X	O	0.8018
O	O	O	X	O	O	X	0.7782
O	O	O	X	O	X	O	0.8080
O	O	O	X	X	O	O	0.7582
O	O	X	O	O	O	X	0.7691
O	O	X	O	O	X	O	0.7861
O	O	X	O	X	O	O	0.7485
O	O	X	X	O	O	O	0.7519
O	X	O	O	O	O	X	0.7698

Table A.4: Sentence BERT Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
O	X	O	O	O	X	O	0.7870
O	X	O	O	X	O	O	0.7559
O	X	O	X	O	O	O	0.7564
O	X	X	O	O	O	O	0.7493
X	O	O	O	O	O	X	0.7743
X	O	O	O	O	X	O	0.7863
X	O	O	O	X	O	O	0.7585
X	O	O	X	O	O	O	0.7588
X	O	X	O	O	O	O	0.7499
X	X	O	O	O	O	O	0.7534
O	O	O	O	O	O	X	0.7744
O	O	O	O	O	X	O	0.7868
O	O	O	O	X	O	O	0.7587
O	O	O	X	O	O	O	0.7591
O	O	X	O	O	O	O	0.7499
O	X	O	O	O	O	O	0.7536
X	O	O	O	O	O	O	0.7562
O	O	O	O	O	O	O	0.7564

Table A.5: OpenAI Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	X	X	X	X	X	X	0.8413
O	X	X	X	X	X	X	0.8411
X	O	X	X	X	X	X	0.8425
X	X	O	X	X	X	X	0.8330
X	X	X	O	X	X	X	0.8412
X	X	X	X	O	X	X	0.8392
X	X	X	X	X	O	X	0.8090
X	X	X	X	X	X	O	0.8373
O	O	X	X	X	X	X	0.8423
O	X	O	X	X	X	X	0.8330
O	X	X	O	X	X	X	0.8410
O	X	X	X	O	X	X	0.8394
O	X	X	X	X	O	X	0.8089
O	X	X	X	X	X	O	0.8376

Table A.5: OpenAI Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	O	O	X	X	X	X	0.8302
X	O	X	O	X	X	X	0.8412
X	O	X	X	O	X	X	0.8410
X	O	X	X	X	O	X	0.8108
X	O	X	X	X	X	O	0.8383
X	X	O	O	X	X	X	0.8277
X	X	O	X	O	X	X	0.8351
X	X	O	X	X	O	X	0.8091
X	X	O	X	X	X	O	0.8352
X	X	X	O	O	X	X	0.8250
X	X	X	O	X	O	X	0.8089
X	X	X	O	X	X	O	0.8255
X	X	X	X	O	O	X	0.6958
X	X	X	X	O	X	O	0.8335
X	X	X	X	X	O	O	0.7935
O	O	O	X	X	X	X	0.8303
O	O	X	O	X	X	X	0.8410
O	O	X	X	O	X	X	0.8411
O	O	X	X	X	O	X	0.8107
O	O	X	X	X	X	O	0.8382
O	X	O	O	X	X	X	0.8278
O	X	O	X	O	X	X	0.8352
O	X	O	X	X	O	X	0.8091
O	X	O	X	X	X	O	0.8352
O	X	X	O	O	X	X	0.8252
O	X	X	O	X	O	X	0.8088
O	X	X	O	X	X	O	0.8257
O	X	X	X	O	O	X	0.6956
O	X	X	X	O	X	O	0.8337
O	X	X	X	X	O	O	0.7935
X	O	O	O	X	X	X	0.8261
X	O	O	X	O	X	X	0.8332
X	O	O	X	X	O	X	0.8088
X	O	O	X	X	X	O	0.8335
X	O	X	O	O	X	X	0.8263
X	O	X	O	X	O	X	0.8107

Table A.5: OpenAI Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	O	X	O	X	X	O	0.8252
X	O	X	X	O	O	X	0.6988
X	O	X	X	O	X	O	0.8345
X	O	X	X	X	O	O	0.7940
X	X	O	O	O	X	X	0.8234
X	X	O	O	X	O	X	0.8090
X	X	O	O	X	X	O	0.8134
X	X	O	X	O	O	X	0.8065
X	X	O	X	O	X	O	0.8317
X	X	O	X	X	O	O	0.7903
X	X	X	O	O	O	X	0.6950
X	X	X	O	O	X	O	0.8084
X	X	X	O	X	O	O	0.7932
X	X	X	X	O	O	O	0.6906
O	O	O	O	X	X	X	0.8262
O	O	O	X	O	X	X	0.8333
O	O	O	X	X	O	X	0.8088
O	O	O	X	X	X	O	0.8336
O	O	X	O	O	X	X	0.8263
O	O	X	O	X	O	X	0.8106
O	O	X	O	X	X	O	0.8252
O	O	X	X	O	O	X	0.6985
O	O	X	X	O	X	O	0.8347
O	O	X	X	X	O	O	0.7938
O	X	O	O	O	X	X	0.8232
O	X	O	O	X	O	X	0.8090
O	X	O	O	X	X	O	0.8135
O	X	O	X	O	O	X	0.8065
O	X	O	X	O	X	O	0.8317
O	X	O	X	X	O	O	0.7904
O	X	X	O	O	O	X	0.6948
O	X	X	O	O	X	O	0.8085
O	X	X	O	X	O	O	0.7932
O	X	X	X	O	O	O	0.6904
X	O	O	O	O	X	X	0.8226
X	O	O	O	X	O	X	0.8088

Table A.5: OpenAI Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
X	O	O	O	X	X	O	0.8130
X	O	O	X	O	O	X	0.8066
X	O	O	X	O	X	O	0.8300
X	O	O	X	X	O	O	0.7903
X	O	X	O	O	O	X	0.6978
X	O	X	O	O	X	O	0.8069
X	O	X	O	X	O	O	0.7937
X	O	X	X	O	O	O	0.5914
X	X	O	O	O	O	X	0.8067
X	X	O	O	O	X	O	0.8004
X	X	O	O	X	O	O	0.7901
X	X	O	X	O	O	O	0.7909
X	X	X	O	O	O	O	0.6887
O	O	O	O	O	X	X	0.8223
O	O	O	O	X	O	X	0.8088
O	O	O	O	X	X	O	0.8131
O	O	O	X	O	O	X	0.8066
O	O	O	X	O	X	O	0.8300
O	O	O	X	X	O	O	0.7905
O	O	X	O	O	O	X	0.6975
O	O	X	O	O	X	O	0.8069
O	O	X	O	X	O	O	0.7935
O	O	X	X	O	O	O	0.5910
O	X	O	O	O	O	X	0.8067
O	X	O	O	O	X	O	0.8004
O	X	O	O	X	O	O	0.7903
O	X	O	X	O	O	O	0.7911
O	X	X	O	O	O	O	0.6885
X	O	O	O	O	O	X	0.8069
X	O	O	O	O	X	O	0.8009
X	O	O	O	X	O	O	0.7901
X	O	O	X	O	O	O	0.7911
X	O	X	O	O	O	O	0.6900
X	X	O	O	O	O	O	0.7873
O	O	O	O	O	O	X	0.8069
O	O	O	O	O	X	O	0.8009

Table A.5: OpenAI Text Preprocessing Pipeline PC

EMO	HAS	PUN	LOW	LEM	STE	STO	PC
O	O	O	O	X	O	O	0.7903
O	O	O	X	O	O	O	0.7912
O	O	X	O	O	O	O	0.6896
O	X	O	O	O	O	O	0.7874
X	O	O	O	O	O	O	0.7876
O	O	O	O	O	O	O	0.7877

Bibliography

[1] Elon Musk [@elonmusk]. I don't read the legacy media propaganda much anymore. It's a waste of time and a sadness generator. Just get my news from X – much more immediate, has actual world-class subject matter experts and tons of humor. Sooo much better!, September 2023. URL https://twitter.com/elonmusk/status/1707860072595083623.

[2] Elon Musk [@elonmusk]. Newspapers basically just report on what they read yesterday on X lmao, September 2023. URL https://twitter.com/elonmusk/status/1707864159784927270.

[3] Yuhao Zhang, Peng Qi, and Christopher D. Manning. Graph Convolution over Pruned Dependency Trees Improves Relation Extraction, September 2018. URL http://arxiv.org/abs/1809.10185. arXiv:1809.10185 [cs].

[4] Francesca Incitti, Federico Urli, and Lauro Snidaro. Beyond word embeddings: A survey. *Information Fusion*, 89:418–436, January 2023. ISSN 15662535. doi: 10.1016/j.inffus.2022.08.024. URL https://linkinghub.elsevier.com/retrieve/pii/S1566253522001233.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space, September 2013. URL http://arxiv.org/abs/1301.3781. arXiv:1301.3781 [cs].

[6] Billy Chiu and Simon Baker. Word embeddings for biomedical natural language processing: A survey. *Language and Linguistics Compass*, 14, December 2020. doi: 10.1111/lnc3.12402.

[7] E. Adar and L.A. Adamic. Tracking Information Epidemics in Blogspace. In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pages 207–214, Compiegne, France, 2005. IEEE. ISBN 978-0-7695-2415-3. doi: 10.1109/WI.2005.151. URL http://ieeexplore.ieee.org/document/1517844/.

[8] Anirudh Joshi and Richard O. Sinnott. Modelling Implicit Content Networks to Track Information Propagation Across Media Sources to Analyze News Events.

In *2018 IEEE 14th International Conference on e-Science (e-Science)*, pages 475–485, October 2018. doi: 10.1109/eScience.2018.00136.

[9] Mario Nawfal [@MarioNawfal]. BREAKING: Israeli Commander Nimrod Aloni has been captured by Hamas in the ongoing war. Analyst: "This is big. Israel is meant to be a big military force. This shows the extent to which they are on the back foot and not ready to respond to this attack. This also shows the... https://t.co/UCCy3NSHhD, October 2023. URL https://twitter.com/MarioNawfal/status/1710602173262840257.

[10] Ramón Casillas, Helena Gómez-Adorno, Victor Lomas-Barrie, and Orlando Ramos-Flores. Automatic Fact Checking Using an Interpretable Bert-Based Architecture on COVID-19 Claims. *Applied Sciences*, 12(20):10644, January 2022. ISSN 2076-3417. doi: 10.3390/app122010644. URL https://www.mdpi.com/2076-3417/12/20/10644. Number: 20 Publisher: Multidisciplinary Digital Publishing Institute.

[11] Rosa Sicilia, Stella Lo Giudice, Yulong Pei, Mykola Pechenizkiy, and Paolo Soda. Twitter rumour detection in the health domain. *Expert Systems with Applications*, 110:33–40, November 2018. ISSN 0957-4174. doi: 10.1016/j.eswa.2018.05.019. URL https://www.sciencedirect.com/science/article/pii/S0957417418303129.

[12] L. Morandini, A. R. Mohammad, and R. O. Sinnott. MAPPING THE CHATTER: SPATIAL METAPHORS FOR DYNAMIC TOPIC MODELLING OF SOCIAL MEDIA. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVIII-4/W1-2022:315–320, August 2022. ISSN 2194-9034. doi: 10.5194/isprs-archives-XLVIII-4-W1-2022-315-2022. URL https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLVIII-4-W1-2022/315/2022/.

[13] Felipe Almeida and Geraldo Xexéo. Word Embeddings: A Survey, May 2023. URL http://arxiv.org/abs/1901.09069. arXiv:1901.09069 [cs, stat].

[14] Elisa Shearer and Amy Mitchell. News Use Across Social Media Platforms in 2020. Report, Pew Research Center, January 2021. URL https://www.pewresearch.org/journalism/wp-content/uploads/sites/8/2021/01/PJ_2021.01.12_News-and-Social-Media_FINAL.pdf.

[15] Jeffrey Gottfried and Elisa Shearer. News use across social media platforms 2016. Report, Pew Research Center, May 2016. URL https://www.pewresearch.org/journalism/wp-content/uploads/sites/8/2016/05/PJ_2016.05.26_social-media-and-news_FINAL-1.pdf.

[16] Xinzhi Zhang and Wenshu Li. From Social Media with News: Journalists' Social Media Use for Sourcing and Verification. *Journalism Practice*, 14(10):1193–1210, November 2020. ISSN 1751-2786. doi: 10.1080/17512786.2019.1689372. URL https://doi.org/10.1080/17512786.2019.1689372. Publisher: Routledge _eprint: https://doi.org/10.1080/17512786.2019.1689372.

[17] Rune Karlsen and Toril Aalberg. Social Media and Trust in News: An Experimental Study of the Effect of Facebook on News Story Credibility. *Digital Journalism*, 11(1):144–160, January 2023. ISSN 2167-0811. doi: 10.1080/21670811.2021.1945938. URL https://doi.org/10.1080/21670811.2021.1945938. Publisher: Routledge _eprint: https://doi.org/10.1080/21670811.2021.1945938.

[18] Valerio La Gatta, Chiyu Wei, Luca Luceri, Francesco Pierri, and Emilio Ferrara. Retrieving false claims on Twitter during the Russia-Ukraine conflict. In *Companion Proceedings of the ACM Web Conference 2023*, pages 1317–1323, April 2023. doi: 10.1145/3543873.3587571. URL http://arxiv.org/abs/2303.10121. arXiv:2303.10121 [cs].

[19] PolitiFact, . URL https://www.politifact.com/.

[20] Gene M. Amdahl. Computer Architecture and Amdahl's Law. *Computer*, 46(12):38–46, December 2013. ISSN 1558-0814. doi: 10.1109/MC.2013.418. URL https://ieeexplore.ieee.org/document/6689270. Conference Name: Computer.

[21] John L. Gustafson. Reevaluating Amdahl's law. *Communications of the ACM*, 31(5):532–533, 1988. ISSN 0001-0782. doi: 10.1145/42411.42415. URL https://dl.acm.org/doi/10.1145/42411.42415.

[22] Daniel Jurafsky and James Martin. *Speech and Language Processing*. Third edition draft edition, January 2023. URL https://web.stanford.edu/~jurafsky/slp3/.

[23] Abhinav Kathuria, Anu Gupta, and R. K. Singla. A Review of Tools and Techniques for Preprocessing of Textual Data. In Vijendra Singh, Vijayan K. Asari, Sanjay Kumar, and R. B. Patel, editors, *Computational Methods and Data Engineering*, Advances in Intelligent Systems and Computing, pages 407–422, Singapore, 2021. Springer. ISBN 9789811568763. doi: 10.1007/978-981-15-6876-3_31.

[24] Lipika Dey and S K Mirajul Haque. Opinion mining from noisy text data. In *Proceedings of the second workshop on Analytics for noisy unstructured text data*, AND '08, pages 83–90, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 978-1-60558-196-5. doi: 10.1145/1390749.1390763. URL https://dl.acm.org/doi/10.1145/1390749.1390763.

[25] Xindong Wu and Xingquan Zhu. Mining With Noise Knowledge: Error-Aware Data Mining. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 38(4):917–932, July 2008. ISSN 1558-2426. doi: 10.1109/TSMCA.2008.923034. URL https://ieeexplore.ieee.org/abstract/document/4544889. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans.

[26] Magdalena Biesialska, Katarzyna Biesialska, and Marta R. Costa-jussà. Continual Lifelong Learning in Natural Language Processing: A Survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6523–6541, 2020. doi: 10.18653/v1/2020.coling-main.574. URL http://arxiv.org/abs/2012.09823. arXiv:2012.09823 [cs].

[27] Alper Kursat Uysal and Serkan Gunal. The impact of preprocessing on text classification. *Information Processing & Management*, 50(1):104–112, January 2014. ISSN 0306-4573. doi: 10.1016/j.ipm.2013.08.006. URL https://www.sciencedirect.com/science/article/pii/S0306457313000964.

[28] Matthew J. Denny and Arthur Spirling. Text Preprocessing For Unsupervised Learning: Why It Matters, When It Misleads, And What To Do About It. *Political Analysis*, 26(2):168–189, April 2018. ISSN 1047-1987, 1476-4989. doi: 10.1017/pan.2017.44. URL https://www.cambridge.org/core/journals/political-analysis/article/text-preprocessing-for-unsupervised-learning-why-it-matters-when-it-misleads-and-AA7D4DE0AA6AB208502515AE3EC6989E. Publisher: Cambridge University Press.

[29] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. "O'Reilly Media, Inc.", June 2009. ISBN 978-0-596-55571-9. Google-Books-ID: KGIbfiiP1i4C.

[30] Christopher Manning and Hinrich Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, May 1999. ISBN 978-0-262-30379-8. Google-Books-ID: 3qnuDwAAQBAJ.

[31] Julia Silge and David Robinson. tidytext: Text Mining and Analysis Using Tidy Data Principles in R. *The Journal of Open Source Software*, 1(3):37, July 2016. ISSN 2475-9066. doi: 10.21105/joss.00037. URL http://joss.theoj.org/papers/10.21105/joss.00037.

[32] Subbu Kannan, Vairaprakash Gurusamy, S. Vijayarani, J. Ilamathi, Ms Nithya, S. Kannan, and V. Gurusamy. Preprocessing techniques for text mining. *International Journal of Computer Science & Communication Networks*, 5(1):7–16, 2014. URL https://www.academia.edu/download/54879053/PreprocessingTechniquesforTextMining.pdf.

[33] Zellig S. Harris. Distributional Structure. *WORD*, 10(2-3):146–162, August 1954. ISSN 0043-7956. doi: 10.1080/00437956.1954.11659520. URL https://doi.org/10.1080/00437956.1954.11659520. Publisher: Routledge _eprint: https://doi.org/10.1080/00437956.1954.11659520.

[34] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990. ISSN 1097-4571. doi: 10.1002/(SICI)1097-4571(199009)41:6⟨391::AID-ASI1⟩3.0.CO;2-9. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-4571%28199009%2941%3A6%3C391%3A%3AAID-ASI1%3E3.0.CO%3B2-9. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291097-4571%28199009%2941%3A6%3C391%3A%3AAID-ASI1%3E3.0.CO%3B2-9.

[35] Kevin Lund and Curt Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208, June 1996. ISSN 1532-5970. doi: 10.3758/BF03204766. URL https://doi.org/10.3758/BF03204766.

[36] Douglas LT Rohde, Laura M. Gonnerman, and David C. Plaut. An improved model of semantic similarity based on lexical co-occurrence. *Communications of the ACM*, 8(627-633):116, 2006. URL https://www.cnbc.cmu.edu/~plaut/papers/pdf/RohdeGonnermanPlautSUB-CogSci.COALS.pdf.

[37] Paramveer Dhillon, Dean P Foster, and Lyle Ungar. Multi-View Learning of Word Embeddings via CCA. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper_files/paper/2011/hash/6c4b761a28b734fe93831e3fb400ce87-Abstract.html.

[38] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 160–167, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390177. URL https://dl.acm.org/doi/10.1145/1390156.1390177.

[39] Andriy Mnih and Geoffrey E Hinton. A Scalable Hierarchical Distributed Language Model. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008. URL https://proceedings.neurips.cc/paper_files/paper/2008/hash/1e056d2b0ebd5c878c550da6ac5d3724-Abstract.html.

[40] Rémi Lebret and Ronan Collobert. Word Emdeddings through Hellinger PCA, January 2017. URL http://arxiv.org/abs/1312.5542. arXiv:1312.5542 [cs].

[41] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

[42] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

[43] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, January 1988. ISSN 0306-4573. doi: 10.1016/0306-4573(88)90021-0. URL https://www.sciencedirect.com/science/article/pii/0306457388900210.

[44] Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *IJCAI*, volume 3, pages 587–592. Citeseer, 2003. URL https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=8402877d099255897e9fc0f2fb42a40fbf746050. Issue: 2003.

[45] Omer Levy and Yoav Goldberg. Neural Word Embedding as Implicit Matrix Factorization. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/hash/feab05aa91085b7a8012516bc3533958-Abstract.html.

[46] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3(null): 1137–1155, 2003. ISSN 1532-4435.

[47] Yoshua Bengio and Jean-Sébastien Senecal. Quick Training of Probabilistic Neural Nets by Importance Sampling. In *International Workshop on Artificial Intelligence*

and Statistics, pages 17–24. PMLR, January 2003. URL https://proceedings.mlr.press/r4/bengio03a.html. ISSN: 2640-3498.

[48] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *International workshop on artificial intelligence and statistics*, pages 246–252. PMLR, 2005. URL http://proceedings.mlr.press/r5/morin05a/morin05a.pdf.

[49] Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 641–648, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273577. URL https://dl.acm.org/doi/10.1145/1273496.1273577.

[50] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, March 2016. URL http://arxiv.org/abs/1603.04467. arXiv:1603.04467 [cs].

[51] Michael U. Gutmann and Aapo Hyvärinen. Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. *Journal of machine learning research*, 13(2), 2012. URL https://www.jmlr.org/papers/volume13/gutmann12a/gutmann12a.pdf.

[52] Andriy Mnih and Yee Whye Teh. A Fast and Simple Algorithm for Training Neural Probabilistic Language Models, June 2012. URL http://arxiv.org/abs/1206.6426. arXiv:1206.6426 [cs].

[53] Quoc Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1188–1196. PMLR, June 2014. URL https://proceedings.mlr.press/v32/le14.html. ISSN: 1938-7228.

[54] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter*

of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

[55] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[56] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. URL https://www.mikecaptain.com/resources/pdf/GPT-1.pdf. Publisher: OpenAI.

[57] Deepak Suresh Asudani, Naresh Kumar Nagwani, and Pradeep Singh. Impact of word embedding models on text analytics in deep learning environment: a review. *Artificial Intelligence Review*, 56(9):10345–10425, September 2023. ISSN 1573-7462. doi: 10.1007/s10462-023-10419-1. URL https://doi.org/10.1007/s10462-023-10419-1.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[59] M. V. Koroteev. BERT: A Review of Applications in Natural Language Processing and Understanding, March 2021. URL http://arxiv.org/abs/2103.11943. arXiv:2103.11943 [cs].

[60] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015. URL https://www.cv-foundation.org/openaccess/content_iccv_2015/html/Zhu_Aligning_Books_and_ICCV_2015_paper.html.

[61] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa:

A Robustly Optimized BERT Pretraining Approach, July 2019. URL http://arxiv.org/abs/1907.11692. arXiv:1907.11692 [cs].

[62] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, February 2020. URL http://arxiv.org/abs/1910.01108. arXiv:1910.01108 [cs].

[63] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.2. URL https://aclanthology.org/2020.emnlp-demos.2.

[64] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410.

[65] Anne-Lyse Minard, Manuela Speranza, Eneko Agirre, Itziar Aldabe, Marieke Van Erp, Bernardo Magnini, German Rigau, and Ruben Urizar. SemEval-2015 Task 4: TimeLine: Cross-Document Event Ordering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 778–786, Denver, Colorado, 2015. Association for Computational Linguistics. doi: 10.18653/v1/S15-2132. URL http://aclweb.org/anthology/S15-2132.

[66] Marco Rospocher, Marieke van Erp, Piek Vossen, Antske Fokkens, Itziar Aldabe, German Rigau, Aitor Soroa, Thomas Ploeger, and Tessel Bogaard. Building event-centric knowledge graphs from news. *Journal of Web Semantics*, 37-38:132–151, March 2016. ISSN 1570-8268. doi: 10.1016/j.websem.2015.12.004. URL https://www.sciencedirect.com/science/article/pii/S1570826815001456.

[67] Jaewon Yang and Jure Leskovec. Modeling Information Diffusion in Implicit Networks. In *2010 IEEE International Conference on Data Mining*, pages 599–608, December 2010. doi: 10.1109/ICDM.2010.22. ISSN: 2374-8486.

[68] Jure Leskovec, Lars Backstrom, and Jon Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 497–506,

New York, NY, USA, 2009. Association for Computing Machinery. ISBN 978-1-60558-495-9. doi: 10.1145/1557019.1557077. URL https://dl.acm.org/doi/10.1145/1557019.1557077.

[69] Caroline Suen, Sandy Huang, Chantat Eksombatchai, Rok Sosic, and Jure Leskovec. NIFTY: a system for large scale information flow tracking and clustering. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1237–1248, Rio de Janeiro Brazil, May 2013. ACM. ISBN 978-1-4503-2035-1. doi: 10.1145/2488388.2488496. URL https://dl.acm.org/doi/10.1145/2488388.2488496.

[70] Giovanni Colavizza, Mario Infelise, and Frédéric Kaplan. Mapping the Early Modern News Flow: An Enquiry by Robust Text Reuse Detection. In Luca Maria Aiello and Daniel McFarland, editors, *Social Informatics*, Lecture Notes in Computer Science, pages 244–253, Cham, 2015. Springer International Publishing. ISBN 978-3-319-15168-7. doi: 10.1007/978-3-319-15168-7_31.

[71] Svitlana Vakulenko, Max C. Göbel, Arno Scharl, and Lyndon JB Nixon. Visualising the Propagation of News on the Web. In *NewsIR@ ECIR*, pages 60–62, 2016. URL https://ceur-ws.org/Vol-1568/paper11.pdf.

[72] George A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL https://dl.acm.org/doi/10.1145/219717.219748.

[73] David PA Corney, Dyaa Albakour, Miguel Martinez-Alvarez, and Samir Moussa. What do a million news articles look like? In *NewsIR@ ECIR*, pages 42–47, 2016.

[74] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-5010. URL https://aclanthology.org/P14-5010.

[75] Jey Han Lau and Timothy Baldwin. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation, July 2016. URL http://arxiv.org/abs/1607.05368. arXiv:1607.05368 [cs].

[76] Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. *SEM 2013 shared task: Semantic Textual Similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the*

Main Conference and the Shared Task: Semantic Textual Similarity, pages 32–43, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics. URL https://aclanthology.org/S13-1004.

[77] Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada, July 2012. Association for Computational Linguistics. URL https://aclanthology.org/S12-1051.

[78] Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91, Dublin, Ireland, August 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2010. URL https://aclanthology.org/S14-2010.

[79] Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, Denver, Colorado, June 2015. Association for Computational Linguistics. doi: 10.18653/v1/S15-2045. URL https://aclanthology.org/S15-2045.

[80] Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1081. URL https://aclanthology.org/S16-1081.

[81] Eneko Agirre, Aitor Gonzalez-Agirre, Iñigo Lopez-Gazpio, Montse Maritxalar, German Rigau, and Larraitz Uria. SemEval-2016 Task 2: Interpretable Semantic Textual Similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 512–524, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1082. URL https://aclanthology.org/S16-1082.

[82] Anna Huang. Similarity measures for text document clustering. *Proceedings of the 6th New Zealand Computer Science Research Student Conference*, January 2008.

[83] Nitin Bhatia and Vandana. Survey of Nearest Neighbor Techniques, July 2010. URL http://arxiv.org/abs/1007.0085. arXiv:1007.0085 [cs].

[84] P. Berkhin. A Survey of Clustering Data Mining Techniques. In Jacob Kogan, Charles Nicholas, and Marc Teboulle, editors, *Grouping Multidimensional Data: Recent Advances in Clustering*, pages 25–71. Springer, Berlin, Heidelberg, 2006. ISBN 978-3-540-28349-2. doi: 10.1007/3-540-28349-8_2. URL https://doi.org/10.1007/3-540-28349-8_2.

[85] Farzeen Zehra, Maha Javed, Darakhshan Khan, and Maria Pasha. Comparative Analysis of C++ and Python in Terms of Memory and Time, December 2020. URL https://www.preprints.org/manuscript/202012.0516/v1.

[86] Amiya K. Maji, Lev Gorenstein, and Geoffrey Lentner. Demystifying Python Package Installation with conda-env-mod. In *2020 IEEE/ACM International Workshop on HPC User Support Tools (HUST) and Workshop on Programming and Performance Visualization Tools (ProTools)*, pages 27–37, November 2020. doi: 10.1109/HUSTProtools51951.2020.00011. URL https://ieeexplore.ieee.org/abstract/document/9308091.

[87] Sunsetting Python 2, . URL https://www.python.org/doc/sunset-python-2/.

[88] Amir Bakarov. A Survey of Word Embeddings Evaluation Methods, January 2018. URL http://arxiv.org/abs/1801.09536. arXiv:1801.09536 [cs].

[89] Wei Xu, Chris Callison-Burch, and Bill Dolan. SemEval-2015 Task 1: Paraphrase and Semantic Similarity in Twitter (PIT). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 1–11, Denver, Colorado, 2015. Association for Computational Linguistics. doi: 10.18653/v1/S15-2001. URL http://aclweb.org/anthology/S15-2001.

[90] Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. Real-time Rumor Debunking on Twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1867–1870, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 978-1-4503-3794-6. doi: 10.1145/2806416.2806651. URL https://dl.acm.org/doi/10.1145/2806416.2806651.

[91] Jing MA, Wei GAO, Prasenjit MITRA, Sejeong KWON, Bernard J. JANSEN, Kam-Fai WONG, and Meeyoung CHA. Detecting rumors from microblogs with

recurrent neural networks. *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*, pages 3818–3824, July 2016. URL https://ink.library.smu.edu.sg/sis_research/4630.

[92] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12 (85):2825–2830, 2011. ISSN 1533-7928. URL http://jmlr.org/papers/v12/pedregosa11a.html.

[93] Radim Rehurek and Petr Sojka. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2):2, 2011.

[94] OpenAI Platform, . URL https://platform.openai.com.

[95] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-demos.14. URL https://aclanthology.org/2020.acl-demos.14.

[96] Thomas N. Theis and H.-S. Philip Wong. The End of Moore's Law: A New Beginning for Information Technology. *Computing in Science & Engineering*, 19(2):41–50, March 2017. ISSN 1558-366X. doi: 10.1109/MCSE.2017.29. URL https://ieeexplore.ieee.org/abstract/document/7878935. Conference Name: Computing in Science & Engineering.

[97] X. H. Sun and L. M. Ni. Scalable Problems and Memory-Bounded Speedup. *Journal of Parallel and Distributed Computing*, 19(1):27–37, September 1993. ISSN 0743-7315. doi: 10.1006/jpdc.1993.1087. URL https://www.sciencedirect.com/science/article/pii/S0743731583710877.

[98] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 675–684, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 978-1-4503-0632-4. doi: 10.1145/1963405.1963500. URL https://dl.acm.org/doi/10.1145/1963405.1963500.

[99] Michael Mathioudakis and Nick Koudas. TwitterMonitor: trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, SIGMOD '10, pages 1155–1158, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 978-1-4503-0032-2. doi: 10.1145/1807167.1807306. URL https://dl.acm.org/doi/10.1145/1807167.1807306.

[100] Lin Tian, Xiuzhen Zhang, Yan Wang, and Huan Liu. Early Detection of Rumours on Twitter via Stance Transfer Learning. In Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins, editors, *Advances in Information Retrieval*, Lecture Notes in Computer Science, pages 575–588, Cham, 2020. Springer International Publishing. ISBN 978-3-030-45439-5. doi: 10.1007/978-3-030-45439-5_38.

[101] Jing MA, Wei GAO, and Kam-Fai WONG. Detect rumors in microblog posts using propagation structure via kernel learning. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017), Vancouver, Canada, 2017 July 30 - August 4*, pages 708–717, August 2017. doi: 10.18653/v1/P17-1066. URL https://ink.library.smu.edu.sg/sis_research/4563.

[102] Sahana V. P, Alwyn R Pias, Richa Shastri, and Shweta Mandloi. Automatic detection of rumoured tweets and finding its origin. In *2015 International Conference on Computing and Network Communications (CoCoNet)*, pages 607–612, December 2015. doi: 10.1109/CoCoNet.2015.7411251.

[103] Aditi Gupta, Hemank Lamba, Ponnurangam Kumaraguru, and Anupam Joshi. Faking Sandy: characterizing and identifying fake images on Twitter during Hurricane Sandy. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13 Companion, pages 729–736, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 978-1-4503-2038-2. doi: 10.1145/2487788.2488033. URL https://dl.acm.org/doi/10.1145/2487788.2488033.

[104] Oktie Hassanzadeh, Parul Awasthy, Ken Barker, Onkar Bhardwaj, Debarun Bhattacharjya, Mark Feblowitz, Lee Martie, Jian Ni, Kavitha Srinivas, and Lucy Yip. Knowledge-Based News Event Analysis and Forecasting Toolkit. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 5904–5907, Vienna, Austria, July 2022. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-1-956792-00-3. doi: 10.24963/ijcai.2022/850. URL https://www.ijcai.org/proceedings/2022/850.

[105] Gregor Leban, Blaz Fortuna, Janez Brank, and Marko Grobelnik. Event registry: learning about world events from news. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 107–110, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 978-1-4503-2745-9. doi: 10.1145/2567948.2577024. URL https://dl.acm.org/doi/10.1145/2567948.2577024.

[106] Bayu Distiawan Trisedya, Gerhard Weikum, Jianzhong Qi, and Rui Zhang. Neural Relation Extraction for Knowledge Base Enrichment. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 229–240, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1023. URL https://aclanthology.org/P19-1023.

[107] Bayu Distiawan Trisedya, Jianzhong Qi, and Rui Zhang. Entity Alignment between Knowledge Graphs Using Attribute Embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):297–304, July 2019. ISSN 2374-3468. doi: 10.1609/aaai.v33i01.3301297. URL https://ojs.aaai.org/index.php/AAAI/article/view/3798. Number: 01.

[108] Australian Digital Observatory -, March 2023. URL https://www.ado.eresearch.unimelb.edu.au/.

[109] Maarten Grootendorst. BERTopic: Neural topic modeling with a class-based TF-IDF procedure, March 2022. URL http://arxiv.org/abs/2203.05794. arXiv:2203.05794 [cs].

[110] Kevin Makice. *Twitter API: Up and Running: Learn How to Build Applications with the Twitter API*. "O'Reilly Media, Inc.", March 2009. ISBN 978-0-596-55551-1. Google-Books-ID: kBXXZNcRCGcC.

[111] J. Chris Anderson, Jan Lehnardt, and Noah Slater. *CouchDB: The Definitive Guide: Time to Relax*. "O'Reilly Media, Inc.", January 2010. ISBN 978-1-4493-8293-3.

[112] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. ISSN 0001-0782. doi: 10.1145/1327452.1327492. URL https://dl.acm.org/doi/10.1145/1327452.1327492.

[113] Justine Calma. Scientists say they can't rely on Twitter anymore, May 2023. URL https://www.theverge.com/2023/5/31/23739084/twitter-elon-musk-api-policy-chilling-academic-research.

[114] Peter Mell and Tim Grance. The NIST Definition of Cloud Computing. Technical Report NIST Special Publication (SP) 800-145, National Institute of Standards and Technology, September 2011. URL https://csrc.nist.gov/pubs/sp/800/145/final.

[115] Melbourne Research Cloud Documentation, . URL https://docs.cloud.unimelb.edu.au/guides/access_from_nectar/.

[116] ARDC Nectar Research Cloud | ARDC, April 2023. URL https://ardc.edu.au/services/ardc-nectar-research-cloud/.

[117] TikTok for Developers | TikTok, . URL https://developers.tiktok.com/products/research-api/.