A Word Embedding Model Learned from Political Tweets

Noufa N. Alnajran, Keeley A. Crockett, David McLean, Annabel Latham
Department of Computing, Mathematics, and Digital Technology
Manchester Metropolitan University
Noufa.alnajran@stu.mmu.ac.uk,{k.crockett, d.mclean, a.latham}@mmu.ac.uk

***Abstract*— Distributed word representations have recently contributed to significant improvements in many natural language processing (NLP) tasks. Distributional semantics have become amongst the important trends in machine learning (ML) applications. Word embeddings are distributed representations of words that learn semantic relationships from a large corpus of text. In the social context, the distributed representation of a word is likely to be different from general text word embeddings. This is relatively due to the unique lexical semantic features and morphological structure of social media text such as tweets, which implies different word vector representations. In this paper, we collect and present a political social dataset that consists of over four million English tweets. An artificial neural network (NN) is trained to learn word co-occurrence and generate word vectors from the political corpus of tweets. The model is 136MB and includes word representations for a vocabulary of over 86K unique words and phrases. The learned model shall contribute to the success of many ML and NLP applications in microblogging Social Network Analysis (OSN), such as semantic similarity and cluster analysis tasks.**

***Keywords*–** Word Embedding, Language Modelling, Deep Learning, Social Network Analysis, Twitter Analysis

I. Introduction

The concept of "word embedding" is based on the linguistic distributional hypothesis that words occurring in similar contexts tend to have similar meanings [1]. However, the curse of dimensionality have always been a fundamental issue in most language modelling and learning representations. High dimensionality usually require thousands or millions of dimensions for sparse word vectors [2], which experience memory latencies when traversing the sparse data structures. Word-embedding models are less prone to this problem as they are generally composed of dense continuous-valued vector representations. These representations are produced such that vectors that are closer to each other in the vector space should represent words with similar meanings. Conventional word frequency models such as bag-of-words (BOW) fail to capture the semantic distances between words. That is, words *write*, *draw* and *drive* are considered equally distant despite the fact that *write* is semantically less distant to *draw* than it is to *drive*. The training iterations in the neural embedding model updates the context entries in the words' associated embedding vectors, which leads to the pre-trained model recognizing the little semantic distance between *draw* and *write* in the vector space.

Word embedding models have shown significant improvements in the performance of many NLP applications such as sentiment analysis [3-5] and text classification [6-8] and recommendation [9]. However, the lack of neural embedding models trained on social corpora has led microblogging computational linguistic related tasks to use embedding models trained on general data. These models represent words vectors according to the appearance of the words in a more formal context compared to their colloquial use in social contexts, such as Twitter. Tweets have unique lexical and structural features that are different from general English texts found in traditional documents. Out-of-vocabulary (OOV) words are prevalent in tweets, which are not found in models trained on general text corpora. The user generated content found in microblogging OSN, particularly Twitter, is usually a fertile environment for noise and common user conventions and emoticons. The informal nature of this social medium and the character limit restriction lead people to cut off conjunctions, pronouns, and substitute expressive terms with emoji in order to, ultimately, use the allowed range of characters in delivering the intended meaning. These special features of short texts posted in microblogs require NLP applications to have embeddings that model the behavior of words used in the social context.

This paper presents the methodology and training of a political word embedding model learned from a corpus of over four million political tweets. Politics is an active domain in Twitter and a rich source of controversial views. The EU Referendum event that took place on 23rd of June 2016 was targeted for data collection. The dataset not only included political news tweets and tweets related to politicians, but also daily chitchat on people's views and expectations on the event of "Brexit". However, vectors that are generated from raw tweets generally exhibit lots of noise and introduce inaccuracies to target applications. Therefore, due to the high level of noise and redundancy in Twitter, the collected tweets underwent several pre-processing stages in order to construct a rich corpus of positive examples, from which an accurate embedding model can be generated. In this research, the authors aim to train a model to learn embeddings not only for unigrams (i.e. single tokens), but for bi-grams (two-word phrases) as well. Phrases are commonly observed as hashtags in Twitter, particularly in the domain under consideration. Therefore, it is important to learn embeddings for these phrases such as *EU referendum*, *vote leave*, *stronger in*, etc. instead of each word separately. Towards detecting possible bi-grams, this research computes the probabilities of words occurring together using the Chi-squared test.

In the proposed embedding model, the iterative learning process is computed based on implementing a single hidden layer NN that generates word vectors encoding linguistic regularities and patterns, which can be represented as linear translations. The major contributions of this paper are demonstrated as follows:

1. Streaming real-time tweets in the political domain and constructing a preprocessed corpus of over four million tweet and 12.3 million words and phrases.
2. Generating a word embedding model that is learned from the constructed corpus, which shall be useful for different computational linguistic and NLP tasks in the context of microblogging social media posts, particularly tweets.

The rest of the paper is organized as follows: Section II presents related work in the field, Section III describes methods used in this study, Section IV represents the data collection and corpus construction methodology, Section V demonstrate the learned embedding model. Finally, the conclusion and future work are provided in Section VI.

II. Related Work

Language modelling and word embedding models have become a subject of intense discussion. Previous work have been investigating the significance of dense vector models in reducing the curse of dimensionality and improving the performance of NLP applications. In this section, we review the related work that were conducted in this field and discuss limitations and potential research extensions.

An early language model have been proposed by Bengio, Ducharme [2] and Schwenk and Gauvain [10]. The authors proposed a neural embedding model that estimates the probability of a word based on a context window of previous words in a sentence. The model estimates conditional probabilities of words in order to learn: 1) a distributed representation for each word, 2) the probability function for word sequences using a corpus of over 1 million examples. Collobert, Weston [11] introduced C&W deep learning model based on a convolutional neural network (CNN). The CNN learns word embedding vectors based on the syntactic contexts of words. Towards a generalizable model that can handle a number of NLP tasks, the authors performed unsupervised training on the entire Wikipedia corpus, which contains about 631 million words. Although these approaches represent vectors with less dimensionality than one-hot encoding, there is large room for model improvement in term of scalability and computational efficiency.

The revolution of digital user generated content in the era of big data has contributed to further implementations of language models. These neural embedding models have improved the learning speed and capacity in order to handle corpora with thousands of millions of words. Mikolov, Chen [12] introduced a Word2Vec model representing words as real-valued vectors. Word2Vec can have two training architectures, 1) the continuous bag-of-words (CBOW) and 2) the Skip-gram model. Based on the Skip-gram model, the authors published a pre-trained model on a Google News corpus. This model contains 300-dimention vector representations that capture both syntactic and semantic word relationships for the 1 million most frequent words in that corpus. On the other hand, the Global Vectors for Word Representation (GloVe) [13] is an extension to the Word2vec model, which rather than using a window to define local context, GloVe uses a statistical computation across the entire corpus in order to construct an explicit word co-occurrence matrix. Word2vec and GloVe have demonstrated better performance than traditional embedding models such as LSA in the field of topic segmentation [14]. Furthermore, compared to GloVe, Word2Vec produces better word vector representations with a small dimensional semantic space.

In terms of embedding models generated from microblogging social media posts, there is not much research conducting in this area. Tang, Wei [4] extended the word embedding model presented by Collobert, Weston [11]. The authors developed three neural networks to effectively capture sentiment-specific word co-occurrences learned from tweets. The artificial NN are trained through incorporating the sentiment information into the networks' loss functions. The training was performed on a corpus of distant-supervised five million positive and five million negative tweets with emoticons. The effectiveness of the model was demonstrated by using it as a feature in a sentiment classification task. The evaluation was performed using the benchmark dataset of SemEval-2013 [15] and verified by measuring sentiment lexicon similarity. Li, Shah [16] presented several embedding models trained on tweets as well as general text corpora. The authors trained NN models on both raw and pre-processed tweets and demonstrated that the latter generally performs better in tasks related to tweet semantic topic identification. The models were extrinsically evaluated on two tasks, which are sentiment analysis and topic classification. Results show that combining tweets and general texts improves the word embedding quality in terms of the topic classifier performance.

It has been observed from the literature around word embedding models in microblogging OSN that there is a lack of pre-trained models learned from domain-specific social media text corpora. This paper presents the training process and methodology of a NN embedding model that generates real-valued vectors from a corpus of political controversial tweets.

III. Methods

This section briefly describes the general methodology undertaken towards building the word embedding model learned from the Twitter-based corpus under consideration.

A. *Data Collection and Storage Layer*

This layer involves setting up the Twitter Streaming API and its configuration on the political domain for data collection. The streamed tweets are stored in Mongo DB NoSQL database on the flow. That is, in a real-time mode rather than storing them to an external file and transferring them to Mongo DB in batches afterwards.

B. *Corpus Manipulation Layer*

The input to this layer is the raw tweets obtained from the previous layer. Corpus manipulation includes pre-processing steps including n-gram identification and corpus annotation.

C. *Neural Embedding Layer*

In this layer, the actual training of the word embedding model is performed on the pre-processed and annotated corpus.

The goal is to learn the weights of the neural networks hidden layer, which are actually the distributed word representations.

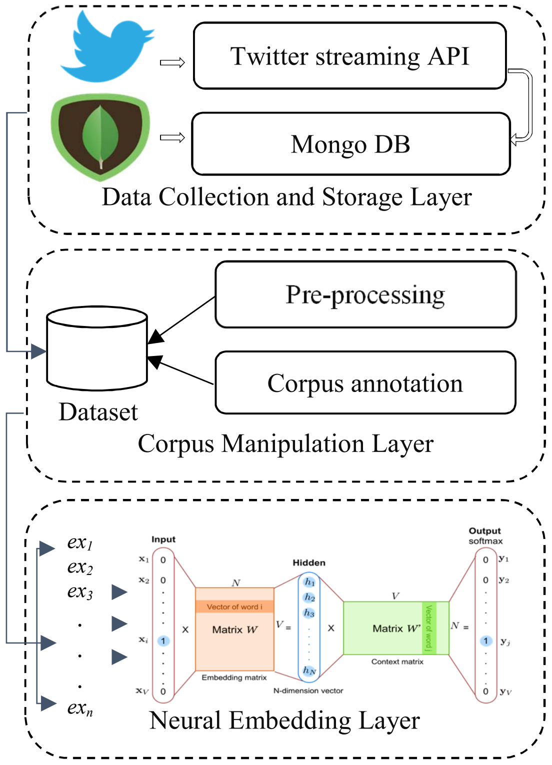

Fig. 1. Layers of the Twitter-based word embedding model framework

Figure 1 shows a hierarchical representation of the layers in the proposed model's training process. The processes undertaken in each layer and the training configurations are further elaborated in the subsequent sections.

IV. Building the Word Embedding Dataset

In this section, the methods used for building the Twitter-based political corpus, through data collection, harvesting and cleaning, where tweets undergo several pre-processing stages before getting into the learning iterations are described.

A. The Domain

In this study, the political domain of the EU Referendum is considered as it has been an active trend in OSNs and a rich source of controversial views. The United Kingdom European Union Membership (known as *EU Referendum*) took place on June 23, 2016 in the UK. Based on a voting criteria, the voters were exposed to two opposing campaigns supporting remaining or leaving the EU. Three months prior to the day of the referendum, the data collection process has commenced using the Twitter Application Programming Interface (API), and continued until one month past that day.

B. Data Collection

The Twitter streaming API allows for establishing a connection and continuously streaming real-time tweets according to a predefined set of search terms. Communicating with the Twitter platform was made possible via the Open Authentication (OAuth) mechanism. This mechanism requires an application registration on the Twitter platform beforehand. Kumar, Morstatter [17] provided a comprehensive overview of the authentication process required by the Twitter API. Amongst various programming languages that interface with the API, Python has been used for its flexibility and prebuilt selection of Twitter software packages and NLP libraries.

Twitter streamed instances are returned as JavaScript Object Notations (JSON) data structures, which are composed of multiple metadata per tweet. These JSON objects were stored in a NoSQL database called MongoDB [18]. MongoDB is used as it is a fully scalable non-relational database, intended for storing unstructured data, such as text, as documents instead of tuples in tables. It has been trusted by several web 2.0 big data sites such as Foursquare, Disney Interactive Media Group, The Guardian, GitHub, and Forbes [18]. The entire 1.2TB text corpus of Wordnik online social dictionary [19] is also stored in over five billion MongoDB records. In this study, the documents inserted into MongoDB are the tweets JSON objects that were retrieved by Twitter API.

```
client = MongoClient('localhost', 27017)
db = client['twitter_db']
collection = db['twitter_collection']
tweet = json.loads(data)
collection.insert(tweet)
```

Fig. 2. Tweets streaming and storing script

The Python-based implemented code snippet for retrieving JSON objects from the Twitter streaming API and storing them in a MongoDB database is shown in Figure 2. In a relational database, *twitter_db* would be the name of the database instance and *twiter_collection* would be the table in which the data objects are stored.

C. Dataset Size and Features

Following the data collection methodology described in Section IV.B, a dataset of four million tweets have been collected and stored in MongoDB. Each instance in the dataset is a tweet associated with multiple metadata. These metadata (i.e. features) contain information relating to the text of a tweet, users, and entities. Tweets are associated with multiple features that represent their syntactic and semantic status. However, this research is concerned with the textual features that make up a tweet. These are the features from which the embedding vectors will be generated.

The collected raw tweets had undergone preliminary scraping stages as discussed in Section IV.D. The tweets semantic features are preserved, while the unwanted noise such as redundant tweets (i.e. retweets) and tweets where length is less than a certain threshold are eliminated. Tokenization and phrase identification are performed to identify *n*-gram features. The removal of reposts and non-informative instances has reduced the dataset to one million examples.

D. Pre-processing

Accuracy of the learned word embedding vectors is linearly related to the dataset size of training examples. However, the quality of the training dataset is of no less importance than the quantity. According to the research report of The Data Warehousing Institute, '*Poor quality customer data costs U.S. business an estimated 611 billion dollars …'* [20]. Pre-processing techniques are, therefore, a prerequisite for various information systems to maintain data quality. While structured data is usually manipulated in relational databases and schemas, features of free natural text often require special means of management and storage due to its lack of structure. Unstructured text data is highly susceptible to noise, redundancy, and inconsistency as they are generated from heterogeneous sources. Pre-processing techniques are required to remove redundancies and inconsistencies as analyzing low-quality data usually result in low-quality mining results [21].

The focus of this research is to train a neural embedding model to learn real-valued vectors from the collected tweets (i.e. examples). However, the majority of raw tweets are erroneous and highly unstructured, due to the informal nature of the communication channel in which these tweets are propagated as discussed in Section I. Therefore, in order to learn efficient embedding models that accurately capture the semantic relations between words, it is necessary to learn from clean data. The pre-processing stages carried out on the raw corpus of tweets are as follows:

1) Removal of redundant and non-informative tweets: in this stage, all duplicate tweets and reposts are excluded. Tweets that contain nothing but a URL are also removed from the dataset. Similarly, tweets that are composed of only one word are eliminated as these lack sufficient context for the embedding model to learn.

2) Removal of URLs and punctuations: URLs and punctuations such as interrogation and exclamation marks are removed. While these parts may carry structural and syntactic information for other applications, they provide nothing but noise to the learning process of the embedding model, which tries to capture the relationships and latent semantics between words.

3) Canonicalization of hashtags and mentions: common user conventions such as #hashtags and @mentions are prevalent in Twitter. Hashtags are actual words that contribute to the meaning of a tweet and may occure in a different tweet without the hash sign. For example, some tweets may contain the word *brexit* and others may contain it as *#brexit*, wich are different representations for the same word. If the hash sign is left in the training corpus, the model will generate different embedding vectors for each form of the word even though they carry the same meaning. Therefore, only the hash sign (i.e. prefix) is removed and the rest of the word is retained. Mentions in tweets are references to other users in Twitter. Different usernames do not contibute to the relationship between word. However, because tweets are very short text, the plot where a username appears has an impact on the morphological structure of the sentence. Thus, all user meantions are replaced with special symbols rather than words such as 'user', as these may appear in the corpus and therefore cannot be used for replacement.

4) Splitting joint words: joint words and hashtags such as 'BetterOffOut' and 'strongerin' are comnmon in tweets due to the character limit. These are splitted following the probability driven hueristic proposed in [22]. Phrases are identified as discribed in Section IV.E.

5) Normalization of special symbols: the proposed pre-trained model is meant to learn embeddings for words and thus, all integer and decimal numbers are replaced with special characters.

The pre-processing stages discussed in this section are performed on each instance retrieved by the Twitter streaming API. Considering the raw tweet, *T,* and the preprocessed version of it, *Ť*, in the following illustrative example:

T: *#skydebate #EUvote The more people like @Barak_Obama stick their noses in to #Brexit vote, the more I want to vote #leave*

Ť: *sky debate EU vote The more people like xxx stick their noses in to Brexit vote, the more I want to vote leave*

These consecutive steps aim at reducing confusion during the learning iterations and consequently, generating efficient embedding vectors, which shall contribute to the enhancement of social media NLP applications.

E. N-gram Identification and Corpus Annotation

Theoretically, training a NN embedding model assuming all words in the corpus are isolated from each other is memory intensive [23]. Additionally, many phrases have a single meaning that is not simply a composition of the meaning of its individual words, such as 'New Jersey'. In this research, the authors perform a composite method that commence with detecting common phrases in the pre-processed tweets, then annotating the corpus with these words that are most likely phrases.

1) Discovering phrases in the corpus: the data driven approach used in [23] for identifying phrases in a corpus is followed. In this approach, phrases are identified based on the frequently occurring bigrams that are commonly embedded in discourse, such as 'vote leave' and 'stronger in'. The following formula is used:

$$Score\left(w_i, w_j\right) = \frac{freq(w_i w_j) - \delta}{freq(w_i) \times freq(w_j)} \qquad (1)$$

Where w_i and w_j are words occurring in a phrase and δ is a discounting coefficient that prevents bigrams of infrequently occurring words to be considered as phrases. The bigrams of frequency scores above the predefined threshold are formed as phrases.

2) Corpus tagging: this process involves annotating the corpus with the two-word phrases identified in the previous step. The words that make a phrases are joined using the underscore character. For example, '… *visited New York and San Francisco…*' would become '*…visited new_york and san_francisco…*'. Finally, the resulting corpus consists of unigrams and explicitly tagged bigrams.

V. The Word Embedding Model

This section describes the methodology undertaken in constructing the word embedding model and learning from the political tweets corpus that was collected and pre-processed ad discussed in Section IV.

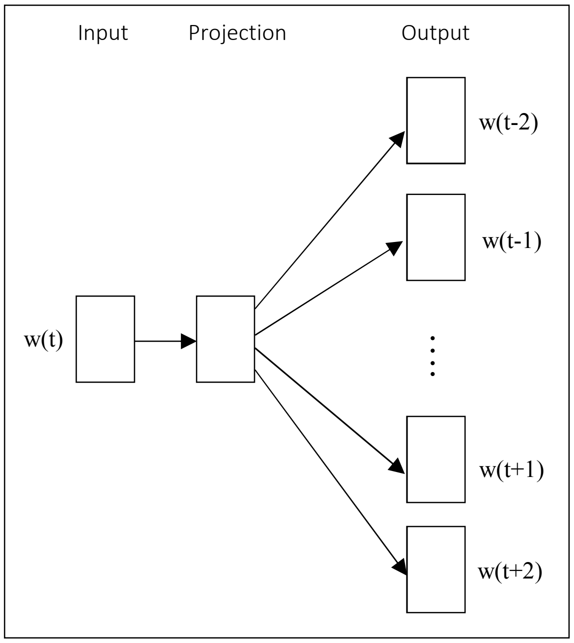

Fig. 3. Skip-gram model architecture [12]

A. Vocabulary Trimming

A vocabulary of 12.3 million words and phrases are included in the corpus. However, this vocabulary may contain rarely occurring words that lack enough context. Therefore, the minimum word frequency threshold is set to *min_count* = 3. Words and phrases that do not satisfy the *min_count* are discarded due to two reasons: 1) the NN model does not have adequate trainings examples to learn meaningful embedding vectors for those words, and 2) through performing basic corpus statistics, words occurring less than 3 times in the entire corpus appear to be mostly typos. The value of the *min_count* threshold has been determined empirically. The application of the minimum frequency threshold has generated a vocabulary V = 86K unique words and phrases in the training embedding model.

B. Model Architecture and Hyperparameter Configuration

In this research, a Word2Vec Skip-gram NN model with negative sub-sampling is used [23]. The use of the Skip-gram model and sub-sampling frequently occurring words decreases the number of training examples, and consequently, reduces the computational burden of the training process. This model is a shallow neural network with a single hidden layer. The learning process is unsupervised, in which the goal is to learn the weights between the input layer and the hidden layer that are actually the embedding vector representations of words. This is similar to the unsupervised feature learning in training an auto-encoder. The architecture of the implemented neural network model is shown in Figure 3.

1) Input layer: in this layer, the training examples (i.e. word pairs) are fed into the network. It has been reported that a context window size of $w' = 5$ is considered a good trade-off between efficiency and accuracy [16]. Empirical experiments were performed on different window sizes, $w' \in \{3, 4, 5, 6\}$ and have shown that $w' = 5$ produces the best embedding vectors for tweets. The output probabilities predict the likelihood of a word occurring in the domain of the input word (i.e. the word's context window). For example, training the network on the word 'TTIP', which is a typical acronym of *translatic trade and investment partnership* in the event of brexit, the output probabilities are higher for words like 'trade' and 'union'.

TABLE 1
ILLUSTRATIVE EXAMPLE OF MODEL'S TRAINING INPUT FOR w• = 5

Sliding window (w' = 5)	**Target word**	**Context**
[Brexit issue no organization afford to]	Brexit	issue, no, organization, afford, to
[Brexit issue no organization afford to ignore]	Issue	Brexit, no, organization, afford, to, ignore
[Brexit issue no organization afford to ignore]	No	Brexit, issue, organization, afford, to, ignore
[Brexit issue no organization afford to ignore]	Organization	Brexit, issue, no, afford, to, ignore
[Brexit issue no organization afford to ignore]	Afford	Brexit, issue, no, organization, to, ignore
[Brexit issue no organization afford to ignore]	To	Brexit, issue, no, organization, afford, ignore
[issue no organization afford to ignore]	Ignore	issue, no, organization, afford, to

Considering *T, 'Brexit issue no organization afford to ignore'* as an example tweet in the annotated corpus described in Section IV.E, the training samples for T at $w' = 5$ are shown in Table 1. Subsampling is implemented to eliminate highly frequent words with marginal information content, such as 'the'. The probability, p, of which a given word is retained in the vocabulary, is calculated as follows:

$$p(w_i) = \left(\sqrt{\frac{z(w_i)}{0.001}} + 1\right) \times \frac{0.001}{z(w_i)} \tag{2}$$

$$p(w_i) = \begin{cases} 1, & z(w_i) < 0.0026 \\ 0.5, & z(w_i) = 0.00746 \\ 0.033, & (w_i) = 1.0 \end{cases}$$

Where $z(w_i)$ is the fraction of the total occurrence of the word w_i in the corpus and the sample value of 0.001 is the default sampling parameter [22].

2) Hidden layer: in this layer, the dimensions of the embedding vectors is set to d = 300. That is, the configured model is learning word vectors with 300 features instead of the high dimensional vocabulary size. The hidden layer is thus represented by a weight matrix A (86K×300d), with 86K raws (1 per each record in the vocabulary) and 300 columns (1 per each hidden neuron).

3) Output layer: A vector for each word in the vocabulary acts as an input to the output layer. To optimize the computational burden in this layer, a negative sampling is performed to avoid updating every neuron weights for each vector in the vocabulary during training. Rather, only a small ratio of the weights are modified by each training vector. We randomly select five negative words, in which their weights are updated as well as the weights of the word in the training iteration. It has been reported in [22] that negative sampling value of five words works well for our dataset size range. The selection of the negative samples is based on a unigram distribution approach, in which more frequent words are more likely to be sampled.

TABLE 2
CORPUS AND MODEL METADATA AND HYPER-PARAMETERS

Metadata and Hyper-parameters	**Political Corpus of Tweets**
Raw tweets	4 million
Pre-processed tweets	1 million
Words in the corpus	12.3 million
Unique tokens in the trained embedding model	V = 86K
Neural network architecture	*Word2Vec Skip-gram / negative sub-sampling*
Negative samples	*5*
Vector dimension	$d = 300$
Minimum frequency threshold	*min_count* = 3
Learning context window	$w' = 5$
Training time	17 minutes
Training complexity	$O(\log_2(\lvert V\rvert))$
Trained model size	136MB
Processor and memory	intel core i7 CPU / 16GB RAM

C. Trained Model and Complexity

The model's training complexity is $O(\log_2(|V|))$, where V is the vocabulary size. Training the Word2Vec model on the political tweets dataset has taken about seventeen minutes on intel core i7 CPU and 16GB RAM. The statistical information on the learning corpus, trained embedding model, training configurations, and processor and memory specifications are shown in Table 2.

VI. CONCLUSION AND FUTURE WORK

Distributed word representations have shown to be successful in many computational linguistic applications as discussed in Section I. However, upon conducting a literature review, it has been observed that there is a lack of embedding models trained on domain specific microblogging posts, particularly tweets. This paper contributes to the literature in several significant ways. First, a corpus of over four million political tweets on the EU Referendum rich domain of controversial views is collected. The constructed corpus is pre-processed according to the methodology described in Section IV. Second, a word embedding model is trained on the collected and pre-processed corpus of tweets in order to learn meaningful words representations. The generated pre-trained model contains representations for 86K unique words and phrases.

Future work carries on as follows:

- The performance of the trained embedding model will be extrinsically evaluated through a semantic similarity measure for tweets. Alongside the NLP application evaluation, a machine learning based application of semantic cluster analysis will be carried out to evaluate the learned word vectors..
- A further extension political dataset on the event of 'leaving the EU' on March 2019 will be collected, and the trained model will be augmented to learn extended representations and increase the vocabulary size. Maintenance works to configure the artificial neural network's hyper-parameters and metadata will be performed accordingly.
- The augmented and optimised pre-trained model will be evaluated in different microblogging OSN computational intelligent applications.

VII. REFERENCES

1. Harris, Z.S., *Distributional structure.* Word, 1954. **10**(2-3): p. 146-162.
2. Bengio, Y., et al., *A neural probabilistic language model.* Journal of machine learning research, 2003. **3**(Feb): p. 1137-1155.
3. Socher, R., et al. *Recursive deep models for semantic compositionality over a sentiment treebank.* in *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013.
4. Tang, D., et al. *Learning sentiment-specific word embedding for twitter sentiment classification*. in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2014.
5. Li, Q., et al. *Tweet Sentiment Analysis by Incorporating Sentiment-Specific Word Embedding and Weighted Text Features*. in *Web Intelligence (WI), 2016 IEEE/WIC/ACM International Conference on*. 2016. IEEE.
6. Kim, Y., *Convolutional neural networks for sentence classification.* arXiv preprint arXiv:1408.5882, 2014.
7. Li, Q., et al. *Tweetsift: Tweet topic classification based on entity knowledge base and topic enhanced word embedding*. in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2016. ACM.
8. Taddy, M., *Document classification by inversion of distributed language representations.* arXiv preprint arXiv:1504.07295, 2015.
9. Li, Q., et al. *Hashtag recommendation based on topic enhanced embedding, tweet entity data and learning to rank.* in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2016. ACM.
10. Schwenk, H. and J.-L. Gauvain. *Connectionist language modeling for large vocabulary continuous speech recognition*. in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*. 2002. IEEE.
11. Collobert, R., et al., *Natural language processing (almost) from scratch.* Journal of Machine Learning Research, 2011. **12**(Aug): p. 2493-2537.
12. Mikolov, T., et al., *Efficient estimation of word representations in vector space.* arXiv preprint arXiv:1301.3781, 2013.
13. Pennington, J., R. Socher, and C. Manning. *Glove: Global vectors for word representation*. in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014.
14. Naili, M., A.H. Chaibi, and H.H.B. Ghezala, *Comparative study of word embedding methods in topic segmentation.* Procedia Computer Science, 2017. **112**: p. 340-349.
15. Kozareva, Z., et al. *Sentiment analysis in twitter.* in *Proceedings of the 7th International Workshop on Semantic Evaluation. Association for Computation Linguistics*. 2013.
16. Li, Q., et al., *Data sets: Word embeddings learned from tweets and general data.* arXiv preprint arXiv:1708.03994, 2017.
17. Kumar, S., F. Morstatter, and H. Liu, *Twitter data analytics*. 2014: Springer.
18. Banker, K., *MongoDB in action*. 2011: Manning Publications Co.

19. Davidson, S., *Wordnik.* The Charleston Advisor, 2013. **15**(2): p. 54-58.
20. Eckerson, W.W., *Data quality and the bottom line: Achieving business success through a commitment to high quality data.* The Data Warehousing Institute, 2002: p. 1-36.
21. Ciszak, L. *Application of clustering and association methods in data cleaning*. in *Computer Science and Information Technology, 2008. IMCSIT 2008. International Multiconference on*. 2008. IEEE.
22. Alnajran, N., et al. *A Heuristic Based Pre-processing Methodology for Short Text Similarity Measures in Microblogs*. in *High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2018 IEEE 20th International Conference on*. 2018. IEEE.
23. Mikolov, T., et al. *Distributed representations of words and phrases and their compositionality*. in *Advances in neural information processing systems*. 2013.