Inf Retrieval J (2018) 21:183–207
https://doi.org/10.1007/s10791-017-9319-5

NEURAL INFORMATION RETRIEVAL

Using word embeddings in Twitter election classification

Xiao Yang[1] · Craig Macdonald[1] · Iadh Ounis[1]

Received: 8 November 2016 / Accepted: 2 October 2017 / Published online: 9 November 2017

Abstract Word embeddings and convolutional neural networks (CNN) have attracted extensive attention in various classification tasks for Twitter, e.g. sentiment classification. However, the effect of the configuration used to generate the word embeddings on the classification performance has not been studied in the existing literature. In this paper, using a Twitter election classification task that aims to detect election-related tweets, we investigate the impact of the background dataset used to train the embedding models, as well as the parameters of the word embedding training process, namely the context window size, the dimensionality and the number of negative samples, on the attained classification performance. By comparing the classification results of word embedding models that have been trained using different background corpora (e.g. Wikipedia articles and Twitter microposts), we show that the background data should align with the Twitter classification dataset both in data type and time period to achieve significantly better performance compared to baselines such as SVM with TF-IDF. Moreover, by evaluating the results of word embedding models trained using various context window sizes and dimensionalities, we find that large context window and dimension sizes are preferable to improve the performance. However, the number of negative samples parameter does not significantly affect the performance of the CNN classifiers. Our experimental results also show that choosing the correct word embedding model for use with CNN leads to statistically significant improvements over various baselines such as random, SVM with TF-

This manuscript extends a version previously made available on arXiv (https://arxiv.org/abs/1606.07006) entitled “Using Word Embeddings in Twitter Election Classification”, and presented at the Workshop on Neural Information Retrieval at SIGIR 2016.

✉ Xiao Yang
Xiao.Yang@glasgow.ac.uk

Craig Macdonald
Craig.Macdonald@glasgow.ac.uk

Iadh Ounis
Iadh.Ounis@glasgow.ac.uk

1 School of Computing Science, University of Glasgow, Glasgow, UK

IDF and SVM with word embeddings. Finally, for out-of-vocabulary (*OOV*) words that are not available in the learned word embedding models, we show that a simple *OOV* strategy to randomly initialise the *OOV* words without any prior knowledge is sufficient to attain a good classification performance among the current *OOV* strategies (e.g. a random initialisation using statistics of the pre-trained word embedding models).

Keywords Word embedding · CNN · Twitter · Election classification · Word2vec

1 Introduction

Word embeddings have been proposed to produce effective word representations. For example, in the *Word2Vec* model (Mikolov et al. 2013c), by maximising the probability of seeing a word within a fixed context window, it is possible to learn for each word in the vocabulary a dense real-valued embedding vector from a shallow neural network. As a consequence, similar words are close to each other in the embedding space (Bengio et al. 2003; Collobert et al. 2011; Mikolov et al. 2013c), such as "quick" and "quickly" that are syntactically similar. However, word embeddings can provide more complex semantic relationships by applying algebraic operations to certain word vectors. For example, Mikolov et al. (2013d) observed that a simple algebraic operation *vector("King")* − *vector("Man")* + *vector("Woman")* results in a vector that is closest to the vector representation of "Queen". A similar example is to find the country that a city belongs to, such as "France" is to "Paris" what "China" is to "Beijing". Such semantic relationships are useful for various tasks such as information retrieval (Mitra et al. 2016), named entity tagging (Godin et al. 2015), text classification (Kim 2014) and machine translation (Mikolov et al. 2013b).

However, a common issue of using word embeddings is the out-of-vocabulary (*OOV*) words that appear in the datasets but not in the word embedding models. Thus, the vector representations of the *OOV* words cannot be obtained from the learned word embedding model. There are several existing *OOV* strategies in the existing literature. For example, Kim (2014) used the statistics from learned word embedding models to randomly initialise the vector representations for the *OOV* words. On the contrary, the *OOV* words were ignored by both Bojanowski et al. (2016) and Mitra et al. (2016). Dhingra et al. (2016) proposed character-based distributed representations which learn embeddings for each character rather than each word, and therefore the vectors for *OOV* words are resolved at the character level. However, there has been little exploration of which strategy is better at dealing with the *OOV* words from the word level, and indeed both Kim (2014) and Mitra et al. (2016) suggested that further study on the *OOV* strategies for word embedding models is needed.

Recently, the use of word embeddings together with convolutional neural networks (CNN) has been shown to be effective for various classification tasks such as sentence classification (Kim 2014) and sentiment classification on Twitter (Ebert et al. 2015; Severyn and Moschitti 2015). In such approaches, word embeddings are used to construct the vector representation of a sentence or a tweet as the input of a CNN classifier. In order to investigate the effect of different CNN settings in sentence classification performance, a sensitivity analysis of a one-layer CNN classifier has been conducted by Zhang and Wallace (2015), by varying the hyperparameters such as the filter region size, the number of feature maps and the pooling strategy. However, the effect of the configuration used to generate the word embeddings on the classification performance has not been studied in the literature. Indeed, while different background corpora (e.g. Wikipedia and Twitter) and parameters (e.g. context window and dimensionality) could lead to different word embeddings, there has been little exploration of how such background corpora and parameters affect the classification performance.

In this paper, using two Twitter datasets collected during the Venezuela parliamentary election in 2015 and the Philippines general election in 2016 respectively, we investigate the use of word embeddings with CNN in a new classification task, which aims to identify those tweets that are related to the election. Such a classification task is challenging because election-related tweets are usually ambiguous and it is often difficult for human assessors to reach an agreement on their relevance to the election (Bermingham and Smeaton 2011). For example, such tweets may refer to the election implicitly without mentioning any political party or politician. In order to tackle these challenges, we propose to use word embeddings to build richer vector representations of tweets for training the CNN classifier on our election datasets.

Our contributions are three-fold: First, we thoroughly investigate the parameter settings (e.g. context window, the dimensionality and negative sample size) of word embeddings on our election classification task. We show that word embeddings trained using a large context window size and dimension size can help CNN to achieve a significantly better classification performance over our baselines (e.g. SVM with TFIDF). Second, we explicitly study the effect of the background corpus. Our results show that when the type and time period of background corpus align with the classification dataset, the CNN classifier achieves statistically significant improvements over the classification baseline of SVM with TF-IDF on our task. Third, we compare several random strategies to address the *OOV* words. Our results on two datasets demonstrate that simpler *OOV* strategies to randomly initialise the *OOV* words without any prior knowledge is sufficient to attain a good classification performance among the current *OOV* strategies. Thus, our results indeed suggest that the background corpus and parameters of word embeddings have an impact on the classification performance. Moreover, our results contradict the findings on different tasks such as dependency parsing (Bansal et al. 2014) and named entity recognition (NER) (Godin et al. 2015) where a smaller context window is suggested. Such a contradiction suggests that the best setup of parameters such as the context window and dimensionality might differ from a task to another.

In the remainder of this paper, we explain the related work in Sect. 2. We describe the CNN architecture used for our classification task in Sect. 3. In Sect. 4, we describe our datasets and the experimental setup. In Sect. 5, we investigate the impact of the context window size, dimensionality and negative sample size of word embeddings on the classification performance. In Sect. 6, we discuss the impact of two different types of background corpora (Wikipedia articles and Twitter microposts) on the effectiveness of the learned classifier. In Sect. 7, we study the strategies that aim to deal with the out-of-vocabulary (*OOV*) words. We provide concluding remarks in Sect. 8.

2 Related work

In this section, we introduce related work in the areas of word embedding, Twitter classification and how they relate to the study presented in this paper.

2.1 Word embedding models

In most text classification tasks, terms within the documents are often used as features such as the classic TF-IDF vector representation. Word embedding models based on neural networks have emerged as an effective alternative to build vector representations of text (Bengio et al. 2003; Collobert and Weston 2008; Mikolov et al. 2013a; Pennington et al. 2014). The main aim of word embeddings is to learn vector representations of words by mapping semantic information into a geometric word embedding space. In these models, the vector representation w of a given word is usually learned through a fixed context window W. In addition, to capture semantic information from the fixed context window, the recently proposed *GloVe* model (Pennington et al. 2014) also aims to capture global corpus statistics through the word co-occurrence probabilities. Another word embedding model, namely *Word2Vec* (Mikolov et al. 2013a, c), maximises the conditional probability of a word given the context words that appeared around that word within the context window W. After training, the learned vector representations w can be used to reveal the relation between two words using their corresponding vector representations w_i and w_j and a similarity measure (e.g. *cosine* similarity):

$$sim(w_i, w_j) = cosine(w_i, w_j) = \frac{w_i \cdot w_j}{||w_i||||w_j||} \tag{1}$$

In particular, the *Word2Vec* model contains two separate embeddings, namely the input and output embeddings (Mitra et al. 2016). However, the output embedding is usually discarded in most applications (Godin et al. 2015; Kim 2014; Mikolov et al. 2013b). To leverage both of the embeddings, Mitra et al. (2016) proposed a dual word embedding model for document ranking by retaining the output embedding that is often discarded in other applications. Using the *cosine* similarity defined in Eq. (1), Mitra et al. (2016) noted that words in a dual word embedding model are more likely related by topics (e.g. "yale" is close to "faculty"). Without retaining the outputs embeddings, words are more likely related by types (e.g. "yale" is close to "harvard"). In particular, Bansal et al. (2014) and Pennington et al. (2014) observed that the embedding parameters can also affect the type of generated word embeddings. This shows that word embeddings can exhibit different properties in various settings, which leads to our proposed study on the effect of the embedding parameters such as the context window size W on the resulting classification performance.

2.1.1 Embedding parameters

A number of studies have already shown that the context window and dimensionality of the used word embedding vectors can affect performance in various tasks such as dependency parsing (Bansal et al. 2014) and named entity recognition (NER) (Godin et al. 2015). For instance, using publicly available corpora such as Wall Street Journals and Wikipedia, Bansal et al. (2014) investigated *Word2Vec* word embeddings in a dependency parsing task, which aims to provide a representation of grammatical relations between words in a sentence. By only varying the context window size from 1 to 10, their results on the accuracy of part-of-speech (POS) tagging showed that the context window size of *Word2Vec* can affect the type of the generated word embedding. In particular, they observed that a smaller context window gives a better performance on accuracy. In the named entity recognition (NER) task, Godin et al. (2015) investigated three context window sizes $W = \{1, 3, 5\}$ based on the accuracy of NER tagging. From their results, they also reached a similar conclusion, namely that a smaller context window gives a better performance using the *Word2Vec* word embeddings when the model is trained from a large Twitter corpus containing 400 million tweets.

Using a subset of the semantic-syntactic word relationship test set, Mikolov et al. (2013c) investigated the dimensionality of the *Word2Vec* word embeddings and the size of background data. In the test set, word pairs are grouped by the type of relationship. For example "brother–sister" and "grandson–granddaughter" are in the same relationship of "man–woman". The accuracy is measured such that given a word pair, another word pair with the correct relationship should be retrieved. Using this accuracy measure, they noted that at some point increasing the dimensionality or the size of background data only provides slightly better performance. Therefore, they concluded that the dimensionality and background data size should be increased together (Mikolov et al. 2013c). Mikolov et al. (2013c) also studied another parameter, namely the number of negative samples, which defines how many negative examples are randomly sampled from the corpus vocabulary to train the word embedding models. For example, for the context "the cat sits on the mat", a negative sample will be a word (e.g. project) randomly sampled from the entire corpus, which is often irrelevant to the current context. Such negative examples help the word embedding model to differentiate the correct word relationships from noise (i.e. negative samples). Therefore, during training, the model maximises the probabilities to real word relationships and minimises the probabilities to the noise words. Mikolov et al. (2013c) observed that a large negative sample size is useful for small background corpora while for large corpora the negative sample size could be as small as 2–5. However, Mikolov et al. (2013c) only investigated the *Word2Vec* parameters using the GoogleNews background corpus.

The aforementioned studies provide a useful guide about the effect of the word embeddings configuration on performance in the specific applications they tackled, but their findings were obtained on tasks different from Twitter classification tasks. Hence, the question arises as whether such findings will generalise to classification tasks on Twitter, which is the object of our study in this paper.

2.2 Twitter classification

In fact, there is little work in the literature tackling the task of election classification on Twitter. However, similar classification tasks such as Twitter sentiment classification have been well studied (Ebert et al. 2015; Severyn and Moschitti 2015; Tang et al. 2014). In particular, word embeddings were recently used to build effective tweet-level representations for Twitter sentiment classification (Severyn and Moschitti 2015; Tang et al. 2014). For instance, in the SemEval-2015 Twitter Sentiment Analysis challenge, Severyn and Moschitti (2015) proposed to use word embeddings learned from two Twitter corpora to build the vector representations of tweets. Using the *Word2Vec* model, default parameter values such as context window size 5 and dimensionality 100 were applied to train the word embedding. In their approach, one Twitter background corpus (50 million tweets) was used to train the word embedding, while another one (10 million tweets) containing positive and negative emoticons was used to refine the learned word embeddings using the proposed CNN classifier. The CNN classifier was then trained on the SemEval-2015 Twitter sentiment analysis dataset, which contains two subsets: phrase-level and message-level datasets. Each subset contains 5K+ and 9K+ training samples, respectively. The official ranking in SemEval-2015 showed that this system ranked 1st and 2nd on the phaselevel dataset and the message-level dataset, respectively. However, Severyn and Moschitti (2015) focused on refining the word embeddings by using another Twitter corpus with emoticons to learn sentiment information, but did not study the impact of the background corpus and the chosen parameters on the classification performance.

In another approach based on the word embeddings model proposed by Collobert and Weston (2008) and Tang et al. (2014) proposed a variation to learn sentiment-specific word embeddings (SSWE) from a large Twitter corpus containing positive and negative emoticons. Tang et al. (2014) empirically set the context window size to 3 and the embedding dimensionality to 50. The SemEval-2013 Twitter sentiment analysis dataset, which contains 7K+ tweets was used to evaluate the effectiveness of their proposed approach. Compared to the top system of the SemEval-2013 Twitter Sentiment Analysis challenge, their approach of using an SVM classifier with SSWE outperformed the top system on the F1 measure. However, only the Twitter background corpus was used by Tang et al. (2014), which contains 10 million tweets with positive and negative emoticons. On the other hand, the parameters of word embeddings such as the context window and dimensionality were not studied by Tang et al. (2014), nor in the existing literature for Twitter classification tasks. In particular, the aforementioned studies do not address the out-of-vocabulary (*OOV*) words that are not appeared in the learned embedding model. As such, in this paper, we conduct a thorough investigation of word embeddings together with CNN on a Twitter classification task and explore the impact of both the background corpus, the context window, the dimensionality and the negative sample size of word embeddings and *OOV* words on the classification performance.

3 The CNN model

For our Twitter election classification task, we use a simple CNN architecture described by Kim (2014) and Severyn et al. (2015) and highlighted in Fig. 1. It consists of a convolutional layer, a max pooling layer, a dropout layer and a fully connected output layer. Each of these layers is explained in turn.

Tweet-level representation The inputs of the CNN classifier are preprocessed tweets that consist of a sequence of words. Word embeddings play an important role in building the tweet-level representations. The semantic relations carried by word embeddings are helpful to find semantic similarities between tweets.

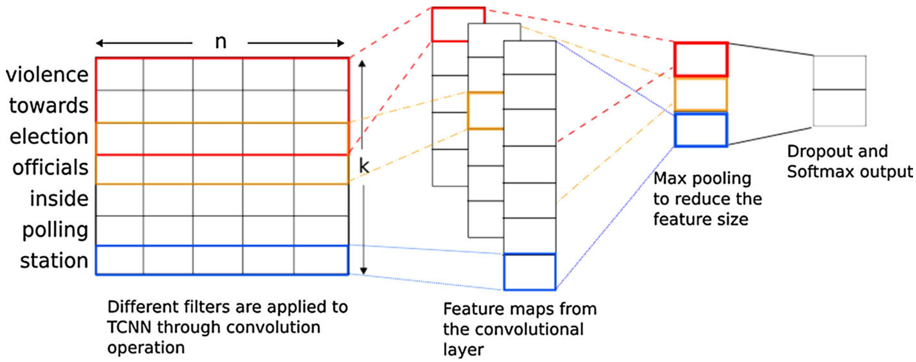

Fig. 1 Convolutional neural network architecture for tweet classification. Adapted from Kim (2014)

Using word embeddings, tweets are converted into vector representations in the following way. Assuming $w_i \in \mathbf{R}^n$ to be the n-dimensional word embeddings vector of the i-th word in a tweet, a tweet-level representation for convolutional neural networks (denoted *TCNN*) is obtained by looking up the word embeddings and concatenating the corresponding word embeddings vectors of the total k words:

$$TCNN = w_1 \oplus w_2 \oplus \cdots \oplus w_k \tag{2}$$

where $\oplus$ denotes the concatenation operation (Kim 2014). As suggested by Kim (2014), for a word not appearing in a word embeddings (also known as *out-of-vocabulary OOV*), we generate its vector by sampling each dimension from the uniform distributions $U_i[m_i - s_i, m_i + s_i]$, where m_i and s_i are the mean and standard deviation of the i-th dimension of the word embeddings. For training purposes, short tweets are padded to k—the length of the longest tweet—using a special token. The vector representation of each word is concatenated and stacked as illustrated in Fig. 1. Hence, the total dimension of the vector representation *TCNN* is always $k \times n$. Afterwards, the tweet-level representation will feed to the convolutional layer.

Convolutional layer The convolutional layer consists of a set of learnable filters that are applied to the network input *TCNN* using convolution operations. Since the size of each filter $F_i \in \mathbf{R}^{m \times n}$ is usually smaller than *TCNN*, a filter aims to only detect the presence of specific features or patterns. This is the core building block of convolutional neural networks, which helps the network to learn the important patterns no matter where they appear in a tweet (Severyn and Moschitti 2015). In this layer, the filter $F_i \in \mathbf{R}^{m \times n}$ is randomly initialised and applied to the tweet-level representation *TCNN*. By varying the size of m, multiple filter sizes can be used to cover m words during the convolution operation as shown in Fig. 1, where three filter sizes are illustrated in different colours. By this means, the network learns important features by considering two or more adjacent words together. In this layer, by varying another parameter, namely stride s (Krizhevsky et al. 2012), we can shift the filters across s word embeddings vectors at each step. Therefore, a larger s leads to less computation. By sliding the filters over m word vectors in *TCNN* using stride s, the convolution operation produces a new feature map c_i for all the possible words in a tweet:

$$c_i = f(F_i \cdot TCNN_{i:i+m-1} + b_i) \tag{3}$$

where $i : i + m - 1$ denotes the word vectors of word i to word $i + m - 1$ in *TCNN*. b_i is the corresponding bias term that is initialised to zero and learned for each filter F_i during training. In Eq. (3), f is the activation function. In this CNN architecture, we use a rectified linear function (ReLU) (Hahnloser et al. 2000) as the activation function f since ReLU shows very promising performance in convolutional neural networks (Glorot et al. 2011). Its output is given by:

$$Output = Max(0, Input) \tag{4}$$

Therefore, the ReLU unit ensures its output is always positive.

Max pooling layer A pooling layer aims to reduce the spatial size of features and the computation in the network. All the feature maps c_i from the convolutional layer are then applied to the max pooling layer where the maximum value c_i^{max} is extracted from the corresponding feature map. In this way, only the most salient features are kept. Afterwards, the maximum values of all the feature maps c_i are concatenated as the feature vector of a tweet.

Dropout layer Dropout is a simple yet effective regularisation technique that is often used in neural networks (Srivastava et al. 2014). A neuron is a basic unit in such neural networks, which is a mathematical function that attempts to model a biological neuron. It sums the inputs and passes them to an activation function to produce an output. However, dropout only keeps a neuron active with some probability p (e.g. $p = 0.5$) during training (Kim 2014). After training, $p = 1$ is used to keep all the neurons active for predicting unseen tweets. We apply both dropout and the well-known $L2$ regularisation technique to control overfitting.

Softmax layer The fully connected softmax layer transforms the output scores of positive and negative classes into normalised class probabilities (Kim 2014) using the softmax function:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{t=0}^{1} e^{z_t}} \quad for \quad i = 0, 1, \tag{5}$$

where $\hat{y}_i$ indicates the normalised probability of class i. z_i denotes the output score of class i. The value of i can only be 0 or 1 in our task since we aim to only classify tweets as "election-related" or "not election-related", which is a binary classification task. Let $\mathbf{y}$ be a vector representing the true label distribution and $\hat{\mathbf{y}}$ be the vector of the normalised probabilities from the softmax layer, the cross-entropy cost function is defined as follows:

$$loss(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=0}^{1} y_i log(\hat{y}_i) \tag{6}$$

Equation (6) calculates the dissimilarity between the true label distribution $\mathbf{y}$ and the predicted label distribution $\hat{\mathbf{y}}$. During training, the weights of each layer are updated according to the calculated loss. Once a CNN classifier is trained from a training set, all of its parameters and learned weights are saved. Unseen tweets can then be classified by applying their tweet-level representations to the trained CNN classifiers.

4 Experimental setup

In this paper, we argue that the types of background corpora as well as the parameters of *Word2Vec* model could lead to different word embeddings and could affect the performance on Twitter classification tasks. In the following sections, we address three research questions using our Twitter election classification task:

- **RQ1**: For Twitter election classification task, do the CNN classifiers prefer different word embedding settings from other tasks, e.g. dependency parsing (Bansal et al. 2014) and NER (Godin et al. 2015)?
- **RQ2**: By using the same type of background corpus as the background dataset, does the learned word embedding model improve the classification performance?
- **RQ3**: For out-of-vocabulary (*OOV*) words, what particular strategy (e.g. random initialisation using statistics of the pre-trained word embedding models) is preferred to attain good classification performance?

Our experiments are tailored to conduct a thorough investigation of word embeddings together with CNN on a Twitter classification task by addressing **RQ1** in Sect. 5, **RQ2** in Sect. 6 and **RQ3** in Sect. 7. The remainder of this section details our datasets (Sect. 4.1), our experimental setup and used word embedding models (Sect. 4.2), the used baselines (Sect. 4.3) and measures (Sect. 4.4)

4.1 Datasets

Our two manually labelled election datasets are sampled from tweets collected about the 2015 Venezuela parliamentary election (held on 06/12/2015) and the 2016 Philippines general election (held on 09/05/2016), respectively. Both of the datasets cover the period of one month before and after the election dates. We use the Terrier information retrieval (IR) platform (Macdonald et al. 2012) and the DFReeKLIM (Amati et al. 2011) weighting model—designed for microblog search—to retrieve tweets related to selected query terms that were provided by social science experts (we list all our query terms in Table 1a). Only the top 7 retrieved tweets are selected for each query term on each of the 60 days, making the size of the collection realistic for human assessors to examine and label the tweets. The sampled tweets are merged into one pool and judged by 5 experts who label a tweet as: "Election-related" or "Not Election-related". To determine the judging reliability, an agreement study was conducted for the Venezuela dataset using 482 random tweets that were judged by all 5 assessors. Using Cohen's *kappa* (Cohen 1968), we found a moderate agreement of 52% between all assessors. For tweets without a majority agreement, an additional expert of Venezuela or Philippines politics was used to further clarify their categories. Therefore, we obtain a dataset with good quality human labels. In total, our Venezuela election dataset consists of 5747 Spanish tweets, which contains 9904 unique words after preprocessing (stop-word removal & Spanish Snowball stemmer). In our Philippines election dataset, there are 4163 English tweets with a total 10,229 unique words after preprocessing (stop-word removal & English Snowball stemmer). Overall, both of our labelled election datasets cover significant events during the elections. For example, the killing of opposition politician Luis Diaz (BBC 2015) in the 2015 Venezuela parliamentary election is observed in our Venezuela dataset. From the general statistics shown in Table 1b, we observe that both datasets are unbalanced; the majority class (non-election) has more tweets than the minority class (election).

Table 1 Query terms and statistics of the datasets used in the experiments

	(a) Query terms used to retrieve tweets
	Query terms
Venezuela dataset	eleccion, violencia, votar, pistola, armas, ametralladora, ataque electora, muerto, miedo, muerte, asesinato, disparar, fraude muere, delincuente, herido, agreden, asesinar, guachiman, protesta
Philippines dataset	violence, attack, dead, fraud, assault, protest, intimidation, unrest gunshot, racial, die, kill, threat, vote buying, murder, corrupt terrorize, ambush, explosion, shoot, fire, harass, injure, burn selling vote, cheating, election

	(b) Statistics of the labelled datasets				
	Language	Election	Non-Election	Total	# Words
Venezuela dataset	Spanish	2274	3474	5747	9904
Philippines dataset	English	1755	2408	4163	10,229

4.2 Word embeddings

As part of our experiments, we train word embedding models on different background corpora. When training on Twitter data, we removed tweets with less than 10 words since such tweets contains less information and are often meaningless (e.g. only contain Twitter handles or are dominated by stopwords and emoticons) for training the word embeddings in our task. For consistency, we apply the same preprocessing, namely stop-word removal and stemmer (see Sect. 4.1), to all of the background corpora. Since the Venezuela dataset is in Spanish while the Philippines dataset is in English, we train both Spanish and English word embedding models for our experiments.

4.2.1 Spanish word embedding models

Our Spanish word embeddings are for use with the Venezuela dataset that contains annotated tweets of the 2015 Venezuela parliamentary election (held on 06/12/2015). The word embeddings used in this paper are trained from three different background corpora:

- `es-Wiki`: a Spanish Wikipedia snapshot dated 02/10/2015.
- `es-Twitter-general`: a general Spanish Twitter data collected from 05/01/2015 to 30/06/2015, which does not align with the election period of the 2015 Venezuela parliamentary election.
- `es-Twitter-time`: a Spanish Twitter data collected from the period 01/11/2015 to 31/12/2015, which covers the election period of the 2015 Venezuela parliamentary election.

Over 1 million Spanish articles are observed in `es-Wiki`. After removing tweets with less than 10 words, over 20 million Spanish tweets are collected in the corpus `es-Twitter-general`. In `es-Twitter-time`, over 5 million Spanish tweets are observed. After preprocessing, the `es-Wiki` corpus contains 436K unique words, `es-Twitter-general` has 629K unique words while `es-Twitter-time` has only 196K unique words. Salient statistics are provided in Table 2a. Indeed, by comparing the unique words in our election datasets with the words in the three background corpora, we observe that

Table 2 Statistics of the background corpora used to train word embedding models and words coverage on the election datasets

	(a) Spanish word embeddings		
	`es-Wiki`	`es-Twitter-general`	`es-Twitter-time`
# Documents	1M+	20M+	5M+
Vocabulary size	436K	629K	196K
Word coverage count	5111	6612	6309
Word coverage rate (%)	51	66	63
	(b) English word embeddings		
	`es-Wiki`	`es-Twitter-general`	`es-Twitter-time`
# Documents	4.9M+	32M+	18M+
Vocabulary size	925K	1.05M	649K
Word coverage count	4745	5335	5610
Word coverage rate	46%	52%	54%

5111 words in our Venezuela dataset appear in `es-Wiki`, 6612 words appear in `es-Twitter-general` while 6309 words appear in `es-Twitter-time`. This shows that `es-Twitter-general` has a better word coverage on our election datasets. We notice that `es-Twitter-time` has a very similar word coverage to `es-Twitter-general` though it has a much fewer number of documents.

4.2.2 English word embedding models

Our English word embeddings are for use with the Philippines dataset that contains annotated tweets of the 2016 Philippines general election (held on 09/05/2016).

- `en-Wiki`: an English Wikipedia snapshot dated 02/10/2015
- `en-Twitter-general`: a general English Twitter data collected from 05/01/2015 to 23/03/2015, which does not align with the election period of the 2016 Philippines general election
- `en-Twitter-time`: an English Twitter data that is collected from the period 01/04/2016 to 31/05/2016, which covers the election period of the 2016 Philippines general election

For English word embedding corpora, over 4.9 million English articles are observed in `en-Wiki`, over 32 million English tweets in the corpus `en-Twitter-general` and over 18 million English tweets are observed in `en-Twitter-time`. As shown in Table 2b, the statistics of English word embeddings after the preprocessing, the `en-Wiki` corpus contains 925K unique words, `en-Twitter-general` has 1.05M unique words while `en-Twitter-time` has only 649K unique words. Similar to the Spanish word embedding models, by comparing the unique words in our Philippines election dataset with the words in the three English word embedding models, we observe that `en-Wiki` has the lowest word coverage count, which shows that Twitter corpora have better word coverage on our election datasets. Hence, potentially the `en-wiki` model cannot work as well as `en-Twitter-general` and `en-Twitter-time` due to the low word coverage.

We use the *Word2Vec* implementation in *deeplearning4j*[1] to generate a set of word embeddings by varying the context window size W, the dimensionality D and the number of negative samples ns. We use context window sizes $W = \{1, 5, 10\}$ to study both small and large context window sizes. For each context window W, we use three different dimension sizes $D = \{200, 500, 800\}$ to study both of the low and high dimensionalities of the word embedding vectors. Therefore, 9 word embedding models in total are generated by varying W and D for both `es-Twitter-general` and `en-Twitter-general`. For other parameters, we use the same values that were set by Mikolov et al. (2013c): We set the batch size to 50, negative sampling to 10, minimum word frequency to 5 and iterations to 5. In addition, we also study the effect of the negative sample size on both `es-Twitter-general` and `en-Twitter-general` by using negative sample sizes $ns = \{2, 10\}$.

4.3 Baselines

To evaluate the CNN classifiers and word embeddings, we use four baselines, namely:

Random classifier According to the class distribution in the training set, the random classifier simply makes random predictions to the test instances.

[1] https://deeplearning4j.org/.

SVM with TF-IDF (SVM+TFIDF) As a more sophisticated baseline than the random classifier, the traditional weighting scheme, namely TF-IDF, is used in conjunction with an SVM classifier for the Twitter election classification.

SVM with word embeddings (SVM+WE) We use a similar scheme that was used by Wang et al. (2015) to build the tweet-level representation for the SVM classifiers. The vector representation (i.e. *TWE*) of a tweet is constructed by averaging the word embedding vectors along each dimension for all the words in the tweet:

$$TWE = \sum_{i=1}^{k} w_i / k \tag{7}$$

where k is the number of words in a tweet and $w_i \in \mathbf{R}^n$ denotes the word embedding vector of the i-th word. By this means, the vector representation of each tweet has exactly the same dimension as the word embedding vector w_i, which is the input of an SVM classifier. The concatenation scheme used in the CNN classifiers is not applied to the SVM classifiers because it gives worse performance according to our initial experiments. Indeed, a key advantage of the CNN classifier is to detect important patterns within a context window and capture word order, attributes that cannot be captured using SVM with or without word embeddings.

fastText As a state-of-the-art text classifier, *fastText* provides both effective and efficient learning of word representations and sentence classification (Bojanowski et al. 2016; Joulin et al. 2016). Based on the *Word2Vec* model, *fastText* can classify text documents using the word embeddings learned from the given dataset. In addition, it also allows to use *N-gram* features to capture some partial information about the local word order (Joulin et al. 2016). The efficiency and effectiveness of *fastText* has been tested on several datasets (e.g. the Yelp review and Amazon review datasets). Compared to other systems (Conneau et al. 2016; Tang et al. 2015; Zhang and LeCun 2015; Zhang et al. 2015) using convolutional neural networks and recurrent neural networks, *fastText* shows comparable results but significantly less training time.

4.4 Hyperparameters and measures

To train the classifiers and evaluate their performances on our datasets, we use a 5-fold cross validation, such that in each fold, 3 partitions are used for training, 1 partition for validation and 1 partition for test. Afterwards, the overall performance on the test instances is assessed by averaging the scores across all folds. We report effectiveness in terms of classification measures, precision (denoted P), recall (denoted R) and F1 score (denoted $F1$). For statistical testing, we use the non-parametric *McNemar's* test as proposed by Dietterich (1998) for an computationally inexpensive method of hypothesis testing.

For all the experiments, we use 3 filter sizes $m = \{1, 2, 3\}$, stride $s = 1$ and dropout probability $p = 0.5$ for our CNN classifier, which gives better performance according to our initial experiments on the validation set. For each filter size, 200 filters are applied to the convolutional layer and therefore 600 feature maps are produced in total. For the SVM classifiers, we use the *LinearSVC* model in *scikit-learn*[2] (Pedregosa et al. 2011) and tune the parameter c for each SVM classifier using the validation set. For *fastText*[3], the implementation provided by Joulin et al. (2016) is used. We tune the dimensionality

[2] http://scikit-learn.org.

[3] https://github.com/facebookresearch/fastText.

parameter *dim* to 200, the *N-gram* parameter *wordNgrams* to 2 and context window size *ws* to 5, which attains good performance according to our initial experiments. For other parameters, such as learning rate, we use the default settings.

5 Effect of word embeddings parameters

In this section, we address **RQ1** and investigate the effect of parameters (e.g. context window, dimensionality and negative samples) for the Twitter election classification task. As shown in Table 2, since word embedding models `es-Twitter-general` and `en-Twitter-general` have good word coverage on our Venezuela and Philippines datasets respectively, we use them for all the experiments in this section.

5.1 Effect of context window and dimensionality

For the Venezuela dataset, Table 3a shows the results of our four baselines, while Table 3b shows the results of classifiers using word embeddings, namely SVM with word embeddings (SVM+WE) and CNN. In Table 3b, the measurements for SVM+WE and CNN are arranged by the dimensionality and context window size of word embeddings. For each row of *W*1, *W*5 and *W*10, Table 3b shows results for context window sizes of $W = \{1, 5, 10\}$ along each dimension sizes of $D = \{200, 500, 800\}$. The best overall scores are highlighted in bold. Table 4 has the same arrangement as Table 3 but shows the corresponding results for the Philippines dataset.

In Table 3b, where both approaches (e.g. SVM+WE and CNN) use the same word embeddings, we observe that SVM+WE and CNN show similar preferences in word embeddings dimensionality. When we fix the context window size, SVM classifiers clearly show improvements on F1 score for all sizes of $W = \{1, 5, 10\}$ by increasing the dimensionality D. In particular, a larger dimensionality $D = \{500, 800\}$ results in better performances in terms of precision, recall and F1 scores compared to the embedding models with a smaller dimensionality $D = 200$. We observed similar results for the CNN classifiers. In particular, when $W = \{5, 10\}$, increasing the dimensionality D improves the recall and F1 scores.

Next, we study the effect of context window size W. If we fix the dimensionality D while increasing the context window size W from 1 to 10, CNN classifiers show improvement on F1 score. In particular, larger context window sizes $W = \{5, 10\}$ also attain better recall compared to the embedding models using $W = 1$ when $D = \{500, 800\}$. Similarly, the SVM classifiers also demonstrate improvements in recall and F1 when $D = 800$ when increasing context window size W. In particular, by considering both the context window and dimensionality, SVM and CNN classifiers attain best performances in of terms of F1 scores when $W = 10$ and $D = 800$. As such, the experimental results on the Venezuela dataset show that a word embedding model with a higher context window size and dimensional is potentially the most appropriate for our task.

For the Philippines dataset, as shown in Table 4b, we have observed similar results of the Venezuela dataset. By fixing context window W and increasing the dimensionality D, SVM classifiers clearly show improvements on all metrics. For CNN classifiers, the use of word embedding models with the largest dimensionality $D = 800$ attains the best F1 score. Next, we study the effect of context window. If we fix the dimensionality $D = \{200, 500\}$ and increase the context window W from 1 to 10, all metrics are improved for the SVM

Table 3 Results of our baselines and CNN models in the Twitter election classification task using the Venezuela dataset

(a) Results of random classifier, SVM+TFIDF, SVM+WE and *fastText*

	Precision	Recall	F1 score
Random*	38.6	28.5	38.5
SVM+TFIDF*	**82.6**	69.6	75.5
SVM+WE*	79.1	70.5	74.5
fastText=	81.2	**72.7**	**76.7**

(b) Results of SVM with word embeddings (SVM+WE) and CNN

	D200						D500						D800					
	SVM+WE			CNN			SVM+WE			CNN			SVM+WE			CNN		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
W1	78.1	65.7	71.3	81.2	71.6	76.1	78.3	69.4	73.6	80.7	71.9	76.0	79.6	69.0	74.0	80.8	71.6	75.9
W5	78.6	68.1	73.0	82.0	71.5	76.4	79.2	69.7	74.1	80.9	72.2	76.3	79.3	69.6	74.1	80.6	74.0	77.1
W10	78.9	65.6	71.6	80.8	72.8	76.6	78.9	68.3	73.2	80.7	73.9	77.1	79.1	70.5	74.5	81.6	73.9	**77.6**

*W*1 means context window size 1 and *D*200 denotes word embeddings dimension size 200

* Indicates that the difference between the CNN classifier ($W = 10$ and $D = 800$) and that baseline is statistically significant (McNemar's test, $p < 0.05$), while $^{=}$ indicates the difference is not significant

Table 4 Results of our baselines and CNN models in the Twitter election classification task using the Philippines dataset

(a) Results of random classifier, SVM+TFIDF, SVM+WE and *fastText*

	Precision	Recall	F1 score
Random*	42.4	42.1	42.2
SVM+TFIDF$^{=}$	**80.3**	**80.9**	**80.6**
SVM+WE*	79.0	78.2	78.6
fastText$^{=}$	79.4	**80.9**	80.0

(b) Results of SVM with word embeddings (SVM+WE) and CNN

	D200						D500						D800					
	SVM+WE			CNN			SVM+WE			CNN			SVM+WE			CNN		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
W1	76.8	74.8	75.8	79.1	80.2	79.6	78.8	76.0	77.3	81.2	79.2	80.1	79.0	78.2	78.6	80.9	79.7	80.2
W5	77.7	75.6	76.6	79.4	80.2	79.8	79.0	76.6	77.8	80.4	80.5	80.4	80.3	76.8	78.5	80.6	80.9	80.6
W10	78.4	76.4	77.4	81.5	79.3	80.4	79.0	77.0	78.0	79.8	81.1	80.3	79.9	77.3	78.5	82.2	79.4	**80.8**

*W*1 means context window size 1 and *D*200 denotes word embeddings dimension size 200

* Indicates that the difference between the CNN classifier ($W = 10$ and $D = 800$) and the baseline is significant. However, $^{=}$ indicates the difference is not significant

classifiers. We note that the size of context window W does not affect the performance of the SVM classifiers much when $D = 800$. However, the use of larger context window sizes $W = \{5, 10\}$ improves F1 score for CNN classifiers compared to a small window size $W = 1$. In particular, the CNN classifier attains the best F1 score by using the word embedding model with $W = 10$ and $D = 800$, which is the same to our finding on the Venezuela dataset. Hence, a word embedding model with a larger context window size and dimensionality is the most appropriate for our task.

5.1.1 Comparison to baselines

We first compare the results of the CNN classifiers to the random baseline and the SVM+WE baseline (shown in Tables 3a, 4a). Clearly, for the Venezuela dataset (Table 3a), the CNN classifiers outperform these two baselines across all measures. By comparing CNN classifiers to the SVM+TFIDF baseline, the CNN classifiers consistently outperform the SVM+TFIDF baseline on recall and F1 scores. As a state-of-the-art text classifier, *fastText* also leverages the word embedding model and neural networks, and its classification performance is very similar to the CNN classifier. Compared to the CNN classifier that uses the word embedding model of $W = 5$ and $D = 800$, *fastText* has slightly worse performance on all the metrics, which shows the effectiveness of convolution neural networks with word embeddings on the Twitter election classification task. Comparing CNN and SVM+TFIDF, our statistical test result shows the difference between them is significant, with a p-value less than 0.05. However, we note that the performance difference between CNN and *fastText* is not statistically significant.

For the Philippines dataset, the CNN classifiers consistently outperform the random and SVM+WE baselines (shown in Table 4a) on all the metrics. However, the performances of CNN, SVM+TFIDF and *fastText* are very similar. When the word embedding model with $W = 10$ and $D = 800$ is used, the CNN classifier has a slightly better F1 score compared to SVM+TFIDF. However, we note that the differences between CNN, SVM+TFIDF and *fastText* are not statistically significant, with p-values greater than 0.05. However, considering the overall performance on the two datasets, the CNN classifiers consistently attain the best results in terms of F1 score compared to the other baselines. Although *fastText* shows similar performance with CNN classifiers, it uses a bag of words that is invariant to the words ordering. Therefore, to take the order information into account, a bag of *N-grams* are used as additional features by *fastText*. On the contrary, the CNN classifier inherently captures the local word ordering using existing word embedding models and therefore does not need to learn *N-gram* features. In particular, the CNN parameters, such as the filter sizes and stride, provide more flexibility on how to capture the words ordering information as introduced in Sect. 3.

5.2 Effect of negative samples

We have shown that word embedding models with larger context window and dimensionality improve the classification performance in Sect. 5.1. However, there is another important parameter, namely the negative sample size (denoted *ns*), that could affect the classification performance as well. When negative sampling is used to train word embedding models, *ns* defines how many negative samples are randomly sampled for each data. Such negative samples help the word embedding model differentiate correct word relationships from noise. Mikolov et al. (2013a) observed that negative sample size *ns* in the range 5–10 is useful for small training corpora while for a large training corpus *ns* can

Table 5 CNN classification results by using word embedding models with different negative sampling size $ns = \{2, 10\}$ on both of our datasets

Negative sampling	Venezuela			Philippines		
	P	R	F1	P	R	F1
$ns = 2^{=}$	80.2	71.9	75.8	**81.6**	79.0	80.2
$ns = 10^{=}$	**80.9**	**72.2**	**76.3**	80.4	**80.5**	**80.4**

$^{=}$ indicates that the difference between $ns = 2$ and $ns = 10$ is not significant

be as small as 2–5. To study the effect of negative sample size *ns*, we train word embedding models with $ns = \{2, 10\}$ to cover a value from each range. Since in Sect. 5.1, a word embedding model with context window $W = 5$ and dimensionality $D = 500$ already attains good performance for the CNN classifiers, we use this setting for all the experiments in this part.

Results on both Venezuela and Philippines datasets are shown in Table 5 by varying the value of *ns*. A larger *ns* gives slightly better performances on recall and F1 scores across two different datasets as highlighted in Table 5. However, since a larger negative sample size *ns* requires much more training time, $ns = 10$ does not benefit the CNN classifiers much compared to $ns = 2$ in this case. The result of the *McNemar's* test also indicates that there is no significant difference between CNN classifiers learned using word embeddings with $ns = 2$ and $ns = 10$. In particular, the word embedding corpora (`en-Twitter-general` and `es-Twitter-general`) we used have different size: `en-Twitter-general` contains 12M more tweets and 421K more unique words than `es-Twitter-general`. As such we conclude that, a small negative sample size (i.e. *ns*) is sufficient for our Twitter election classification task.

5.3 Discussion

Compared to the studies on other tasks such as named entity recognition (NER) and dependency parsing (see Sect. 2), our results differ from their conclusions that "a smaller context window size gives a better performance" (Bansal et al. 2014; Godin et al. 2015). Such a contradiction suggests that the best setup of parameters such as context window and dimensionality may differ from our task to another. For example, both Bansal et al. (2014) and Mitra et al. (2016) have noted that the learned word embeddings can be type-based or topic-based due to different context window sizes. Bansal et al. (2014) clustered two word embeddings that are learned using context window size 1 and 10, respectively. The example clusters showed that, when context window size $W = 1$, syntactic words (e.g. his, your, her and its) are close to each other in the embedding space. On the contrary, topical words (e.g. financing, equity, investor and stock) are close to each other in the embedding space learned using context window size $W = 10$.

Following the aforementioned work, we study our Spanish Twitter word embeddings that were learned with dimensionality $D = 200$ and context window sizes $W = \{1, 5\}$. Using the *cosine* similarity defined in Eq. (1), we retrieve the most similar words from the embedding space given a query word. Examples are shown in Table 6 using two query words "futbol (football)" and "venezuela". We observe that the closest words to "futbol" are mostly the names of football players when $W = 1$. However, for $W = 5$, we note that some topical words such as "despidieron (dismiss)" and "banquillo (bench)" are also

Table 6 Example words retrieved using *cosine* similarity for embeddings with window size $W = 1$ and $W = 5$

W	Query	Retrieved words
$W1$	futbol	hokey, messi, suarez, #barcelonavsrealmadrid, neymar, ronaldo
	venezuela	#envide, oposicion, #vzlanoesunaamenaz, @ntn24ve, #angelustim, @fholland
$W5$	futbol	lionel, segun, messi, despidieron, banquillo, #barcelonavsrealmadrid
	venezuela	venezue, @delgadoantoniom, #concluvzla, venezeula, vzla, veneuela

included. For the query word "venezuela", the retrieved words are mostly hashtags and Twitter handles for context window $W = 1$, while some typos and abbreviations (e.g. "veneuela" and "vzla") of "venezuela" appear in the retrieved words for $W = 5$. This clearly shows that a large context window size (e.g. $W = 5$) allows word embedding models to learn relationships of two words that are more topic-based than a small context window size (e.g. $W = 1$). Therefore, in our examples, the abbreviations and typos of "venezuela" can be related in the embedding space.

Our findings together with the observations from Bansal et al. (2014) and Mitra et al. (2016) are helpful to explain the contradiction between the results of the NER task (Godin et al. 2015) and our Twitter election classification task. In the NER task, the classifier aims to identify named entities (e.g. names of persons, organisations and locations), which are often composed of a few adjacent words or the same type of words. Therefore, a word embedding model learned with a small context window benefits the NER task by relating the words having the same type (e.g. relating "futbol" to football players' names as shown in Table 6). In our Twitter election classification task, the CNN classifiers aim to classify tweets by the topics within their content. As such, a word embedding model learned with a large context window helps the CNN classifiers to capture the semantic information, the typos and even the abbreviations about the same topic.

In summary, for the Twitter election classification task using CNNs, word embeddings with large context window and dimension size can outperform all of our baselines, including the state-of-the-art text classifier *fastText*, in particular, achieving a statistically significant improvement over the classic classification baseline of SVM with TF-IDF for the Venezuela dataset. Therefore, we answer **RQ1**, that for our Twitter election classification task, a large context window size is preferred which is different to that from other tasks, e.g. Dependency parsing (Bansal et al. 2014) and NER (Godin et al. 2015).

6 Effect of the background corpora

Due to the noisy nature of Twitter data, Twitter posts can often be poor in grammar and spelling. Meanwhile, Twitter provides more special information such as Twitter handles, HTTP links and hashtags which would not appear in common text corpora. In order to study **RQ2** and infer whether the type of background corpus could affect the Twitter classification performance, we compare the background corpora of `es-Wiki`, `es-Twitter-general` and `es-Twitter-time` for the Venezuela dataset. We compare `en-Wiki`, `en-Twitter-general` and `en-Twitter-time` for the Philippines dataset. By considering the various experimental results reported in Sect. 5, we set the context window $W = 5$ and the dimensionality $D = 500$ for the word embeddings used in this section since they have demonstrated good performance for the CNN classifiers.

6.1 Types of background corpora

Before we show the effect of the types of background corpora, we first compare the word embedding models trained from Wikipedia corpus and Twitter corpus. Take the Spanish word embeddings as an example, the pairwise comparison (Table 7) between `es-Wiki` and `es-Twitter-general` shows vocabulary difference of the word embedding models. As we show the salient statistics of the two background corpora in Table 2, 66% of the vocabulary of our Venezuela election dataset appear in the word embedding model trained from `es-Twitter-general` while only 51% appear in `es-Wiki`. By removing the words shared by both embedding models, we observe that 1527 unique words are covered by `es-Twitter-general` but not covered by `es-Wiki` from Table 7. However, there are only 26 unique words that are covered by `es-Wiki` only. In Table 7, `es-Twitter-general` versus `es-Wiki` categorises the words only found in `es-Twitter-general`, which are mostly words unique to Twitter, such as Twitter handles and hashtags. The other 374 words are mainly incorrect spellings and elongated words such as “bravoooo”, “yaaaa” and “urgenteeeee”, which occur more often in Twitter than in other curated types of data such as Wikipedia. Our initial study on the vocabulary coverage shows that when the type of background corpus aligns with our Twitter classification task, it can potentially cover more unique terms that often occur in Twitter.

The classification results are shown in Table 8, where the first column shows the dataset we used. In other columns, we report three measures for embedding models trained from two types of background corpora `es-Wiki` and `es-Twitter-general` for the Venezuela election dataset and `en-Wiki` and `en-Twitter-general` for the Philippines

Table 7 Statistics of the pair comparison in unique vocabulary of Spanish background corpora (e.g. `es-Twitter-general` versus `es-Wiki` shows unique words only covered by `es-Twitter-general` compared to `Wiki`)

	Twitter handles	Hashtags	Others	Total
`es-Twitter-general` versus `es-Wiki`	818	225	374	1,527
`es-Twitter-general` versus `es-Twitter-time`	156	43	378	577
`es-Twitter-time` versus `es-Wiki`	731	437	312	1,480
`es-Twitter-time` versus `es-Twitter-general`	69	145	60	274
`es-Wiki` versus `es-Twitter-general`	0	0	26	26
`es-Wiki` versus `es-Twitter-time`	0	0	282	282

Table 8 Classification results by using different background corpora on Venezuela and Philippines datasets

Dataset	`es-Wiki`$^{=}$			`es-Twitter-general`$^{=}$		
	P	R	F1	P	R	F1
Venezuela	**81.6**	70.7	75.8	80.9	**72.2**	**76.3**
Philippines	**80.7**	**80.6**	**80.6**	80.4	80.5	80.4

$^{=}$ indicates that the difference between Wikipedia and Twitter corpus is not significant election dataset. For each dataset, the best scores are highlighted in bold. From Table 8, we observe that when the type of background corpus aligns with the Twitter election dataset, the performance is better for the Venezuela dataset in terms of recall and F1 scores. However, the performances of the two word embedding models on the Philippines dataset are very similar. As such, we conduct the *McNemar's* test, which indicates that the difference between the two types of word embedding models is not significant when applied to our Twitter election classification task. The additional words learned by `es-Twitter-general` (as categorised in Table 7) do not significantly affect the classification performance.

6.2 Time periods of background corpora

We have shown in Sect. 6.1 that there is no significant difference between the two background data types. However, when the type of background corpus aligns with the dataset (e.g. Twitter data), will the covered time period affect the classification performance? To address this question, we first compare the performance of `es-Twitter-general` and `es-Twitter-time` where the latter one covers the election period of the Venezuela election dataset; similarly, for the Philippines dataset, we compare `en-Twitter-general` and `en-Twitter-time`.

For each dataset, the classification results are shown in Table 9. For the Venezuela dataset, `es-Twitter-time` slightly outperforms `es-Twitter-general` in recall and F1 scores. However, according to McNemar's test, this difference between the embedding models trained from `es-Twitter-general` and `es-Twitter-time` is not statistically significant, with a p-value greater than 0.05. For the Philippines dataset, as shown in Table 9, `en-Twitter-time` outperforms `en-Twitter-general` on all the metrics. Moreover, the *McNemar's* test confirms that the performance difference is significant between `en-Twitter-time` and `en-Twitter-general`. From Table 2, by comparing the number of tweets collected in both of the corpora of `en-Twitter-general` and `en-Twitter-time`, we notice that although the corpus `en-Twitter-time` has a fewer number of tweets and unique words after preprocessing, it has a similar word coverage on our Philippines election dataset. Thus, it indicates that for our particular

Table 9 Classification results of CNN classifiers using word embedding models with and without overlap with the election periods

Dataset	`es-Twitter-time`$^{=}$			`es-Twitter-general`$^{=}$		
	P	R	F1	P	R	F1
Venezuela	80.1	**73.3**	**76.7**	**80.9**	72.2	76.3
Dataset	`en-Twitter-time*`			`en-Twitter-general*`		
	P	R	F1	P	R	F1
Philippines	**82.9**	**82.4**	**82.7**	80.4	80.5	80.4

$^{=}$ Indicates that the difference between `es-Twitter-time` and `es-Twitter-general` is not significant

* Indicates the difference between `en-Twitter-time` and `en-Wiki` is significant

classification task a smaller background corpus is capable to capture most of the salient words to distinguish different classes.

Furthermore, from Table 7, we notice that compared to `es-Twitter-general`, `es-Twitter-time` covers more hashtags but fewer Twitter handles. By investigating the hashtags only covered by the `es-Twitter-time`, we note that they correspond to hashtags that were frequently used on Twitter during the 2015 Venezuela parliamentary election period, for example "#venezueladecide", "#vota6d", "#reporte6d" and "#venezuelacambio". Since some hashtags are only popular during the election period, by covering the time period of the election dataset, it allows the word embedding to provide more domain-based features. In particular, although `es-Twitter-time` contains fewer number of tweets, it shows comparable classification results on our Twitter election classification task and requires less time for training a word embedding compared to `es-Twitter-general`. For the Philippines dataset, by covering the election period, `en-Twitter-time` even outperforms `en-Twitter-general` significantly. In summary, in answer to **RQ2**, we find that aligning both the type and time period of background corpus with the classification dataset leads to better feature representations, and hence a more effective classification using the CNN classifier.

7 Out-of-vocabulary words

Out-of-vocabulary (*OOV*) words are those that appear in the dataset but not in the word embedding model, and therefore vector representations cannot be obtained for such words. Although a larger background corpus can help word embeddings to cover more words, *OOV* words can still appear in a Twitter dataset. Indeed, as reported in Table 2, the `es-Twitter-general` corpus can only cover 66% words in our Venezuela election dataset, for instance due to the occurrence of various hashtags and Twitter handles. In recent studies of word embeddings (Bojanowski et al. 2016; Kim 2014; Mitra et al. 2016), the *OOV* words appear in different datasets such as the German dataset, namely `DE-GUR350` (Zesch and Gurevych 2006), which is used to study the semantic relatedness of word pairs. *OOV* words were simply ignored by Bojanowski et al. (2016) and Mitra et al. (2016). On the contrary, Kim (2014) randomly initialised the vector representations of *OOV* words by sampling each dimension from a uniform distribution $U[-a, a]$. a was selected in such way that the initialised vectors have the same variance as the learned word embedding model. Since this strategy uses the statistics of the pre-trained word embedding models, we address **RQ3** in this section to study whether such kind of strategies improves the classification performance.

We use our word embedding models trained from `es-Twitter-general` and `en-Twitter-general` with the context window size $W = 5$ and dimensionality $D = 500$, since it was already shown in Sect. 5 to attain good performance. Five different strategies are used in this section:

1. `Random-full`: Following the strategy used by Kim (2014), we randomly initialise the *OOV* words using the calculated means and standard deviations from the entire pre-trained word embedding model.
2. `Random-local`: We randomly initialise the *OOV* words using the calculated means and standard deviations from only the vector representations of words that appear in the dataset.

3. `Zero-vector`: We apply a simple strategy similar to the one used by Bojanowski et al. (2016) and Mitra et al. (2016) to initialise the vector representations of *OOV* words as *n*-dimensional *zero* vectors.
4. `Random-pure`: We apply a purely random strategy to sample a number from the range (0, 1) for each dimension of the vector representations of *OOV* words.
5. `Random-norm`: We randomly initialise the *OOV* word using normal distribution and the means and standard deviations obtained from the word embedding model.

An alternative strategy would be the use of a character-based distributed representations approach (e.g. as proposed by Dhingra et al. (2016)) to address the *OOV* problem. However, in this paper, we omit this approach to focus exclusively on word-level embedding models. In particular, character-level approaches (e.g. character-level CNN) cannot reuse pre-trained word-level embedding models, such as those obtained from Twitter or Wikipedia. Therefore, we leave the character-level approaches to future work. The classification performances of the aforementioned retained strategies are compared in Table 10 over precision, recall and F1 scores for Venezuela and Philippines datasets.

From Table 10, we observe that indeed all the random initialisation strategies of `Random-full`, `Random-local`, `Random-pure` and `Random-norm` slightly improve the attained recall and F1 scores compared to the `Zero-vector` strategy – indeed, this conclusion agrees with the observation of Kim (2014), who observed slight improvements using a similar strategy for text classification. However, `Zero-vector` strategy has a slightly better precision over random strategies for the Venezuela dataset. Since `Random-local` has the best performance in terms of F1 score for the Venezuela dataset in Table 10, we conduct the statistical test between the two strategies of `Random-local` and `Zero-vector`. The result of the statistical test yields a *p*-value greater than 0.05, which shows that the difference between them is not statistically significant. For the Philippines dataset, we then conduct the *McNemar's* test to compare `Random-pure` and `Random-norm`, since they outperform the other strategies. The result of statistical test shows that the difference between the two strategies is not significant, with a *p*-value greater than 0.05. Hence, we conclude that, compared to simple strategies (e.g. `Zero-vector` and `Random-pure`), a more complicated random initialisation strategy that uses the means and standard deviations of word embeddings does not significantly improve the performance of CNN classifiers on both datasets.

Table 10 Classification results of CNN classifiers by using different *OOV* strategies

OOV strategy	Venezuela			Philippines		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
`Zero-vector`	**81.7**	71.1	$76.0^{=}$	81.3	79.0	80.1*
`Random-full`	80.9	72.2	$76.3^{=}$	80.4	80.5	80.4*
`Random-local`	81.5	72.2	**76.5**$^{=}$	80.8	79.5	80.1*
`Random-pure`	79.8	**73.2**	$76.3^{=}$	**82.5**	**81.3**	**81.8**$^{=}$
`Random-norm`	80.4	72.3	$76.1^{=}$	81.9	81.1	$81.4^{=}$

Highest score for each dataset is highlighted in bold

$^{=}$ Indicates that the difference between each strategy is not significant.

* Indicates the difference is significant compared to `Random-pure`

In summary, the simplest strategies (e.g. `Zero-vector` and `Random-pure`) are able to achieve comparable or better classification performance to `Random-norm` and `Random-full`, which shows simple strategies are sufficient for our Twitter election classification task. This answers our **RQ3** that there is no obvious preference between the different *OOV* strategies studied in this section (e.g. `Random-norm` and `Random-pure`), since their classification performances are comparable according to the statistical test.

8 Conclusions

Since previous investigations on the parameter configuration of word embeddings focus on different tasks such as named entity recognition (Godin et al. 2015) and dependency parsing (Bansal et al. 2014), their findings may not generalise to Twitter classification tasks. Meanwhile, similar work on Twitter classification tasks (Ebert et al. 2015; Severyn and Moschitti 2015; Tang et al. 2014) have not studied the impact of the background corpora and the *Word2Vec* parameters such as the context window and dimensionality. Our finding shows that these two factors can indeed affect the classification performance on Twitter classification tasks. In particular, in this paper, we studied word embeddings when using convolutional neural networks. Using two different types of background corpora, we observed that when the type and time period of background corpus align with the classification dataset, the CNN classifier can achieve significantly better performance on Twitter data (Sect. 6). In particular, our investigation showed that choosing the correct type of background corpus can potentially cover more vocabulary of the classification dataset. Thus, the alignment between the background corpus and the classification dataset provides better tweet-level representations. For inferring the best setup of the *Word2Vec* parameters (e.g. context window, dimensionality and negative samples), we applied word embeddings with various parameter setup to convolutional neural networks (Sect. 5). As a practical guide for a Twitter classification task, word embeddings with both large context windows and dimensions are preferable to attain high effectiveness with a convolutional neural network (CNN) classifier. In contrast, the number of negative samples does not affect the performance of a CNN classifier in our task. In addition, we show that there is no obvious winner among the current *OOV* strategies for our Twitter classification task using CNN classifiers and word embedding models (Sect. 7). Thus, the simplest random strategy of sampling each dimension of the *OOV* word vectors from range (0, 1) is sufficient to deal with the *OOV* words.

Acknowledgements This paper was supported by a grant from the Economic and Social Research Council (ES/L016435/1). The authors would like to thank the assessors for their efforts in reviewing tweets.

References

Amati, G., Amodeo, G., Bianchi, M., Marcone, G., Bordoni, F. U., Gaibisso, C., Gambosi, G., Celi, A., Di Nicola, C., & Flammini, M. (2011). FUB, IASI-CNR, UNIVAQ at TREC 2011 microblog track. In: *Proceedings of TREC*.

Bansal, M., Gimpel, K., & Livescu, K. (2014). Tailoring continuous word representations for dependency parsing. *Proceedings of ACL*, *2*, 809–815.

BBC. (2015). Venezuela opposition politician luis manuel diaz killed. http://www.bbc.co.uk/news/world-latin-america-34929332. Accessed 15 May 2016.

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*, 1137–1155.

Bermingham, A., & Smeaton, A. F. (2011). On using Twitter to monitor political sentiment and predict election results. In: *Proceedings of SAAIP workshop at IJCNLP*.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606.

Cohen, J. (1968). Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin*, *70*(4), 213.

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In: *Proceedings of ICML* (pp. 160–167).

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *12*, 2493–2537.

Conneau, A., Schwenk, H., Barrault, L., & Lecun, Y. (2016). Very deep convolutional networks for natural language processing. arXiv preprint arXiv:1606.01781.

Dhingra, B., Zhou, Z., Fitzpatrick, D., Muehl, M., & Cohen, W. W. (2016). Tweet2vec: Character-based distributed representations for social media. In *Proceedings of ACL*.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, *10*(7), 1895–1923.

Ebert, S., Vu, N.T., & Schütze, H. (2015). CIS-positive: Combining convolutional neural networks and SVMs for sentiment analysis in Twitter. In: *Proceedings of SemEval* (p. 527).

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In: *Proceedings of AISTATS* (Vol. 15, p. 275).

Godin, F., Vandersmissen, B., De Neve, W., & Van de Walle, R. (2015). Multimedia Lab@ ACL W-NUT NER shared task: Named entity recognition for Twitter microposts using distributed word representations. In: *Proceedings of ACL-IJCNLP* (p. 146).

Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., & Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, *405*(6789), 947–951.

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In: *Proceedings of EMNLP* (pp. 1746–1751).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In: *Proceedings of NIPS* (pp. 1097–1105).

Macdonald, C., McCreadie, R., Santos, R.L., & Ounis, I. (2012). From puppy to maturity: Experiences in developing terrier. In: *Proceedings of OSIR workshop at SIGIR* (Vol. 60).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mikolov, T., Le, Q. V., & Sutskever, I. (2013b). Exploiting similarities among languages for machine translation. arXiv preprint arXiv:1309.4168.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In: *Proceedings of NIPS* (pp. 3111–3119).

Mikolov, T., Yih, W. T., & Zweig, G. (2013d). Linguistic regularities in continuous space word representations. In: *Proceedings of HLT-NAACL* (Vol. 13, pp. 746–751).

Mitra, B., Nalisnick, E., Craswell, N., & Caruana, R. (2016). A dual embedding space model for document ranking. arXiv preprint arXiv:1602.01137.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In: *Proceedings of EMNLP* (pp. 1532–1543).

Severyn, A., & Moschitti, A. (2015). UNITN: training deep convolutional neural network for Twitter sentiment classification. In: *Proceedings of SemEval* (pp. 464–469).

Severyn, A., Nicosia, M., Barlacchi, G., & Moschitti, A. (2015). Distributional neural networks for automatic resolution of crossword puzzles. In: *Proceedings of ACL-IJCNLP*.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Tang, D., Qin, B., & Liu, T. (2015). Document modeling with gated recurrent neural network for sentiment classification. In: *Proceedings of EMNLP* (pp. 1422–1432).

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., & Qin, B. (2014). Learning sentiment-specific word embedding for Twitter sentiment classification. *Proceedings of ACL*, *1*, 1555–1565.

Wang, P., Xu, J., Xu, B., Liu, C. L., Zhang, H., Wang, F., et al. (2015). Semantic clustering and convolutional neural network for short text categorization. *Proceedings of ACL-IJCNLP*, *2*, 352–357.

Zesch, T., & Gurevych, I. (2006). Automatically creating datasets for measures of semantic relatedness. In: *Proceedings of linguistic distances workshop at ACL* (pp. 16–24).

Zhang, X., & LeCun, Y. (2015). Text understanding from scratch. arXiv preprint arXiv:1502.01710.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In: *Proceedings of NIPS* (pp. 649–657).

Zhang, Y., & Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820.