Final Year Project Interim Report

A Word Lexicon Builder Using Neural Word Embeddings

Dáire Murphy - 15441458

Supervisor: Derek Greene

A thesis submitted in part fulfilment of the degree of
BSc. (Hons.) in Computer Science with Data Science

UCD School of Computer Science
University College Dublin

Table of Contents

List of Figures

List of Important Abbreviations Used Within

TM	*Text Mining*
TDM	*Text Data Mining*
KDT	*Knowledge Discovery in Textual Databases*
IR	*Information Retrieval*
NLP	*Natural Language Processing*
IE	*Information Extraction*
CBOW	*Continuous bag-of-words*
LIWC	*Linguistic Inquiry and Word Count*

Abstract

Word lexicons are commonly used to systematically filter and search large text corpora, to isolate all documents that are related to the selected concept of interest. Using a combination of pre trained word embeddings, as well as self-developed word recommendation methods, this project evaluates and compares the word recommendations produced from each of these models. This data is examined to identify the most accurate models to be used and an interactive web interface is implemented, using the purposed word embeddings to create a lexicon recommendation method. This web interface allows for editing of recommended words, improving accuracy of the models for users. Visualizations of the embedding spaces are added to the interface to allow exploration by users of the word embedding models used within this project.

Project Specification

Word embeddings, which refer to a set of language modeling and feature learning techniques in natural language processing algorithms that map words or phrases from a vocabulary to vectors of real numbers, have over the last number of years been used in the creation of word lexicons, a list of words interested to a specific theme or concept, on large text corpora. Word embedding models are trained by applying a neural network to a large text dataset, and such many pretrained models are available on an extreme variety of datasets.

A primary focus of this project was the creation, development and evaluation of several word recommendation methods, along with further comparison and evaluation of these models with other pretrained models. The implementation of these models to an interactive web interface is required to create an online lexicon recommendation method. The student will then be expected to add interactive visualizations to the web interface to allow exploration of the embedding spaces by users

Core:

- Develop a word recommendation method, based on a given seed set and a single word embedding model.
- Perform an evaluation to compare the word recommendations produced using embedding models trained on different text datasets and using different algorithms.
- Implement an interactive web interface for lexicon building, which uses the proposed lexicon recommendation method. This interface should allow users to accept or reject recommended words, in order to produce a more useful final lexicon.
- Develop an “ensemble” word recommendation method, which combines the outputs from different embeddings into a single set of recommendations. Incorporate this new method into the web interface.

Advanced:

- Add interactive visualisation functionality to the web interface to allow users to explore the embedding spaces in more detail.
- Design and conduct a user study to assess the usefulness of the web interface and the lexicons which it produces.
- Perform an evaluation to examine the extent to which embedding algorithms can produce different results when applied to the same data.

Chapter 1: **Introduction**

A lexicon is the vocabulary of a person, language or field, commonly used in the medical profession, and are made up of words or phrases, called lexemes, that have a particular meaning to a group or specific topic. Traditionally, the creation of word lexicons was a hugely intensive manual process, requiring a human to select a small set of highly relevant seed words, which are then perceived as central to the concept. Through the process of trial and error, lexicons could be expanded until the lexicon was satisfactory for the required topic. This was undoubtedly a subpar method of text analysis, due to the time-consuming nature of this process and the ad-hoc selection of seed words combining to make any work difficult to justify and challenging for other researchers to replicate.

Recently, to address these issues, work has been done looking at the use of word embeddings, which are a learned representation for text where words that have the same meaning have a similar representation, for recommending relevant words when building a lexicon [1]. Word embeddings are in fact a class of techniques where words are represented in a predefined vector space as real-valued vectors. Each word is mapped to one vector in this vector space, often tens or hundreds of dimensions. This means words which frequently appear in similar context in the original text will appear close together in the vector space, while words that do not frequent together will be dissimilar.

In this project we will look to train an embedding model using a variety of large text datasets and popular algorithms such as Word2Vec [2] and FastText [3]. Using these, we will implement our embedding model on a new web-based tool for constructing word lexicons in an interactive and efficient manner. As the processes of using word embeddings can produce diverse results, different models will be evaluated and compared, and a method for combining the results of different models into a single set of recommendations will be used when creating the lexicon.

1.1 Motivations

With the exponential increase in the amount of data that we as a race are producing, the methods we use to conduct analysis on this data needs to scale accordingly. The use of lexicons to systematically filter text has a long history. Lexicons are especially helpful when detailing practices that are quite nuanced, requiring a set of terms and meanings beyond ordinary language, such as medical texts, and have been used by researchers to search large text corpora to isolate appropriate texts or documents related to the topic of interest.

This led to my focus on creating a web application, that produces word lexicons in an accurate and efficient manner. With the improvement of technology, the ability to analyse huge amounts of text along with the popularisation of word embedding, I was able to create a tool that produces incredibly apt lexicons on practically any topic almost instantly.

1.2 Project Objectives

In this section I outline the direction of my project with a list of objectives, highlighting what I hope to accomplish from the exploration and analysis of this topic and implementation of my work.

- Obtain a number of appropriate large text datasets of varying size and content.
- Create several different word embedding models from these datasets using a combination of popular algorithms.
- Compare and evaluate these models with already available pre-trained models.
- Develop an ensemble recommendation method, combining the outputs of several methods into a single set of recommendations.
- Implement an interactive web interface for lexicon building, using the proposed lexicon recommendations.
- Conduct a user evaluation on the produced web application

1.3 Summary of Report

This report has been split into three sections: related work, progress to date and future work. The first section, related work, discusses three aspects of the project and looks at works done that closely relate to these aspects. This is followed by my current approach to date, which details all the progress I have made with the project so far, an outline to how I approached each aspect of the project and what data considerations are involved. Lastly, this report will cover any future work intended for this project and propose a plan in which this work will be completed.

Chapter 2: **Related Work**

2.1 Text Mining

Data mining in the process of analyzing large data sets to identify hidden patterns of data and establish relationships to solve problems. One of the prominent areas associated with data mining is *text mining* (TM). Text mining, also known as Text Data Mining (TDM) or Knowledge Discovery in Textual Databases (KDT) [6], is the process of extracting useful information from unstructured textual data through the identification and exploration of interesting patterns. It was in the 1980's when the first applications of TM began to be seen, but with recent advancements in technology, an increased focus and expansion of its applications became commonplace. Text mining employs a wide variety of tasks, in which it is possible to distinguish four different applications [8].

1. Information Retrieval (IR)
2. Text Classification
3. Natural Language Processing (NLP)
4. Information Extraction (IE)

Information Retrieval
While structured data is predominantly managed by a database system, unstructured text data is more difficult to parse through. IR systems are used to associate and retrieve relevant information or documents that match a user's query. Due to the ever increasing amounts of textual data being produced, IR has a number of applications including searching for specific books in large libraries, and more recently, the development of web search engines such as Google. [7]

Text Classification
This process is vital for the correct placement of documents, by it's main themes, into predefined sets of topics. Documents being categorized by computer programs are not seen as actual documents, but more like a "bag of words", a common representation in TM and used as a starting point for many IR approaches [9]. The information inside the documents is almost irrelevant, only the word counts of each word is used to identify the main topics of the document, and combined with a glossary determining which topics are set, the document is classified accordingly.

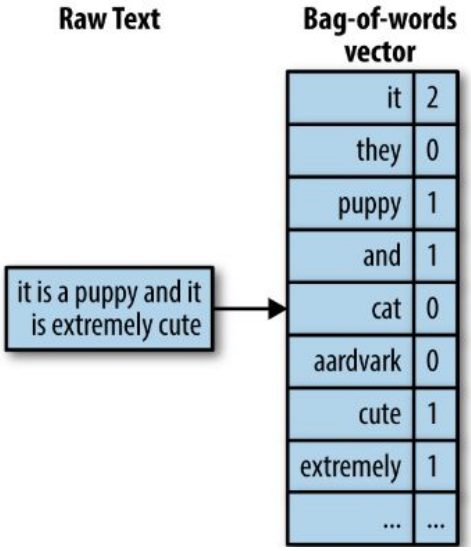


Figure 1: An example of the "Bag-of-words" model for classifying text

Natural Language Processing

NLP is a subfield of linguistics and artificial intelligence that looks at the interaction between computers and natural language text, particularly how to program computers to process large amounts of natural text data. Before text documents can be analysed by computers, the character strings have to be processed, as computers need to be given input in particular format, to help computers understand natural language in a similar way to humans. The reason behind NLPs popularity is that it enables the classification of words into grammatical categories, determine the meaning of a word given its context within a document and generate a complete representation of the grammatical structure of a sentence [10]. This allows the linguistic data of the text document to be extracted

Information Extraction

This method identifies key words and relationships within text documents. In order to be mined, the unstructured data must be converted to a structured, indexed form prior to any query from the user. Once issued, documents related to the query are ranked based on computed similarity to the topic and are presented to the user. There exists numerous methods of how these documents are ranked [11], as well as a number of retrieval strategies.

One retrieval strategy used is the *Vector Space Model*, which like the *Bag-of-Words Model* employs the use of vectors, but while the *Bag-of-Words* produces unigram words to create an unordered list of words to classify the document, the *Vector Space Model* takes the bag of words extracted prior and creates a feature vector for the document where both the query and the document are represented as real numbered vectors in a vector space. If the words of a document can be represented as a vector, its similarity with the query can be measured using a similarity coefficient. Work on this model began as early as the 1960's [12], and with an increase in popularity in the mid 1970's is

still an extremely popular method of computing the similarity between document and query.

2.2 Word Embeddings

The term word embeddings were originally coined by Bengio et al. in 2003 [3], who referred to word embeddings as "distributed representations of words", and used a neural language model along with the model's parameters to train these embeddings. However, it was arguably Collobert and Weston who were the first to demonstrate the power of pre-trained word embeddings in their paper "A unified architecture for natural language processing" [4], back in 2008. This paper established word embeddings as useful tool for downstream task as well as introduces neural network architecture that forms the basis for several future works, and foundation for many current approaches. Initial word embedding models consisted of feed-forward neural networks that would take words from a vocabulary as input, embeds them as vectors into a lower dimensional space, uses through back-propagation as fine tuning, yielding word embeddings as weights of the first layer, known as the Embedding layer.

This all lead to the eventual popularization of word embeddings in 2013, when Milolov et al. produced their paper "Efficient estimation of word representations in vector space" [2], along with the creation of *word2vec*. The primary difference between word2vec and the proposed neural language model by Bengio et al. 2006 is its computational complexity. This would explain why it took until 2013 for this jump in word embeddings popularity, as the increase in cheap computational power made the creation of models such as Word2Vec and GloVe [16]. Partially, the reason behind word2vec's popularity was Mikolov et al. recommended two architectures for learning word embeddings that cut computational costs, Continuous bag-of-words (CBOW) and Skip-Gram.

CBOW in unlike a language model that only bases its prediction on past words, as those models are assed on predicting the next word in the corpus. This model uses both *n* words before and after the target word to predict the output. Rather than receiving just the previous words in the model, a section of words *n* around the target word are used. The second architecture purposed; Skip-gram, uses the logic of CBOW in reverse. Skip-gram uses the center word to predict the closest surrounding words. In actuality, it sums the log probabilities of the surrounding *n* words to the left and right of the target word. In a later paper within the same year [5], Mikolov et al. manage to improve greatly on the speed and accuracy of these models using additional strategies.

The vectors obtained by adding or subtracting words in a vector space will often provide the expected outcomes that a human would give, provided with the same set of words. For example, if the vector of the word "man" is subtracted from the vector "king", combined with the addition of the vector "woman", an outcome of "queen" would be produced by any appropriately trained model.

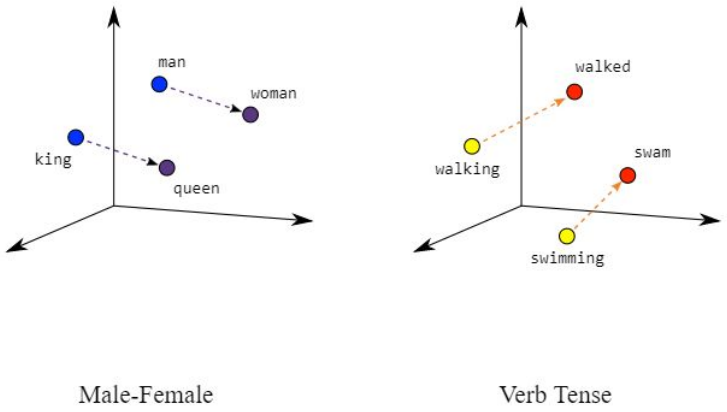


Figure 2: An example of the relationship between words in a word vector space.

2.3 Lexicon Building

In general, building a lexicon for a particular concept requires significant human effort, and only a small number of human-generated concepts have been available, with usually a limited number of keywords contained in each. We now live in a world that produced more textual data daily than ever before. It is impossible to comfortably analyze any domain of text, be it medicine, politics, or finance.

High quality lexicons allow for a detailed analysis of language at a huge scale and across a broad range of signals. Academic researchers searchers often use text analysis via dictionary category methods such as Linguistic Inquiry and Word Count (LIWC) [13], which is an extensively validated dictionary that offers 62 syntactic, topical and emotional categories. LIWC is often used for the analysis of social media posts to count words in lexical categories such as *positive* or *negative emotions*, *sadness* and *health.* However, LIWC, like other popular lexicons is relatively quite small, containing only ~40 emotional categories, many with limited words in these categories.

Work has been done to address these issues, particularly *Empath* [14], "a living lexicon mined from modern text on the web". Empath uses a combination of NLP techniques with the addition of handmade lexicons. Like LIWC, Empath's contents are validated by humans, unlike other machine learning models, but allows users to generate and validate new lexical categories on demand, which LIWC lacks. Given user selected key words such as "fight" or "brawl", Empath produces semantically related target words, such as "violence" in this particular example. It contains 200 pre-validated emotional and topical categories to analyse text across, and uses a combination of deep learning and microtask crowdsourcing to generate and validate its word classification dictionaries. When compared with LIWC, a gold standard lexicon that has been psychometrically validated, across a mixed-corpus dataset a correlation of >.9 is observed.

However, further work done by Park et al. claimed that blindly applying pre-built lexicons for document analysis, without considering the content and keyword usage patterns, can lead to the misunderstanding of the content of said document. This motivated the creation of *ConceptVector* [15], a visual analytics system "that seamlessly integrates a user-driven lexicon-building process with customized document analysis in a highly efficient and flexible manner". During lexicon creation, it allows users to tag recommended words that are unrelated to the topic under consideration as irrelevant to the topic being researched, clarifying the lexicon by weakening the relevance of those words.

ConceptVector allows for the definition and construction of bipolar concepts. Provided with two corresponding yet opposing sets of seed words (e.g. positive vs negative sentiments), two models are created, allowing for comparison.

With the recent popularization of word embeddings such as Word2Vec [2] as well as other automatic text analysis methods, has led to a huge increase in sentiment analysis, documentation summarization and probabilistic topic modeling.

These methods, along with the increasing availability of digital collections of historical literature provided Leavy et al. the motivation to produce Curatr: A Platform for Semantic Analysis and Curation of Historical Literary Texts [1]. In this paper, an online platform called *Curatr* is presented. Its purpose is for "the exploration and curation of literature with machine learning-supported semantic search, designed within the context of digital humanities scholarship."

Having been trained on a large corpus of over 35000 English language digital texts from the 18th and 19th century, *Curatr* provides a text mining workflow that combines neural word embeddings, with expert domain knowledge to generate thematic lexicons of relevant topics to researchers. As mentioned, this platform makes use of word embedding models, specifically *word2vec*, over more complex language models due to the lack of structure in the corpus used to train the model, as well as OCR errors introduced in digitization of the text. To specify, the exact model used on this platform is a 100-dimensional Continuous Bag-of-Words (CBOW) *word2vec* model.

From this model, the top 20 most similar words to the given seed word are given as recommendations to the current lexicon. *Curatr,* contains a 'human in the loop' aspect, as the user must choose which of the recommended words are added to the conceptual lexicon, ensuring that the generated lexicon is informed by the domain of the user. Multiple iterations of this search leads to lexicon refinement, which accumulates in finalized lexicons used as a basis for volume retrieval from the indexed library corpus.

Chapter 3: **Data Considerations**

3.1 Initial Datasets

A primary objective of this project is the creation, evaluation and comparison of several word embedding models. To begin my research, I acquired two pre-trained models, the first being a small Wikipedia dump, just 12MB. This size of a model was chosen to highlight the ineffectiveness of models trained on limited datasets. Following this, another pre-trained model, this time a very large model of 1.5GB trained by Google. It includes word vectors for a vocabulary of 3 million words and phrases that was trained on ~100 billion words from a Google News dataset. The vector length is 300 features, which is much larger than the typical vector dimension used in Word2vec models.

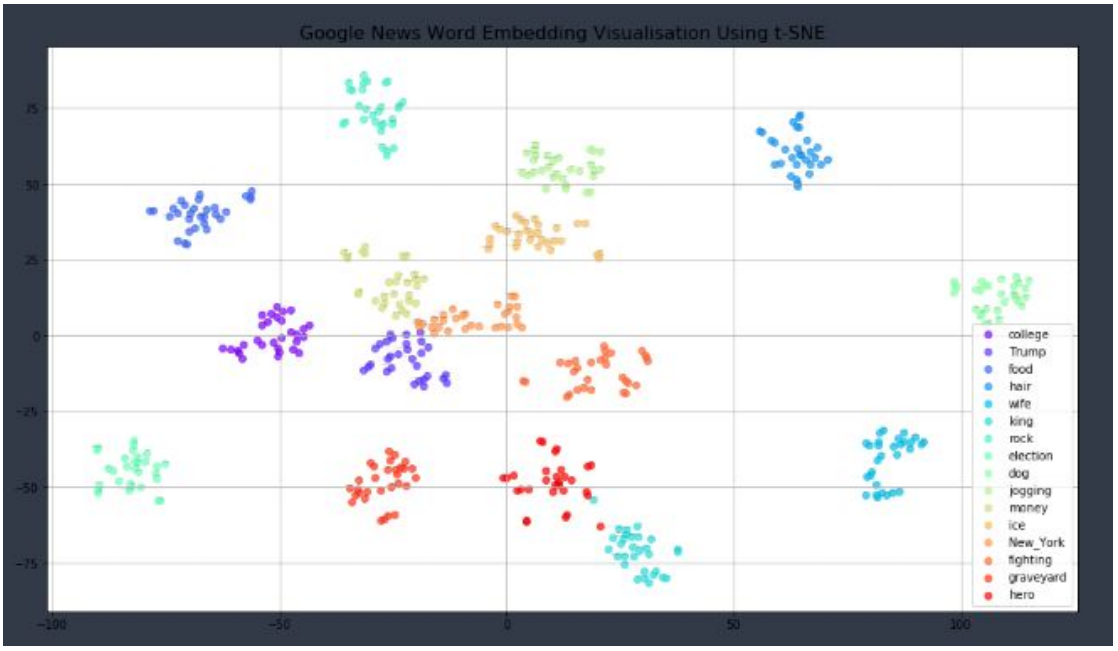


Figure X: Visualisation of the high-dimensional Google News Word2Vec word embeddings using t-SNE[19].

A dataset containing over 500000 Reddit articles, chosen from the "News" subreddit, was my initial choice for creating a self trained word embedding model. Having cleaned and removed unwanted columns of the csv, the target column containing the news articles was tokenized and trained using Word2Vec. This dataset was selected due to its perceived middle ground in size between the two previous pretrained models.

3.2 Future Data

Further datasets are being considered for the training of larger word embedding models, likely two similar datasets to highlight how embedding models trained on even slightly different datasets can produce diverse results (e.g. Two opposite politically leaning news corpora)

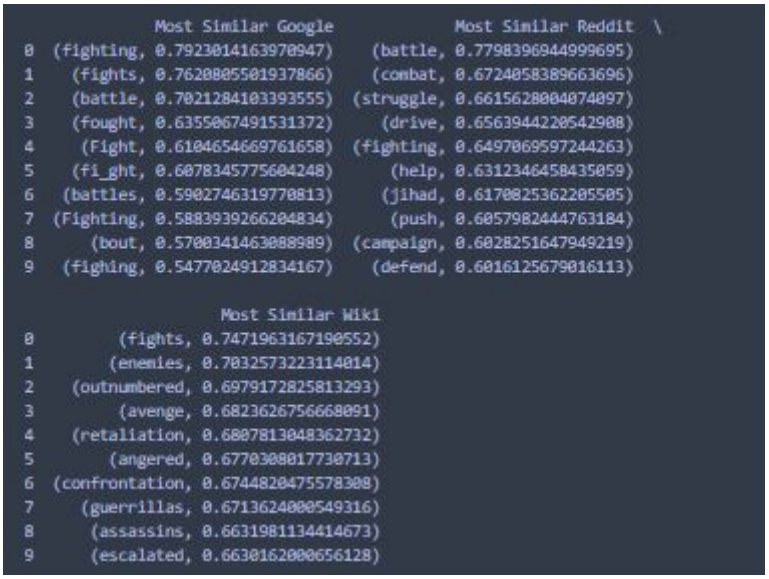
Chapter 4: **An Outline of My Initial Approach**

The objective of this project is to develop a new web-based tool for constructing word lexicons in an interactive and efficient manner. To achieve this goal, the project was broken up into smaller more specific tasks to be tackled.

4.1 Creation and Evaluation of Word Embedding Models

Firstly, the development of a word recommendation method, based on a given seed set and a single word embedding model. Python was the programming language of choice and along with the use of the Genism [17], a popular NLP package for the language, the creation of word embedding models was achievable. Acquiring another, much larger pretrained model as well the creation of a self made word embedding model was needed to perform an evaluation to compare the word recommendations produced using embedding models trained on different text datasets and using different algorithms.

Using several datasets, all initial models were trained using Word2Vec, the dimensional complexity of the vectors used will vary from model to model to provide a wide variety of examinable models of various usefulness. Having created and evaluated a number of models, the addition of several pretrained models of various sizes and origins will be compared against the models created. After comparison, weaker models that provided poor word recommendations were dropped from future analysis.



	Most Similar Google	Most Similar Reddit	Most Similar Wiki
0	(fighting, 0.7923014163970947)	(battle, 0.7798396944999695)	(fights, 0.7471963167190552)
1	(fights, 0.7620805501937866)	(combat, 0.6724058389663696)	(enemies, 0.7032573223114014)
2	(battle, 0.7021284103393555)	(struggle, 0.6615628004074097)	(outnumbered, 0.6979172825813293)
3	(fought, 0.6355067491531372)	(drive, 0.6563944220542908)	(avenge, 0.6823626756668091)
4	(Fight, 0.6104654669761658)	(fighting, 0.6497069597244263)	(retaliation, 0.6807813048362732)
5	(fi_ght, 0.6078345775604248)	(help, 0.6312346458435059)	(angered, 0.6770308017730713)
6	(battles, 0.5902746319770813)	(jihad, 0.6170825362205505)	(confrontation, 0.6744820475578308)
7	(Fighting, 0.5883939266204834)	(push, 0.6057982444763184)	(guerrillas, 0.6713624000549316)
8	(bout, 0.5700341463088989)	(campaign, 0.6028251647949219)	(assassins, 0.6631981134414673)
9	(fighing, 0.5477024912834167)	(defend, 0.6016125679016113)	(escalated, 0.6630162000656128)

Figure X: A comparison of 3 different models produced from the seed word "fight", outputting both the top 10 most similar words and their vector accuracy.

4.2 Development of Ensemble Word Recommendation Method

To create the most relevant lexicons possible, the development of an ensemble word recommendation method is required. This method will combine the outcomes of a number of the previously selected embeddings into a single set of recommendations. To achieve this, we will look to the combining of ranked lists, of which there is a number of methods. This combination of word embeddings will then be the primary model by which the web app will produce its lexicons.

4.3 Implementation of Web Application

With the creation of word embedding model complete, the focus of the project will shift to the development of the interactive web app, which will likely be implemented with the use of the Flask [18], a Python web framework. A number of features will be added to the lexicon building web interface to add to the user experience, including the aforementioned interactive visualisations of the vector spaces, as well as the ability of users to accept or reject words in order to produce more useful final lexicons.

Chapter 5: **Future Work**

5.1 Project Work Plan

This section will discuss my plans on future work for this project. A number of tasks have already been selected in an aim to expand my project from the work I have completed thus far. Firstly a look to expand the number of visualisations present in my work, giving a clearer comparison of different word embedding models. Having completed this, focus will change to looking for the best method of combining the output lists given from different models on the same target word to create a combined ranked list to be used by lexicon recommender in the future. Finally, I will begin working on the implementation of an interactive web interface for lexicon building. The web tools aim, aside from producing accurate lexicons on any given topic, is to allow users to explore interactive visualisations of the different embedding models as well as accept or reject recommended words produced by the lexicon.

Before beginning expansion of my project, I hope to increase the amount of datasets being used, and potentially using new embedding algorithms such a GloVe. I feel a larger number of models being compared, and ultimately joined together will produce more appealing word recommendations.

5.1.1 Visualisation of Word Embedding Models

Having produced some visualisations of my current work already, a look to further the visual aspect of this project with an increased focus on comparison of competing word embedding models created from a variety of datasets. Highlighting the differences between two trained models from seemingly similar datasets could produce many interesting results, particularly when in relation to supposedly unbiased rival news organisations in the United States, such as a comparison between a model trained from a collection of Fox news articles and a model trained from CNN articles.

For visualising embeddings, I have several different options to choose from, including t-SNE [19].

5.1.2 Combining of Ranked lists

Having created a number of word embedding models, comparing and evaluating these models with already available pre trained models and finding which models work best, we aim to create lexicons using a combination of the most effective models. To achieve this, I will be looking at a number of methods for combining ranked list. A method of interest in completing this task is Rank aggregation (RA). RA is the process of combining ranked lists into a single ranking, with these methods being broken into three categories: distributional-based, heuristic and stochastic optimization. Li et al. [5], looks at these methods, characterises various types of lists, and evaluates the performance of said methods in a paper which will be a keystone to any future work I do with ranked aggregation.

5.1.3 Development of Interactive Web Interface

One of the primary aspects of this project is the implementation of a user friendly web based tool for the construction of word lexicons in an interactive and efficient manner. There are a variety of ways to implement a web app from Python, with *Flask* being the most likely candidate for this project due to in common situations the equivalent Flask web application is more explicit than other Python web application frameworks such as *Django*.

The ability of the user to accept or reject words produced by the model is key, to ensure that the lexicon is in the domain of the user. Additionally, a focus on including interactive visualisations for the user to explore the embedding spaces of the models they are using is required. This increased functionality will improve both user understanding of the lexicon builder, and allow for the use of different models to produce lexicons upon the users selection.

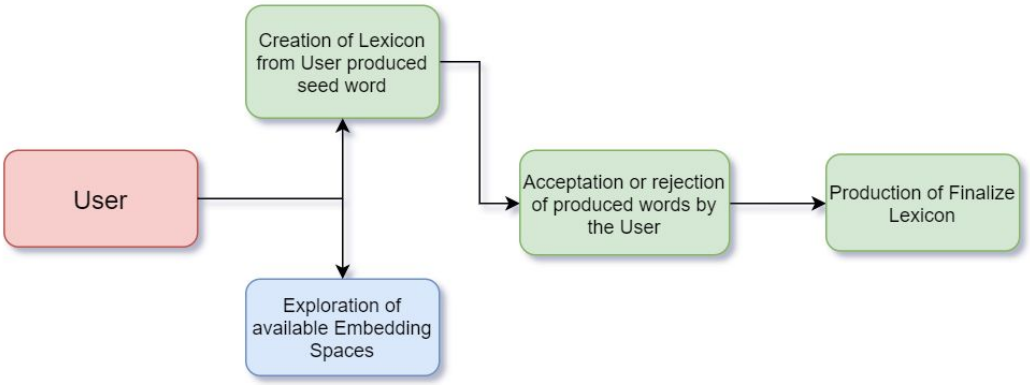


Figure .: A simplified workflow chart of the proposed web application

5.2 Evaluation of Web App

Evaluation is a key aspect of any produced work, which is why an aim for this project is to design and conduct a user study to assess the usefulness of the web interface and the lexicons it produces. To accomplish this, the study will likely consist of an explanation of the web application, followed by an example use of the system by whoever is running the study. The user will then be asked to create a lexicon from seed word of their choice. Following the creation, the number of words rejected from the initial list of recommended words will be monitored, as well as the users overall satisfaction in the lexicon itself. This process will then be repeated several times to evaluate the effectiveness and appeal of the web tool to each user. With enough users partaking in the study, as well as an increase in important questions, this method should produce a valid evaluation of the system.

5.3 Gantt Chart

Below, in Figure X, depicts a Gantt chart of the future work plan for this project. An initial start date is estimated to be on the 27th of December

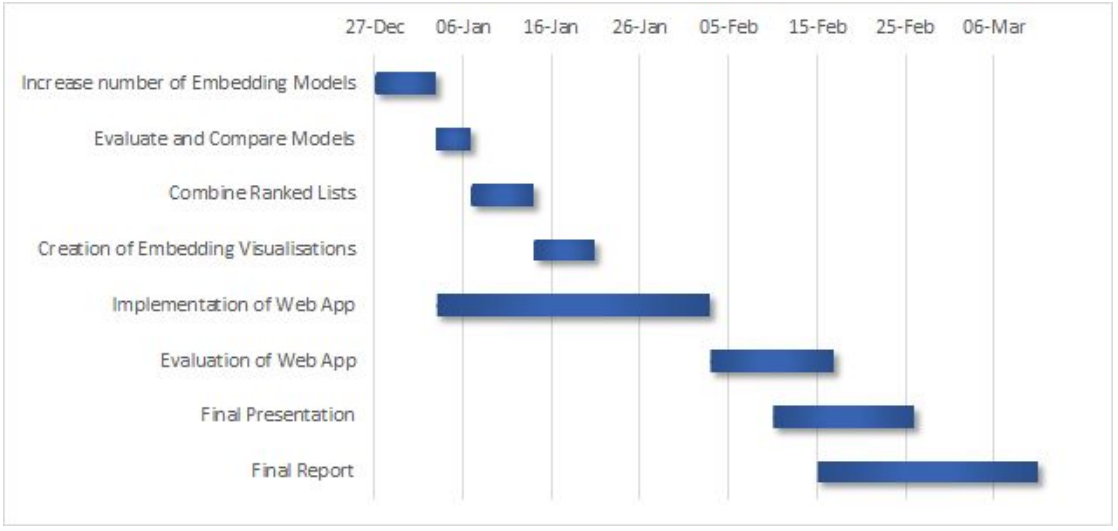


Figure X: Gantt Chart depicting predicted future work plan for project

References

1. Leavy, S., Meaney, G., Wade, K., Greene, D. Curatr: A Platform for Semantic Analysis and Curation of Historical Literary Texts. Proc. 13th International Conference on Metadata and Semantics Research (2019).
2. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
3. Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A Neural Probabilistic Language Model. The Journal of Machine Learning Research, 3, 1137–1155. http://doi.org/10.1162/153244303322533223
4. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. NIPS, 1–9.
5. Xue Li, Xinlei Wang, Guanghua Xiao (2017) A comparative study of rank aggregation methods for partial and top ranked lists in genomic applications. *Briefings in Bioinformatics*, Volume 20, Issue 1, January 2019, Pages 178–189, https://doi.org/10.1093/bib/bbx101
6. Feldman, R. & Dagan, I. (1995) Knowledge discovery in textual databases (KDT). In proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95), Montreal, Canada, August 20-21, AAAI Press, 112-117.
7. Vishal Gupta and Gurpreet S. Lehal, A Survey of Text Mining Techniques and Applications, JOURNAL OF EMERGING TECHNOLOGIES IN WEB INTELLIGENCE, VOL. 1, NO. 1, AUGUST 2009.
8. Vijayarani, S., Ilamathi, M. J., and Nithya, M., 2015. Preprocessing Techniques for Text Mining: An Overview. International Journal Computer Science and Communication Network, 5, 7-16.
9. M.Radovanovic, M. Ivanovic, "Text mining: Approaches and applications," vol. 38, no. 3, pp. 227-234,[Online]. Available:http://www.emis.de/journals/NSJOM/Paper s/383/NSJOM 3[Accessed: December 2013].
10. JISC (2008). Text Mining Briefing Paper. Joint Information Systems Committee. Retrieved from: http://jisc.ac.uk/media/documents/publications/ bptextminingv2.pdf
11. D.A. Grossman and O. Frieder, Information Retrieval: Algorithms and Heuristics. 1998.
12. Salton, G., and Lesk, M. 1971. Computer evaluation of indexing and text processing. Prentice Hall, Ing. Englewood Cliffs, New Jersey. 143–180
13. James W Pennebaker, Martha E Francis, and Roger J Booth. Linguistic inquiry and word count: LIWC 2001. Mahway: Lawrence Erlbaum Associates 71 2001.
14. Fast Ethan, Chen Binbin, Bernstein Michael S. Empath: Understanding Topic Signals in Large-Scale Text. CHI.2016.
15. D. Park, S. Kim, J. Lee, J. Choo, N. Diakopoulos, and N. Elmqvist, "ConceptVector: Text visual analytics via interactive lexicon building using word embedding," IEEE Trans. Vis. Comput. Graph., vol. 24, no. 1, pp. 361–370, Jan. 2018

16. J. Pennington, R. Socher, and C. D. Manning. GloVe: Global vectors for word representation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1532–1543, 2014
17. Řehůřek, R., & Sojka, P. (2011). Gensim - Statistical Semantics in Python
18. A. Ronacher (2010) Flask, https://www.fullstackpython.com/flask.html
19. van der Maaten, L. & Hinton, G. E. Visualizing data using t-SNE. *J. Mach. Learn.Research* 9, 2579–2605 (2008).