(temp) Using Word Embedding for Citation Recommendation in FinTech Scientific Articles

by Ting-Chun Chen,

Submitted to The University of Nottingham

September 2023

in partial fulfilment of the conditions for the award of the degree of

Master of Science in Data Science

I declare that this dissertation is all my own work, except as indicated in the text

Contents

Abstract

According to the guidelines the abstract should not exceed 300 words. However, it is unlikely that anyone will be counting when you submit. If you still wish to do so, then better use Emacs `count-words` command.

Acknowledgements

It is good to thank here your supervisors and any sponsoring bodies, as well as any family, friends, cats, dogs etc. that have been supportive during your time at University.

Chapter 1

Introduction

1.1 Citation in Scientific Articles

Citations are used to demonstrate research background, existing techniques and pieces of evidence for a statement, playing a critical role in scientific writing. Authors can properly acknowledge the source of information of a statement and avoid plagiarism with an accurate citation. It also serves as a verification that the idea of this statement is provided and supported by previous studies. These days, references and citations also include additional information for search engines and citation recommendation systems to recognise the subfields and similar research of this article.

There has been a noticeable increase in the number of scientific articles being published. 3.8 million scientific articles are being published in 2022, according to the Web of Science database. It is expected that there would be more scientific articles available on the internet. It can be harder for academics to absorb all the new perspectives with their exact sources.

When looking for reference papers, keywords are commonly used in most academic databases and search engines. The authors would then review the results of papers from search engines to evaluate their relevance to our research and reliability. It could cost a huge amount of time and effort to go through full papers, not to mention that the academic database and search engines might miss some relevant papers.

1.2 Natural Language Processing

Natural language processing (NLP) refers to the approach of giving computers the ability to understand the meaning and thoughts of words, sentences and articles just like human beings do, building a way of communication between computer and human language. However, human languages have several characteristics that make it tricky to process them. The variability and informality, such as different vocabularies, syntax and phrases in different cultures or individuals representing a similar meaning, bring difficulties to applying a general transformations pipeline for information extraction. A large amount of noise such as mis-spellings, irrelevant contexts and grammar errors confuse not only human readers but also machines and affect the performance of language processing algorithms, not to mention that people might have different understandings of the same sentence which yield even more inaccuracies. Some advanced circumstances such as sarcasm, irony and the speaker's intention, whose implied meanings are beyond plain text, are more difficult for algorithms to detect and extract straightforwardly. Problems in normal vector data such as missing values and lack of labelling occur in natural language as well.

NLP manages to overcome the challenges of processing human language mentioned above and enables machines to analyse, cluster or classify not only vector instances but also texts. Advanced applications of NLP, such as sentiment analysis, text clustering, text summarisation, human language generation and improving automatic translation, provide solutions and assistance to a wide range of tasks. It reduces our workloads and time-spending and helps improve the interaction between humans and computers by mimicking human language to express in an easy-understanding way for most o the people not only programmers.

1.3 Text Embedding and Similarity

Text embedding, also known as text vectorisation and text featurisation, aims to extract information from words and sentences and represent the semantics and meaning of words by numbers and vectors. The process of extracting information has been through significant development since the year 2000. Various techniques have been proposed, including naive ways of bag-of-words model, TF-IDF encoding, LSA encoding, Word2Vec embeddings and GloVe, along with more advanced methods like BERT and GPT, which we are familiar with and commonly use today. After transforming every word into a vector, similarities between two words are to be calculated. The measurements of word similarity are used in word-embedding techniques to evaluate the similarity between prediction value and real value to update parameters in NN model training.

1.4 Semantic Search Engine

Techniques about search engines are a good starting point when it comes to matching a sentence to related articles. Search engines, especially full-text search engines, enable users to look for relative information in a huge full-text database. Among the tasks in a search process, the comparison between keyword and candidate results can be improved with text embedding techniques to retrieve semantic information from keyword and candidate documents, which is known as the semantic search engine.

Semantic search engines enable users to provide a sentence or description instead of just a keyword for querying, while text embedding methods are applied normally to both query texts and candidate documents to evaluate the similarity between their embedded vectors. Semantic search can better catch the contextual meaning within texts and understand the user's intent. Some advanced techniques such as WordNet can also be applied to the search engine to improve the accuracy and coverage of querying results and can serve partly as query expansion that expands a search query with its synonym words or semantic relative words.

1.5 Study Purpose

This project is about building a semantic search engine for scientific articles in the fintech field and comparing the performance of using different text embedding techniques. We want to discover and develop a text embedding method that can best describe sentences in FinTech research articles and yield the best result of finding related scientific articles by a citation sentence. This project aims to help academics find proper references for their statement when writing literature reviews and introductions of scientific reports.

Citations are commonly used in writing scientific articles to acknowledge the source of a statement in our work. Though we usually file and organize papers well while doing literature reviews, it happens that some statements are based on months or years of experience or accumulated knowledge in a particular subject. It could be relatively hard work to find an appropriate source for this idea. Hence, we are proposing a method to find the best math articles for our writing. We want to look for articles that mainly describe similar thoughts to the statement we write down in a certain domain in scientific writing. Among all the processes in the natural language processing pipeline, the text embedding techniques would be mostly focused on in this research, which is to convert contents into meaningful numbers and vectors for algorithms to compute, transform and compare. We decide to apply this research in the fintech field and answer the question "What text embedding techniques can best represent a scientific sentence to derive a semantic search engine for the fintech academic articles?"

The detail of this project is recorded in this document. The main issues and goals of this project are stated in the Introduction chapter. Related works, commonly-used background techniques and open-source tools in NLP field are organised and described in the Literature Review chapter. Data collection criteria, preprocessing steps and experiment details are presented in the Material and Method chapter. The performances and analysis of modeling methods in each experiment are listed in the Results chapter. Last but not least, the Conclusion and Discussion chapter stated interesting or notable findings and simple summarisations of this project.

Chapter 2

Literature Review

2.1 Text Embedding

Text embedding methods can be roughly classified by the level of encoding unit (embed in a word, sentence, or article level), supervised or unsupervised, similarity measurement (such as Euclidean distance or cosine similarity) and other specific technique support such as term-based embedding in particular fields and graph-based embedding. Some advanced methods such as ELMo[1] and GPT[2] use autoregressive models and BERT[3] uses an autoencoder to do bidirectional context encoding. A more complex structure of neural networks and a bigger training corpus could derive a better and more robust model but a huge amount of time to train models as well. Hence, another improvement of embedding methods would be the trade-off between better performance and reasonable training time.

Bag of Word & TF-IDF

Bag-Of-Word (BOW) is one of the approaches to implement the vector space model[4] in the NLP field and one of the most straightforward ways to extract information from texts. BOW uses word counts as the representation of texts, not taking the order, grammar and semantic meaning of words into account. A fixed-length vector is used to record the appearance frequency of a word, encoding a word into a sparse vector.[5] BOW can successfully categorise texts[6], while text analysis from other aspects is still possible to be explored.

Term frequency-inverse document frequency (TF-IDF) model uses the word occurrence in sentences and documents to denote the importance of words. TF-IDF assumes a word to play an important role if it occurs frequently only in some sentences as the key idea of this document. An important word will have a higher term frequency and a lower document frequency, which makes the TF-IDF higher. On the opposite, a less important word equally occurs in most of the sentences, generally used in many situations and topics but implying fewer ideas.[7]

Latent Semantic Analysis

Latent Semantic Analysis (LSA), also known as Latent Semantic Indexing (LSI), is a technique in NLP and information retrieval first used in 1990 [8]. LSA applies math techniques, such as singular value decomposition (SVD, $M = U\Sigma V$), to a term-document matrix, such as TF-IDF matrix, to discover hidden patterns in texts and convert the matrix to lower dimensions. LSA decomposes a term-document matrix (M) into a term-topic matrix (U), a topic-document matrix (V^T) and a topic-topic diagonal matrix (Σ) denotes the importance of a certain topic in the collection. LSA is an unsupervised and relatively simple model compared to those neuron-network methods, making it widely used for identifying latent topics, evaluating document similarity and information retrieval.

Word2Vec: Continuous Bag of Word and Skip Gram

Word2Vec is and neuron network-based approach to perform word embedding, introduced by Google in 2013[9][10]. Word2vec encodes the word by the context of the target word and aims to maximise the probability of predicting the context word(s) as the object of the NN model. Word2vec can be implemented in two ways: continuous bag of words model (CBOW) and skip-gram model (SG), which are both 3-layer NN models but require different input and prediction. The 3-layer NN structure includes an input layer, a hidden layer and an output layer along with a softmax function to yield the output probabilities for every unique word in the document to present in the context given the target word. CBOW uses the context-before and context-after to predict the central word, while SG uses the central word to predict its context-before and context-after. Word2vec is a self-supervised algorithm, which means the NN model needs the central word for CBOW and the context words for SG as the expectation of training, while the prediction labels are implied in the content hence we don't have to provide them to the model.

The training of NN model starts from the one-hot embedding of target words as an input vector. The non-linear activation function, usually in the hidden and output layers, is removed to provide a more efficient calculation. The output vector is the size of unique words and is usually very long, making it a huge time-consuming to calculate the denominator of softmax function. Hence, [9] described hierarchical softmax to speed up the process with a binary tree, the Huffman tree, and reduce the time complexity from O(n) to O(log n). Additionally, within a certain window size, only a few words are present in the context of the target word compared to the number of all unique words, which means the expected output vector of target words is highly sparse. The negative sampling had been presented by [11] [12] and was involved to select less number of not-presented candidates (negative samples) to do the softmax with presented candidates (positive samples), described in [10] as an alternative to hierarchical softmax.

Global Vectors for Word Representation

Global Vectors for Word Representation (GloVe)[13] is a co-occurrence matrix-based method, calculated with the sliding window just like the window size of word2vec. The co-occurrence matrix records the frequency of a pair of words present in the same context of a fixed length of words. This matrix is then transformed into a ratio of two probabilities: (1)the co-occurrence probability of a candidate given one target term, and (2)the co-occurrence probability of the same candidate given another target term. For a given candidate word, the ratio matrix denotes a larger value if the numerator term co-occurs with the candidate word more often than the denominator term does, which indicates a more similar semantic of the numerator to the candidate word, and vice versa. If the candidate words have similar relations to both two target words, the probability ratio would be close to 1.

Compare to Bag-of-Word and LSA, instead of using a whole sentence or paragraph, GloVe uses a sliding window to count the frequency of co-occurrence, which brings the distance information of the pair of words to the co-occurrence matrix. Also, compare to word2vec, GloVe choose a non-NN method to keep the structure fast and simple. GloVe and word2vec both use a sliding window to extract context information, but word2vec uses context words and target words as input and target output to train the parameters in the model and GloVe uses context to calculate co-occurrence.

FastText

FastText[14] is famous for its application in document classification tasks, however, it can also be used to embed words. FastText applies character n-grams features to the target word of word2vec model to not only embed a whole word but also subwords, making it possible to deal with out-of-vocab words, rare words and misspellings. Additionally, FastText learns information from subwords including roots and affixes, which usually contains critical information about whole word semantics. Subwords are generated by breaking down the target word n-characters frame, initially embedded and summed up as the input vector of the word2vec model (CBOW or SG) for the target word. Once comes an unseen word, FastText would split it into subwords and look up the training subwords for embedding vectors. The embedding vectors of existing subwords would be summed up as the embedding vector of the unseen word.

Recurrent Neural Network

Recurrent Neural Network (RNN) is to deal with time series data as an input of NN, which contains time-ordered relation in a series of elements and requires a proper structure to handle the relation within elements. Natural language can be treated as time-series data since the meaning of a sentence can sometimes highly depend on the order of words and the meaning of a word can also depend on the context before and after[15]. In a hidden layer, a contextual vector called hidden state is introduced as a memory to transmit information from the former timestep to the latter timestep. The output hidden state at time timestep t $h'(t)$ is generated by input hidden state $h(t) = h'(t-1)$, neuron input at this timestep $x(t)$, two weights W_x, W_h and a bias b_h, while the neuron output at this timestep $y(t)$ is linearly transformed from output hidden state $h'(t)$, a weight b_y and a bias b_y. The information of the former timesteps is involved and considered in the calculation of the current timestep and the current decision can depend on previous data, making good use of time-dependence data. However, the outputs at every timestep have to be calculated one by one, along with the transmission of the hidden layer state, making it difficult to parallelly compute a series of data.

Considering the fact that basic RNN can only pass a hidden state from the former timestep to a latter timestep, while in some situations the information afterwards can also highly relate to the decision-making currently. Hence, the bidirectional recurrent neural networks (BRNN)[16] is performed as a two-way RNN to be involved in the computation at the current timestep. There are two loops and two sets of hidden states in BRNN to transfer information to the previous timestep and the following timestep respectively to provide information before and after.

Long-Short Term Memory

To take the information at further timesteps into account, Long-Short Term Memory (LSTM) introduced cell state to store and transfer long-term memory. Cell state $(c(t), c'(t))$ works like a conveyor belt, puts important memory into records and forgets less-important information along the timesteps. In a hidden layer, the hidden state vector would be concatenated with the input vector. Forget gate, input gate and output gate would then be formed via the concatenated vector, three weights (W_f, W_i, and W_o, separately) and a sigmoid transformation. The concatenated vector is also transformed by the tanh function and multiplied by the input gate to form a new information vector to store in long-term memory. As for the cell state, it is first multiplied by the forget gate to control how much the information would remain in long-term memory, then added by the new information vector to pass to the next timestep. As for the hidden state, the processed cell state is utilised to be transformed by the tanh function and multiplied by the output gate to pass to the next timestep. Finally, as for the output of the current timestep, the processed hidden state is utilised to be multiplied by a weight vector and transformed by the sigmoid function.

Like RNN, LSTM can also be built bidirectionally as biLSTM**graves2013hybrid**. LSTM is widely used in NLP[17], time series analysis[18], image captioning[19], healthcare[20], etc. Especially, LSTM excels in NLP tasks including language modeling[21], sentiment analysis[22] and text generation[23] due to the benefit of dealing with sequence input and taking into account the information far from current word. However, like RNN, it takes more effort and advanced tricks for the implementation of LSTM to be parallelly computed[24].

Embeddings from Language Model

Embeddings from Language Model (ELMo)[1] is one of the most popular applications of LSTM in NLP tasks. ELMo consists of L bidirectional LSTM (biLM) models and yields $L+1$ latent representations of word embedding, one initial input vector and L hidden states vectors h' from each biLM. The output of ELMo is generated by weighting and summing up the $L+1$ latent representations. In the original paper, ELMo obtains initial input vectors from a pretrained character-level word embedding model cnn-big-lstm[25] developed by Google. cnn-big-lstm is trained by a character-level CNN to receive character tokens, a base LSTM model and several small LSTM models to predict target words char-by-char, which enables ELMo to understand the semantics of roots, affixes and other latent meanings. ELMo is famous in NLP tasks including sentiment analysis, named entity recognition, dependency parsing[26], and conference resolution[27].

self-attention

Self-attention is another approach, developed by Google, to deal with time series data and aims to consider neighbour timesteps while calculating the current timestep but can be parallelly computed[28]. Self-attention calculates the relation between the query element (current word) and key elements (context words) with three weights (W_q, W_k, W_v) instead of passing hidden state and cell state in RNN or LSTM. When currently inputting the t^{th} word, The query word vector $(x(t))$ and key word vectors $(x(i), i \neq t)$ would be multiplied by W_q, W_k respectively to generate weighted query vector $(q(t))$ and weighted key vectors $(k(i), i \neq t)$, and then the dot product between weighted query vector $(q(t))$ and each of the weighted key vectors $(k(i), i \neq t)$ would be computed as the relation between the target word and context words $(\omega(i), i \neq t)$. The third weight, W_v, multiplies every input value to generate weighted input vectors $(v(i))$, which are then multiplied by the corresponding normalised relation $(\omega'(i), i \neq t)$ and summed up to yield the output of hidden layer$(y(t))$.

Although it's not technically a bidirectional structure, taking the pairwise relation into account enables self-attention to contextually embed words. Additionally, without passing hidden states and cell states, self-attention can be implemented parallelly. Like RNN and LSTM, the number of neighbour words to pay attention to can be adjusted by passing arguments. In image processing, Convolutional Neural Network (CNN) can also be seen as a special case of self-attention with appropriate parameters[29].

Multi-head self-attention is a widely used extension of self-attention, which splits the weighted query vectors, key vectors and input vectors into several streams, computes relations within the same stream, and concatenates the outputs from different streams. Multi-head attention allows the model to focus on different features from different perspectives within subspaces. Since self-attention estimates the relation of words regardless of the location or distances of words, positional encoding is used to provide position information of series elements to the model, by manually adding a specific matrix such as one-hot or sin function to the input data series to denote the order of serial input vectors.

transformer

Transformer uses self-attention layers to achieve sequence-to-sequence tasks contextually, consisting of two parts: encoder and decoder. Encoder deals with sequence input by self-attention layers, requires a set of initial embedding vectors of words from input sentences, and yields a set of vectors sharing the same length as the input sentence. Encoder's output would then be applied to the second self-attention layer in the decoder. Decoder manages to generate sequence output based on the understanding of the whole input sequence, while the output sequence doesn't have to correspond to the input sequence and might have a different length from the input sequence.

When training a transformer, the decoder has an input sequence as a prediction label and is processed by the first self-attention layer to embed the prediction label words. In the second self-attention layer in decoder, instead of computing relations between each word in the prediction label, they would be treated as query words and the output vectors from encoder would be imported as key words. The relations of word vectors would be calculated between prediction label vectors and input embedding vectors from the encoder parallelly, hence transformers can deal with seq2seq tasks regardless of the word count of neither input nor output sentence. Decoder's output is a set of prediction vectors representing the probability of the corresponding word, and the objective function of a transformer is to maximise the probabilities. However, when generating a sentence from an unknown input sentence, no prediction label is provided. The input and output of encoder remain the same, and the input of decoder would be a special token, ¡START¿, representing the starting point of the output sentence. The output sentence would be generated word-by-word hence not in parallel. Especially, the first self-attention layer in decoder is called masked self-attention since a former word wouldn't be able to compute the relation to a latter word to mimic a real-word seq2seq situation.

Universal Sentence Encoder

Universal Sentence Encoder (USE) is an NN-based end-to-end sentence embedding technique to encode a sentence to a 512-length vector. Instead of generating the sentence embedding vector by taking an average of all of the word vectors in the sentence, USE provides two ways in its original paper to embed the whole sentence. The first approach stacks six transformer encoders to form the encoder of USE, while a transformer encoder consists of a self-attention and a feed forward network. This transformer-based model has the benefit of better accuracy in later tasks such as sentence classification, however, the complicated structure of stacked transformers and several self-attention modules results in the time complexity of $O(n^2)$ in sentence length and consumes more computing time and resources. The second approach utilises Deep Averaging Network (DAN) to avoid the huge computation of self-attention. DAN takes every single word and the word bi-grams as tokens, takes the average of all tokens, and applies four feed forward layers to generate the sentence embedding vector. The word bi-grams serves contextual information to the embedding model but reduces the computation of self-attention models, however, DAN-based model yields a lower accuracy compared to the transformation-based model.

BERT

Sample text sample text sample text sample text sample text

Applications of Text Embedding Techniques

Text embedding techniques are improved for specific objectives, fields and styles and applied to a wide range of tasks, including various issues of tweets analysis[30], visualisation in the biomedical field[31] and sentiment analysis on movie reviews[32].

Khatua et al.[33] identified crisis-related tweets during the 2014 Ebola and 2016 Zika outbreaks with pre-trained Word2Vec and GloVe models. They found a better classification performance to have a small domain-specific corpus from tweets and scholarly abstracts from PubMed participated in the model. They also observed a higher accuracy from a higher dimension of word vector and skip-gram model than CBOW.

Lee et al.[34] utilized SentiWordNet 3.0 to analyse the effect of several negative emotions in hotel reviews. SentiWordNet 3.0[35] provided sentiment analysis as classification tasks and word embedding with a frequency-weighted bag-of-words model and the help of WordNet corpus. Onan[36] presented a sentiment analysis approach to product reviews from Twitter. This deep-learning-based method applied TF-IDF weighted GloVe to the CNN-LSTM architecture to do word embedding and outperformed conventional deep-learning methods.

2.2 Text Similarity

A pair of similar words should act similarly in most of the features we extracted, while a good similarity metric can aggregate the difference in every feature into a single value, which is comparable between each pair of words. Besides Euclidean distance and cosine similarity as semantic similarity measurements mentioned above, WordNet also provides path similarity to evaluate similarities of words with a lexical hierarchical structure.

Sentence similarity measurements are more complex and various compared to word similarity since sentences can be considered as combinations of words. The most straightforward approach is to aggregate words in the sentence as a representation to compare with other sentences, which can be seen as a baseline measurement of sentence similarity. The steps are to take averages of every word in each sentence, yield a single vector for each sentence and calculate the Euclidean distance or cosine similarity between two sentences with average vectors. This approach, however, doesn't consider the order of words, while it can have a huge effect on a sentence's meaning when the word order differs.

Several algorithms are proposed to map a sentence into a vector and calculate the similarity with the derived value directly from the embedding process. Much of them are extended from an existing word embedding method to apply to sentences or even documents, such as Word2Vec, Sent2Vec and Doc2Vec. Some Approaches consider different levels of embedding at the same time, such as word embedding and sentence embedding of BERT.

Lexical Relation and WordNet

Sample text sample text sample text sample text sample text

2.3 Text Embedding for Search Engine

With the increase of documents and web pages, traditional keywords-based search engines are thought to be powerless to correctly look for users' requirements. Text embedding techniques are applied to search engines to provide machine-readable web pages and semantic annotations to the algorithm to yield a more accurate search result. Many websites are applying text embedding to their search engines, including Google Search, Microsoft's Bing Search, Amazon Search and many e-commerce websites and platforms. Not to confuse the search engine using text embedding and the search engine using semantic web search language, in this paper, we refer to the semantic search as the searching approach with text embedding or other semantic retrieval techniques.

Regarding the search engines for scientific articles, Eisenberg et al.[37] proposed a semantic search for Biogeochemical literature that undergoes paper research by comparing the concepts extracted from queries and academic literature. However, the concept extraction component in the workflow of this research is annotated by domain experts, which can be done automatically by NLP approaches mentioned in the future directions chapter. Fang et al.[38] performed a biomedical article-searching approach called Semantic Sequential Dependence Model (SSDM) that combined semantic information retrieval techniques and the traditional SDM model. Word embedding techniques of the Neural Network Language Model(NNLM)[39], Log-Bilinear Language Model(LBL)[40] and Word2vec are applied to the literature corpus to find synonyms by KNN classification algorithm and generate a domain-specific thesaurus. The thesaurus is then utilized to extend query keywords and the SDM played an important role in the combination strategy of the extended keywords for further comparison to documents.

Compare to the keyword embedding approaches, applying sentence embedding or other wider-level embedding techniques to search engines can better preserve information for query but be more challenging at the same time. Palangi et al.[41] managed to embed semantic information at a sentence level by using LSTM-RNN (Long Short-Term Memory - Recurrent Neural Network) framework to do web document retrieval, and compare the summarisation sentence vector and query sentence vector to yield the best searching result.

2.4 Searching-Based Citation Recommendation

Some academic databases come with citation recommendation tools, which provide relevant articles that share the same category with or are similar to our research. Citation recommendation tools would yield different results resulting from not only different algorithms they used but also candidate papers' published journals, impact factors, times being referred to, and the availability if it's open to everyone or subscription-only.

Citation recommendation methods are always built with three main stages: (1) Generate candidate citations from the publication database (2) Create a recommendation list by ranking the candidate citations (3) Evaluate the accuracy of our recommendation system.[42] Text embedding techniques can be included generally in step 1: generation of candidate citation, that is, applying text embedding to filter out candidate citations from the publication database that better relate to keywords provided by users or meet users' needs.

Since the diversity of terms and patterns between scientific articles in different domains, researchers would tend to train a specific model with their domain data or use transfer learning to fine-tune the model parameters. Tshitoyan et al.[43] used modified Word2Vec embedding to successfully capture complex materials science concepts, without any additional chemistry knowledge insertion, and extract knowledge and relationship from scientific literature. Zhang et al.[44] used 15 text representation models, including 6 term-based methods, 2 word embedding methods, 3 sentence embedding methods, 2 document embedding methods and 2 BERT-based methods, to construct an article recommendation system in the biomedical field. They found BERT and BioSenVec, a Sent2Vec model trained on PubMed corpus, outperformed most of the online and offline citation recommendation systems and an improvement in BERT-based methods after fine-tuning to learn users' preferences.

Wang et al.[45] proposed a sentence-level citation recommendation system called SenCite that used CNN to recognise candidate citation sentences and FastText as the word embedding method to extract information from texts, without summarising an article into sentences. They evaluated the performance of SenCite with several evaluation metrics such as modified reciprocal rank, average precision and normalized discounted cumulative gain and human experts verification. The SenCite is shown to outperform most of the embedding methods and yield a comparable accuracy to BERT. The human experts also stated that SenCite provided the best top-1 citation recommendation.

Chapter 3

Material and Method

3.1 Data Collection

We use the Web of Science search engine and database to find the fintech articles and their journals. Reference papers were collected by the WOS search engine with rules listed as follows:

- Topic: "**Fintech**" or "**Financial technology**" or "**digital finance**" or "**electronic banking**" or "**cryptocurrency**" or "**Blockchain**"
- Document Types: **Article** or **Proceeding Paper** or **Review Article**
- Languages: English

We got 66,974 results on 23rd July 2023 from the Web of Science Core Collection and the top 10 publications with the most article are as follow:

Due to the time restriction mentioned in the following section, we utilise the fintech articles from a single Journal *IEEE Access* as the raw data of this research. The fintech articles are exported to `EndNote 20`[1] and the `Find Full Text` feature is used to look for their full text by their Accession numbers including Digital Object Identifier (DOI), WOS ID and PubMed ID. This feature enables us to automatically batch-download loads of reference papers as PDF files if the journal is available in Open Access resources.

[1]https://endnote.com/

Table 3.1: Table 1. The top 10 publications with the most fintech articles.

Journal	# of papers
IEEE ACCESS	1342
SUSTAINABILITY	1207
SENSORS	505
IEEE INTERNET OF THINGS JOURNAL	489
JOURNAL OF CLEANER PRODUCTION	415
APPLIED SCIENCES-BASEL	404
ENVIRONMENTAL SCIENCE AND POLLUTION RESEARCH	377
ENERGIES	374
ELECTRONICS	344
TECHNOLOGICAL FORECASTING AND SOCIAL CHANGE	326

3.2 Data Structuring

We convert the unstructured PDF file into a structured database consisting of pairs of a citation sentence and its full-text reference paper. Firstly, we extract the main content with citations and the bibliography of the articles from PDF files to a plain text file and a dictionary, respectfully. This process is to remove non-organised and less-informed texts, such as page numbers, article titles, journal names, images, math equations and paragraph/section labels, to make sure the sentences in the main content can be fully extracted for further analysis. The bibliography of the articles is also extracted and the information of each reference paper is well-organised to a dictionary. The reference papers extracted from an article are serialised by `pickle`[2] module to a list of dictionaries. The citation sentence can then look up to its corresponding reference paper via the information in this reference dictionary and be matched with the full text. The Python tool `PyMuPDF`[3] is used to obtain the style and location of lines, spans, and sections in a PDF page. The patterns including font size, style and colour are set manually and utilised to recognise if the texts in the current span are the main content, bibliography, or non of both. Hence, the design and layout of a journal would highly affect the criteria of the extraction. We would have to customise the process of data extraction from one journal to another, which takes a lot of time and effort and we're not yet able to program the automatic recognition.

Secondly, The sentences that are inspired by reference papers, called citation sentences,

[2]https://docs.python.org/3/library/pickle.html
[3]https://pypi.org/project/PyMuPDF/

are extracted and organised with the reference number. We split the full texts into sentences and recognise the citation sentences by square brackets with a number in them. The citation sentence and reference number would be organised and serialised into a list. Last but not least, the citation sentence and the reference paper are matched together to form the database for this research. The reference information of the citation sentence can be found via the reference number and further match the citation sentence to the plain text of the reference paper. The citation sentences from all of the articles and their reference texts are saved as a list of dictionaries. The whole dataset is used for unsupervised models, while a 7:3 ratio is applied to split the database into the training set and the test set for supervised models.

3.3 Data Preprocessing

Data preprocessing includes every approach to convert the database to the required input of every embedding model. Among all the preprocessing steps, `unidecode` is priorly applied to texts to convert special characters to the closest English characters, such as *ø*, *ô*, *õ*, *ō*, *ó*, *ǒ*, *ŏ* and ligature such as *æ*, *œ*, *fi*, *fl* and *ff*. Sentence tokenisation is applied to separate the whole article into sentences basically by the period mark. Word tokenisation is applied to sentences that split them into words roughly by space. Stop words are removed from the word tokens and lemmatisation is applied to the word tokens to reduce inflected forms of a word into the word's lemma to treat them as a single word since they contain a similar meaning even if they look different. The above process is implemented by Natural Language Toolkit, `nltk`.

3.4 Experiments

In this research, we design two modules in the workflow to make good use of different kinds of embedding methods, the vectorisation module and the similarity module. The vectorisation module aims to embed a series of words, which can be a sentence or an article, in a word level or sentence level by corresponding embedding algorithms to a single vector. When a word embedding approach is applied to the texts, several word vectors are generated depending on the word count of the texts, and a single embedding vector for the texts would be the average of several word vectors. While a sentence embedding approach is applied, a single embedding vector is generated directly.

The similarity between the citation sentence and its reference article is evaluated as the performance of the embedding method. The similarity module aims to generate a single similarity score for each citation sentence that can reflect either citation sentence-reference sentences similarities or citation sentence-reference article similarity. In the vectorisation module, a reference article can be vectorised sentence-by-sentence, yielding several sentence vectors depending on the number of sentence in the article, or document-by-document, yielding only one embedding vector. When the article is vectorised at a sentence level, several vector similarities are computed by comparing several sentence vectors to the citation sentence and the average of top-10 vector similarities is generated as the single similarity score of the citation sentence. While the article is vectorised at an article level, a single similarity score is generated by comparing between citation sentence vector and reference article vector.

We utilise 8 text embedding algorithms, details listed as follows:

1. TF-IDF

The `TfidfModel` from `gensim` package is used as an unsupervised sentence embedding approach in this research. All of the citation sentences and reference sentences are input together to generate the bag-of-word matrix for further calculation of TF-IDF.

2. LSA

The `TruncatedSVD` from `sklearn` package is used to perform singular vector decomposition on a TF-IDF matrix. The `explained_variance_ratio_` is utilised to decide the number of topics for LSA model. The Bag-Of-Word and TF-IDF matrixes are generated identically in the last section.

3. Word2Vec

The `Word2Vec` from `gensim` package is used to train our financial-specific academic embedding model, serving as a word-embedding approach. All of the citation sentences and reference sentences are input together to CBOW and skip-gram methods. The minimum word count is set to 1 to retain every unique word in our data. The vector size is set to 100 to embed a word into a 100*1 vector. The window size is set to 5 to consider the central word with the context of 5 words before and 5 words after.

4. GloVe

A PyTorch-based package `TorchText` is used to perform the GloVe as a word embedding algorithm with pre-trained word vectors `glove6B`, 100-dimension version. The 6B word vector is trained on *Wikipedia* 2014 corpus and *English Gigaword Fifth Edition*[46], within 6 billion tokens and 400 billion vocabularies.

5. FastText

The `train_unsupervised` function from `FastText` library[4], presented by *Facebook Open Source*, is used to train our financial-specific academic embedding model subword-supporting as a word embedding approach. The parameters detail and input data for both CBOW and skip-gram methods are identical to the previous section *3. Word2Vec.* In this study, we would focus on the word representation tasks, which is the `train_unsupervised`, instead of the text classification part.

6. ELMo

A TensorFlow-based library `simple-elmo` is used to perform word embedding by the pre-trained English ELMo model trained on Wikipedia Dump of October 2019 in a vector size of 1024. The `simple-elmo` implementation is slightly different from the original ELMo paper but the main algorithm remains unchanged and easy to use.

[4]https://fasttext.cc/docs/en/python-module.html

7. USE

A pre-trained, DAN-based text embedding model from TensorFlow Hub[5] is served as a sentence embedding approach provided by *Google*. This `universal-sentence-encoder` is an encoder of greater-than-word length text, applying to sentences, phrases and paragraphs, requires a flexible length of English texts and generates a 512-length vector.

8. BERT

Twp pre-trained variants of BERT are used to serve as sentence embedding approaches, provided on *Hugging Face* as authorised APIs. The Distilbert

3.4.1 Second Level Heading

Sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text.

[5]https://tfhub.dev/google/universal-sentence-encoder/4

Bibliography

[1] M. E. Peters, M. Neumann, M. Iyyer, *et al.*, "Deep contextualized word representations," *arXiv e-prints*, arXiv:1802.05365, arXiv:1802.05365, Feb. 2018. DOI: `10.48550/arXiv.1802.05365`. arXiv: `1802.05365 [cs.CL]`.

[2] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[3] T. Niven and H.-Y. Kao, "Probing neural network comprehension of natural language arguments," *arXiv preprint arXiv:1907.07355*, 2019.

[4] G SALTON, A WONG, and C. YANG, "Vector-space model for automatic indexing," *COMMUNICATIONS OF THE ACM*, vol. 18, no. 11, pp. 613–620, 1975, ISSN: 0001-0782. DOI: `10.1145/361219.361220`.

[5] A. Sethy and B. Ramabhadran, "Bag-of-word normalized n-gram models," in *INTERSPEECH 2008: 9TH ANNUAL CONFERENCE OF THE INTERNATIONAL SPEECH COMMUNICATION ASSOCIATION 2008, VOLS 1-5*, 9th Annual Conference of the International-Speech-Communication-Association (INTERSPEECH 2008), Brisbane, AUSTRALIA, SEP 22-26, 2008, 2008, pp. 1594–1597, ISBN: 978-1-61567-378-0.

[6] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: A statistical framework," *INTERNATIONAL JOURNAL OF MACHINE LEARNING AND CYBERNETICS*, vol. 1, no. 1-4, pp. 43–52, 2010, ISSN: 1868-8071. DOI: `10.1007/s13042-010-0001-0`.

[7] J.-R. Li, Y.-F. Mao, and K. Yang, "Improvement and application of tf * idf algorithm," in *INFORMATION COMPUTING AND APPLICATIONS*, B. Liu and C. Chai, Eds., ser. Lecture Notes in Computer Science, 2nd International Conference on Information

Computing and Applications, Qinhuangdao, PEOPLES R CHINA, OCT 28-31, 2011, Natl Nat Sci Fdn China (NSFC); NE Univ Qinhuangdao; Yanshan Univ; Nanyang Technol Univ, vol. 7030, 2011, pp. 121+, ISBN: 978-3-642-25254-9.

[8] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[11] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," *arXiv preprint arXiv:1206.6426*, 2012.

[12] M. U. Gutmann and A. Hyvarinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *JOURNAL OF MACHINE LEARNING RESEARCH*, vol. 13, pp. 307–361, 2012, ISSN: 1532-4435.

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

[14] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

[15] T. Mikolov, S. Kombrink, L. Burget, J. Černockỳ, and S. Khudanpur, "Extensions of recurrent neural network language model," in *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2011, pp. 5528–5531.

[16] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[17] D. Wang and E. Nyberg, "A long short-term memory model for answer sentence selection in question answering," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Beijing, China: Association

for Computational Linguistics, Jul. 2015, pp. 707–712. DOI: `10.3115/v1/P15-2116`. [Online]. Available: `https://aclanthology.org/P15-2116`.

[18] S. Selvin, R Vinayakumar, E. Gopalakrishnan, V. K. Menon, and K. Soman, "Stock price prediction using lstm, rnn and cnn-sliding window model," in *2017 international conference on advances in computing, communications and informatics (icacci)*, IEEE, 2017, pp. 1643–1647.

[19] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[20] A. Rajkomar, E. Oren, K. Chen, *et al.*, "Scalable and accurate deep learning with electronic health records," *NPJ digital medicine*, vol. 1, no. 1, p. 18, 2018.

[21] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[22] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based lstm for aspect-level sentiment classification," in *Proceedings of the 2016 conference on empirical methods in natural language processing*, 2016, pp. 606–615.

[23] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[24] Z. Yangyang, N. Wei, and W. Meinan, "Research on parallel lstm algorithm based on spark," in *2021 IEEE/ACIS 20th International Fall Conference on Computer and Information Science (ICIS Fall)*, IEEE, 2021, pp. 30–33.

[25] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[26] T. Dozat and C. D. Manning, "Deep biaffine attention for neural dependency parsing," *arXiv preprint arXiv:1611.01734*, 2016.

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

[28] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[29] J.-B. Cordonnier, A. Loukas, and M. Jaggi, “On the relationship between self-attention and convolutional layers,” *arXiv preprint arXiv:1911.03584*, 2019.

[30] Z. Mottaghinia, M.-R. Feizi-Derakhshi, L. Farzinvash, and P. Salehpour, “A review of approaches for topic detection in twitter,” *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 33, no. 5, pp. 747–773, 2021.

[31] N. Oubenali, S. Messaoud, A. Filiot, A. Lamer, and P. Andrey, “Visualization of medical concepts represented using word embeddings: A scoping review,” *BMC Medical Informatics and Decision Making*, vol. 22, no. 1, pp. 1–14, 2022.

[32] S. Sivakumar and R. Rajalakshmi, “Analysis of sentiment on movie reviews using word embedding self-attentive lstm,” *International Journal of Ambient Computing and Intelligence (IJACI)*, vol. 12, no. 2, pp. 33–52, 2021.

[33] A. Khatua, A. Khatua, and E. Cambria, “A tale of two epidemics: Contextual word2vec for classifying twitter streams during outbreaks,” *Information Processing & Management*, vol. 56, no. 1, pp. 247–257, 2019.

[34] M. Lee, M. Jeong, and J. Lee, “Roles of negative emotions in customers’ perceived helpfulness of hotel reviews on a user-generated review website: A text mining approach,” *International Journal of Contemporary Hospitality Management*, vol. 29, no. 2, pp. 762–783, 2017.

[35] S. Baccianella, A. Esuli, F. Sebastiani, *et al.*, “Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining.,” in *Lrec*, vol. 10, 2010, pp. 2200–2204.

[36] A. Onan, “Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks,” *Concurrency and Computation: Practice and Experience*, vol. 33, no. 23, e5909, 2021.

[37] J. D. Eisenberg, D. Banisakher, M. Presa, *et al.*, “Toward semantic search for the biogeochemical literature,” in *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, IEEE, 2017, pp. 517–525.

[38] F. Fang, B.-W. Zhang, and X.-C. Yin, “Semantic sequential query expansion for biomedical article search,” *IEEE Access*, vol. 6, pp. 45 448–45 457, 2018.

[39] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," *Advances in neural information processing systems*, vol. 13, 2000.

[40] A. Mnih and G. Hinton, "Three new graphical models for statistical language modelling," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 641–648.

[41] H. Palangi, L. Deng, Y. Shen, *et al.*, "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 4, pp. 694–707, 2016.

[42] S. Ma, C. Zhang, and X. Liu, "A review of citation recommendation: From textual content to enriched context," *Scientometrics*, vol. 122, pp. 1445–1472, 2020.

[43] V. Tshitoyan, J. Dagdelen, L. Weston, *et al.*, "Unsupervised word embeddings capture latent knowledge from materials science literature," *Nature*, vol. 571, no. 7763, pp. 95–98, 2019.

[44] L. Zhang, W. Lu, H. Chen, Y. Huang, and Q. Cheng, "A comparative evaluation of biomedical similar article recommendation," *Journal of Biomedical Informatics*, vol. 131, p. 104 106, 2022.

[45] H.-C. Wang, J.-W. Cheng, and C.-T. Yang, "Sentcite: A sentence-level citation recommender based on the salient similarity among multiple segments," *Scientometrics*, vol. 127, no. 5, pp. 2521–2546, 2022.

[46] R. Parker, D. Graff, J. Kong, K. Chen, and K. Maeda, "English gigaword 5th edition ldc2011t07," *Linguistic Data Consortium*, 2011. DOI: `10.35111/wk4f-qt80`. [Online]. Available: `https://doi.org/10.35111/wk4f-qt80`.

Appendix A

Supplementary Materials I

Sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text.

Appendix B

Supplementary Materials II

Sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text sample text.