Multi-Target Stance Detection Based on GRU-PWV-CNN Network Model

Wenfa Li[1,2], Yilong Xu[3], Gongming Wang[4,5]

[1] Institute of Scientific and Technical Information of China, China
[2] College of Robotics, Beijing Union University, China
[3] Beijing North Great Wall Photoelectric Instruments Co., Ltd., China
[4] Beijing Tianyuan Network Co., Ltd., China
[5] Inspur Software Group Co., Ltd., China
644085325@qq.com, 312551615@qq.com, gongmingwang@126.com

Abstract

To uncover opinions on different people and events from text on the internet, stance detection must be performed, which requires an algorithm to mine stance tags for different targets (people or events). Some text contain multiple targets, and the content describing different targets is related, which results in poor stance detection performances. Therefore, stance detection for such data is defined as multi-target stance detection. To address this issues, a network model composed of a gated recurrent unit, a position weight vector, and a convolutional neural network (GRU-PWV-CNN) is proposed. First, the bidirectional GRU (Bi-GRU) is employed to extract the unstructured features, and a position-weight vector is designed to represent the correlation between every word and the given target. Next, these two forms of information are fused and transmitted to a CNN to complete the secondary extraction of features. Finally, a softmax function is used to carry out the final classification. A multi-target stance detection corpus for the 2016 US election was used to compare the performances of our method and other methods, including the Seq2Seq and AH-LSTM. The experimental results showed that the proposed method achieved well and had a 2.82% improvement in the macro-averaged F1-score.

Keywords: CNN, GRU, Position-weight vector, Multi-target, Stance detection

1 Introduction

In recent years, with the rise of the mobile Internet, many online social applications have developed rapidly. Social media services such as Twitter have become the main source of public opinion analysis on the Internet, due to its high degree of openness, high levels of interaction, and rapid information dissemination. In particular, public opinion about subjects such as products and events is an important source of feedback that can help decision makers understand public opinions in real time [1].

To analyze text containing public comments automatically, a stance detection task for social text was proposed in 2016 [2-3]. This is defined as automatically analyzing text and determining whether the text expresses a “favor,” “against,” or “neither” attitude for given target [4]. Unlike sentiment analysis, which has been widely used for public opinion analysis, stance detection only mines the target-related stance from the text. Therefore, it is possible to obtain more valuable tags when analyzing opinions. Thus, many researchers have used stance detection to analyze political topics [5] and fake news [6].

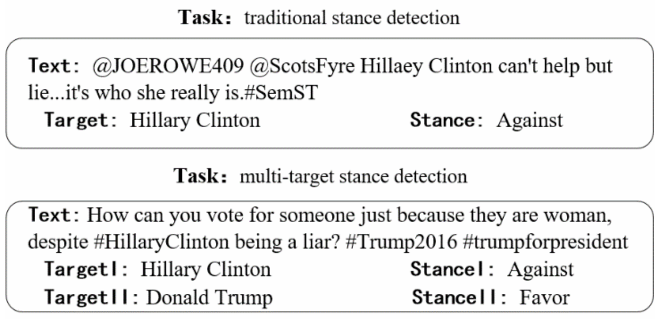
Some text contains multiple targets, which often appears when discussing candidates for elections or different brands of the same product. In addition, the targets in this type of data are related, which leads to similar text content describing different targets. Consequently, it is more difficult for the algorithm to mine the target-related stance. As a result, traditional stance detection algorithms achieve lower accuracy when processing such text. Therefore, researchers began to discuss stance detection for these types of data separately, which is defined as a multi-target stance detection task [7].

Traditional and multi-target stance detection examples are shown in Figure 1. In the traditional stance detection task, the text only mentions one target (e.g., Hillary Clinton), and the algorithm also only detects the stance category of this target. However, in multi-target stance detection, the text mentions two or more targets (e.g., Hillary Clinton and Donald Trump), which requires the algorithm to find the stance categories of multiple targets from one text passage. Therefore, for multi-target stance detection, the algorithm must complete two tasks: (1) detect the

*Corresponding Author: Gongming Wang; E-mail: gongmingwang@126.com

DOI: 10.3966/160792642021052203009

stance label of each target in the same way as the traditional stance detection task, and (2) find the content describing the different targets. Because the targets in such tasks are related, the contents describing different targets are similar, which increases the difficulty of the stance detection process.


Task: traditional stance detection

Text: @JOEROWE409 @ScotsFyre Hillaey Clinton can't help but lie...it's who she really is.#SemST
Target: Hillary Clinton **Stance:** Against

Task: multi-target stance detection

Text: How can you vote for someone just because they are woman, despite #HillaryClinton being a liar? #Trump2016 #trumpforpresident
TargetI: Hillary Clinton **StanceI:** Against
TargetII: Donald Trump **StanceII:** Favor

Figure 1. Examples of traditional stance detection and mul-ti-target stance detection

In this paper, a network model composed of a gated recurrent unit, position weight vector, and convolutional neural network (GRU-PWV-CNN) is proposed for multi-target stance detection. The model contains three main parts: a gated recursive unit (GRU), a position-weight vector (PWV), and a convolutional neural network (CNN). The GRU is used to obtain features from the text data, the PWV is used to represent the position relationship between each word in the text and each target to help the algorithm distinguish the content describing the different targets, and the CNN is used to re-extract the information obtained from the other two parts. The multi-target stance detection corpus of the 2016 US election was used to validate our method. The experimental results showed that this method has advantages over other algorithms.

2 Related Work

The related research is introduced from two aspects: A. traditional stance detection, and B. mul-ti-target stance detection.

2.1 Traditional Stance Detection

The task of stance detection for social media (traditional stance detection) was proposed because the overall emotion of a text passage does not fully express its attitude toward the particular subject when using sentiment analysis to analyze public opinion. The stance detection task requires mining the stance tags for a specific target in the text, rather than the attitude of the entire text, which makes it possible to obtain more valuable results, especially in the field of public opinion analysis [8].

At a technical level, the difficulty of stance detection is determining how to allow the algorithm to selectively extract features from text based on the target information. Early researchers used machine/deep learning methods (such as a support vector machine (SVM) [9], a convolutional neural network (CNN) [10], and long short-term memory (LSTM)) to find the relationships between text and labels through training data.

Augenstein et al. [11] proposed a bidirectional conditional encoding LSTM model. This method used the last hidden state of the LSTM of the processed target as the initial state of the LSTM of the processed text. These researchers hoped to combine the LSTM's ability for processing time series data to transmit the target information to the network that processes the text, so that the algorithm could detect the stance label. Dey et al. [12] proposed a two-stage LSTM model from the perspective of a classifier. This model decomposed the three-category stance detection task into two subtasks: (1) distinguishing whether there was a stance label, and (2) judging whether it was "favor" or "against". This model subdivided the processing tasks of a single network model, thereby reducing the difficulty of stance detection.

Researchers subsequently began to introduce the fusion network model [13] and the attention mechanism [14] into the stance detection task. The former combined the characteristics of multiple network models to increase the algorithm's ability to mine features. The latter could selectively retain the features mined from the text, which is in line with the requirement of the stance detection task to shield information that is not related to the target.

Li et al. [15] designed a fusion network by combining a CNN model and a GRU model. Although the CNN had excellent down-sampling capabilities, it could not analyze time series data. Therefore, GRU was combined with the CNN to supplement the ability of the CNN to extract time series features. Chopra et al. [16] used an attention mechanism to increase the connection between the network that processed the target and the network that processed the text, thereby enhancing the algorithm's ability to mine target-related features.

Some researchers hoped to improve the effectiveness of stance detection through external information or data [17]. One of the most used types of data is the emotion tag, because the stance detection task is derived from the sentiment analysis task, and the labels of the two tasks have a certain similarity [4].

Ebrahimi et al. [18] proposed a stance classification model based on the correlation between stance tags and emotion tags. This method added an emotional dimension into the relationship between the target, text, and stance label, thereby improving the effect of stance detection. Zarrella et al. [19] obtained additional data from the Twitter platform and used the transfer learning method to learn from these external data to improve the detection effect.

2.2 Multi-target Stance Detection

With the application of stance detection technology, researchers have found a type of text that contains multiple targets. This text appears commonly when discussing different candidates in an election or different brands of a product. In addition, the corpora structures and stance labels are more complicated, which causes traditional stance detection methods to perform poorly for such data. Therefore, this type of data greatly affects opinion mining based on traditional stance detection technology in various fields, such as election predictions and product feedback. Therefore, the multi-target stance detection problem is defined as a sub-task of stance detection [7].

Some researchers believed that multi-target stance tags are relevant (e.g., in election-related corpora, few people support multiple candidates simultaneously, and most support only one or none). Therefore, to consider the mutual influence of the stance labels of the different targets, the algorithm they designed considered the stances of other targets when detecting the stance of one target. Therefore, this algorithm typically predicted the stances of multiple targets simultaneously [20].

Sobhani et al. [21] supposed that there was a certain relationship between the stances of multiple targets. Therefore, the Seq2Seq model, which is widely used in text translation and other fields, was used to detect multi-target stances. When predicting a stance, this method considered the previous predictions to detect the stance of the current target. Thus, this method could not only consider the relationship between many labels simultaneously, it also produced multiple outputs for multi-target stance detection. Wei et al. [22] designed an extended network. This network was shared when predicting labels for different targets. Therefore, this method could combine the contents of multiple tags to detect the stances of multiple goals.

Other researchers viewed multi-target stance detection as complex stance detection task. Therefore, the input and output of such algorithms are in the form of single target stance detection (the target and text are input, and the target's stance is predicted). Among these methods, expanding the fusion network and region segmentation are commonly used methods. The former can enhance the ability of the model to adapt to different goals, and the latter can segment the input text containing multiple goals to help the algorithm divide the content describing different targets.

Siddiqua et al. [23] used a network model containing multiple CNN models, attention mechanisms, and LSTMs to complete multi-target stance detection. This model had a larger size and more diverse structures, which made it more adaptable when dealing with complex problems. Liu et al. [24] designed an unsupervised range extraction method to divide text containing two targets into four parts. In this way, content describing different goals was separated to reduce the impact between them.

However, expanding a fusion network not only increases the time complexity, it also enlarges the risk of algorithm overfitting while increasing the adaptability. Although the region segmentation method of Liu et al. [24] reduces the risk of interactions between different targets, it was designed to only segment text with two targets, and the given four parts cannot adapt to other types of text passages. Thus, this method has low robustness.

Aiming at the shortcomings of this region segmentation method, a position-weight vector was designed. Text describing different targets is divided into multiple sub-text passages, which is a different approach from that of the segmentation method. The position-weight vector represents the positional relationship between the each target and each word in the text, and it is embedded into the deep learning model, which makes it possible to highlight text that describes different goals while maintaining the integrity of the text. This method is introduced in the next section.

3 Method

The architecture of our model is shown in Figure 2. It consists of the following five modules: an embedded layer, a Bi-GRU layer, a position-weight fusion layer, a CNN, and a softmax classifier and model training module. The input of the model is the text of the target and the subject (target + text).

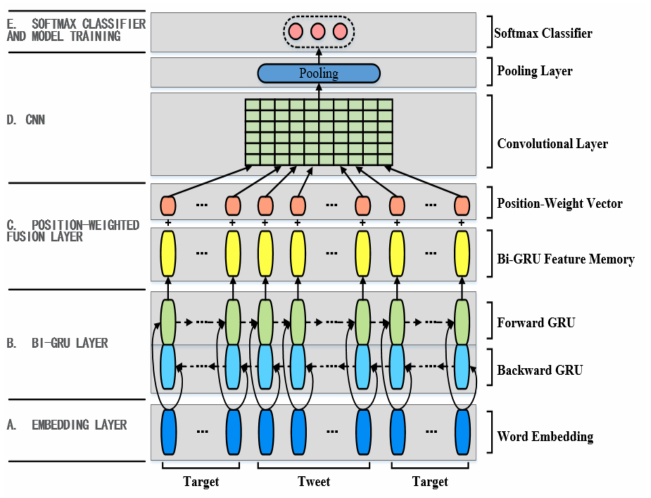


Figure 2. Model architecture

3.1 Embedding Layer

The purpose of this layer is to convert words in the text into corresponding word embeddings. It converts a word of natural language into a vector with a specific meaning so that the algorithm can better analyze the potential semantics. In this process, each word will be sequentially converted into a 1 × d vector. Thus, the entire text is converted into a *l* × d tensor, where *l* is the text size, and d is the dimension of the word vector.

3.2 Bi-GRU Layer

To extract valid features from the unstructured text, a GRU is used to encode the text. The GRU can describe the long-distance lexical dependency in the text and is suitable for text data modeling [25]. The input of the GRU is a tensor formed by arranging the embedded vectors of the words to be processed in order from front to back. The corresponding output is a tensor composed of implicit states of the GRU units in order from front to back.

However, the traditional one-way GRU cannot describe the dependence between words in the direction from back to front. Therefore, a bidirectional GRU (Bi-GRU) composed of a forward GRU and a backward GRU is used. It has been proven that using bidirectional encoding technology to process time-series data is very effective [26].The input of the forward GRU network is composed of the embedded vectors of words arranged in order from front to back, and the input of the backward GRU network is a series of the embedded vectors of words arranged in the opposite order. They are designed to mine the textual features based on the past and future contexts, respectively. After this, the outputs of the forward and backward GRUs at each unit are spliced. Thus, the Bi-GRU can describe the dependence relationship between words in the directions from front to back and back to front. Therefore, each input word of Bi-GRU will be first transmitted to the forward GRU and then to the backward GRU, and the output of the Bi-GRU is the spliced result of the hidden states of the above two GRUs.

In the GRU model [27], a unit t is calculated as follows:

$$z_t = \sigma(x_t U^z + h_{t-1} W^z) \quad \textbf{(1)}$$

$$r_t = \sigma(x_t U^r + h_{t-1} W^r) \quad \textbf{(2)}$$

$$s_t = \tan h(x_t U^s + (h_{t-1} \cdot r_t) W^s) \quad \textbf{(3)}$$

$$h_t = (1 - z_t) \cdot s_t + z_t \cdot h_{t-1} \quad \textbf{(4)}$$

where $x_t \in R^d$ represents the word embedding vector entered at position t, and $h_t \in R^n$ is the hidden state at position t, $s_t \in R^n$ is the output at position t. Moreover, z_t and r_t are two gates (update gate and reset gate, respectively) at position t, σ and $\tan h$ represent sigmoid and tanh activation functions, and $U \in R^{d\times n}$ and $W \in R^{n\times n}$ are weight matrices. For R^d, R^n, and $R^{d\times n}$, d is the dimension of the word vector, and n is the size of the output of the GRU network.

In the Bi-GRU, the hidden states of the forward and backward GRUs at each moment t are concatenated and sent to the output. This process is shown in Figure 3.

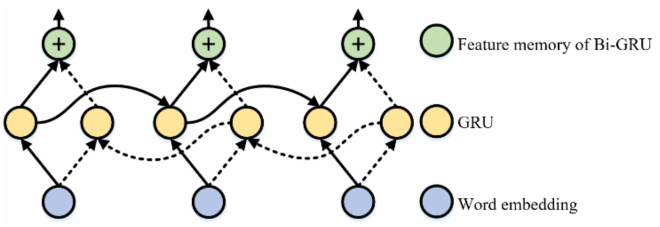


Figure 3. Model architecture of the bidirectional GRU, where the solid line represents the forward GRU, the dashed line represents the reverse GRU, and + represents the cascade

3.3 Position-weight Fusion Layer

When using the Bi-GRU to extract unstructured features, it is difficult to distinguish the contexts that describe different targets. Thus, in this section, a position-weight vector is proposed to represent the positional relationship between each word and the different targets. This vector is then fused into the feature tensors extracted by the GRU, which enlarges the differences of the text describing different targets, thereby helping this algorithm to distinguish the contexts describing different targets.

3.3.1 Position-weight Vector

The position-weight vector is designed to extract the positional relationship between each word in the text and the different targets. In this process, it is assumed that in the text, words that are closer to the target word have a greater influence on the target, and conversely, words that are farther away have a smaller influence. This distance is defined as the number of words in between words. The specific calculation method is as follows.

w_k^m represents the impact weight of the k-th word on the m-th target, which is expressed as follows:

$$w_k^{\max} = 1 - \frac{|k - \tau_m|}{k_{MAX}} \quad \textbf{(5)}$$

where τ_m is the serial number of the m-th target word ($1 \le m \le m_{MAX}$, m_{MAX} is the number of times that the target appears in the text), which represents the position of the m-th target word in the text, and k_{MAX} is the number of words included in the entire text. To deal with the case where one target appears multiple times in the text, the maximum of w_k^m among all the targets is taken as the maximal weight of the k-th word, which is expressed as $w_k^{\max}$:

$$w_k^{\max} = \max\{w_k^1, w_k^2, ..., w_k^m, w_k^{m_{MAX}}\} \quad \textbf{(6)}$$

All the maximum weights $w_k^{\max}$ are then filled into the position vector E, which is shown as follows:

$$E = \{w_1^{\max}, w_2^{\max} ..., w_{k_{MAX}}^{\max}\} \quad \textbf{(7)}$$

At this point, each component of vector E (where each element ranges from 0 to 1) represents the impact of each word on the target, as shown in Figure 4.

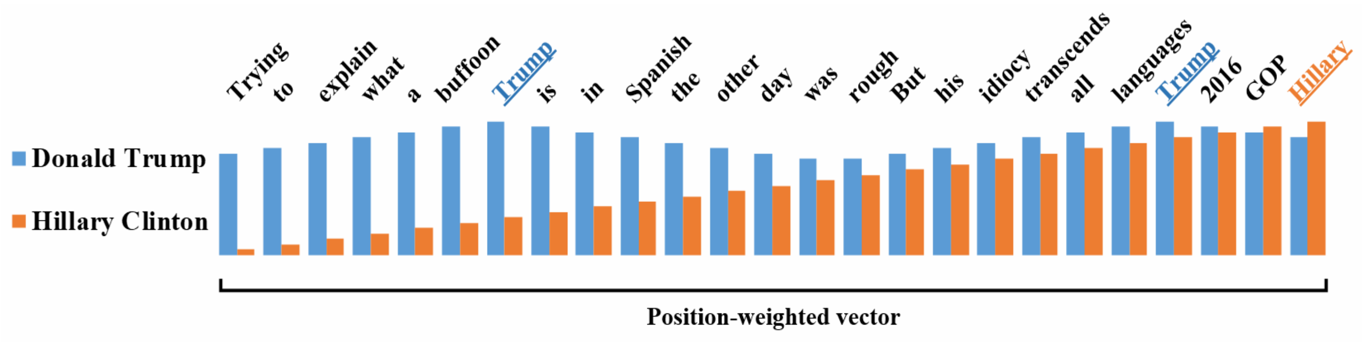


Figure 4. Position-weight vector of two targets in the same text

In the model, the two main parts are the Bi-GRU for mining unstructured features and the PWV for mining linguistic features. Before fusing their outputs, their working processes are independent. Thus, it is necessary to discuss the effects of the two types of characteristics on predicting the stance. To control the effect of the two types of features on the entire system, each component of vector E is expanded by multiplying by a factor of μ. This factor can adjust the ratio of the output tensors of the Bi-GRU and PWV. Therefore, the effect of the position weight on the entire system can be changed by adjusting μ. When the value of μ increases, it indicates that the influence of the position vector on the entire model increases, and the influence of the Bi-GRU decreases, as follows:

$$E_\mu = E \times \mu \quad \textbf{(8)}$$

where E_μ is the position-weight vector (where each element ranges from 0 to μ). Each component of E_μ is composed of two parts: one is the position-weight $w_k^{\max}$ in E, indicating the impact of the each word on the target, and the other is the coefficient μ, expressing the impact of the position-weight $w_k^{\max}$ on the entire system.

The coefficient μ is defined as a trainable variable in the process of building model, which is suitable for finding the global optimum value of the model. The coefficient μ is regarded as a parameter rather than a hyper-parameter. The training process of μ is the same as other parameters (weights and biases) in the model. The value one is recommended to initialize this coefficient and a learning rate of 0.05 is used to train this coefficient. In addition, in order to make this coefficient have a better effect, the L2 regularizer [28] is added to prevent it from overfitting.

3.3.2 Concatenating Position-weight Vector and Output of Bi-GRU

After the position-weight vector is calculated, it should be fused with the feature tensor extracted by the Bi-GRU. In this process, the feature tensor extracted by the Bi-GRU is concatenated with the corresponding element in the position-weight vector. This concatenated tensor is $Y \in R^{l\times(2\times n+1)}$, where n is the size of the output of a unidirectional GRU, $2\times n+1$ represents the dimension of the output of the Bi-GRU plus the corresponding position weight, and *l* is the size of the text passage to be sent to the CNN, which is no less than k_{MAX} in Equation (5). This tensor can not only describe the dependence between words in the different directions, but it can also describe the impact of a word on the different targets.

3.4 CNN

CNN networks are less used than RNNs when processing time series data. Although a CNN can extract local one-dimensional plaques (subsequences) from a sequence and recognize the local patterns within a convolution window, it cannot mine the logic relationship between words in a context like RNNs can [29]. However, the context logic relationship has been extracted through the Bi-GRU and position-weight vector, and the subsequent operation is to re-extract these extracted features. Thus, a CNN was chosen to complete the re-extraction process. The CNN model is shown in Figure 5.

There are two components of the CNN: the convolution layer and the pooling layer

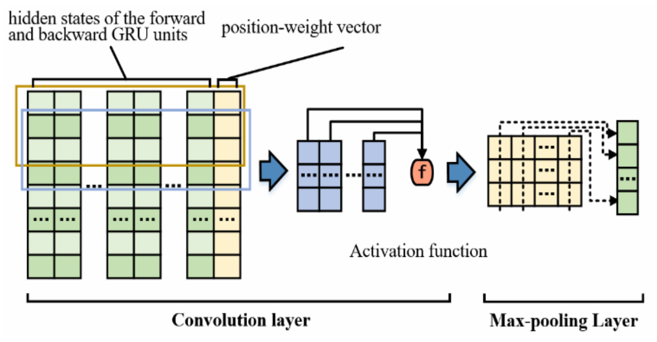


Figure 5. Model architecture of CNN

3.4.1 Convolution Layer

In the convolution layer, the input is the feature memory stitched together from the previous layer arranged in time. This feature memory is $Y \in R^{l\times(2\times n+1)}$, where n is the size of the output of the GRU network, 1 represents the dimension of the corresponding position weight, and *l* represents the size of the text.

The CNN network will then use J × K filters (convolution kernels) to convolve the input, where J is time domain length of the convolution kernel, and K is input dimension (K = $2\times n+1$) and the number of filters [10]. After this, activation functions are used to process these features, and the result is sent to the pooling layer.

3.4.2 Pooling Layer

A pooling layer is commonly used to reduce the features mined by the network model and is often placed behind the convolutional layer. Therefore, after the feature is extracted by the convolutional layer, the pooling layer is used to extract features again to obtain more critical elements. Global max-pooling is used in this layer.

3.5 Softmax Classifier and Model Training

The output of the CNN layer is taken as the input of this layer, and the softmax classifier [30] is used to predict the stance labels of the different targets. It consists of multilayer perceptron and softmax activation function. In addition, loss function is the minimum cross entropy, which is shown as follows:

$$H(y,\hat{y}) = -\sum_{i=0}^{K} y_i \log \hat{y}_l \quad \textbf{(9)}$$

where y is the true result, $\hat{y}$ is the output of the softmax classifier (the dimensions of y and $\hat{y}$ are K), and y_i and $\hat{y}_l$ are the i-th elements of y and $\hat{y}$, respectively.

During the training process, we save the model after each epoch. And then, the model that has the best effect in the development set among these models is chosen as the output model.

4 Experiments

4.1 Experimental Setting

4.1.1 Data, Data Preprocessing, and Hyper-parameter Settings

Dataset: The multi-target stance detection corpus provided by Sobhani et al. was used to verify the effectiveness of our model [7]; it contained text from Twitter discussing candidates in the 2016 US election. This corpus contained three sub-datasets corresponding to three groups of targets (Clinton–Sanders, Clinton–Trump, and Cruz–Trump). Table 1 shows the distribution of the training set, development set and test set in each sub-dataset.

Table 1. Details of the experimental datasets

Target	Train	Dev	Test	Total
Clinton–Sanders	957	137	272	1366
Clinton–Trump	1240	177	355	1722
Cruz–Trump	922	132	263	1317
Total	3199	446	890	4455

Data preprocessing: There were many misspellings (e.g., “Hilary”) and word concatenations (e.g., “Trumpforpresident”) in this dataset, because the language of social media is not usually standardized, and the corpus producer is not concerned with processing noisy text. Therefore, we simply normalized all the target words in the text. For example, “Hilary” was modified to “Hillary” and “Trumpforpresident” was modified to “Trump for president”.

Text size: The structure of the model was fixed, but the number of words in each text passage was different. Thus, a fixed size should be set to define our model. We choose the length of the text containing the most words in the data set as the text size.

Word embedding: A 300-dimensional pre-trained word embedding from fastText [31] was used to complete the feature representation.

Hyper-parameters: The optimization method is Adagrad, the learning rate is 1e-3, the minimum batch is 16, loss function is the minimum cross entropy，and the epoch is 20. The GRU output is 256, the kernel size is 5, the number of filters is 513, the pooling method is max-pooling, and all the parameter matrices values are initialized by Xavier [32]. In addition, the model that achieved the best results in the development set is selected to validate in the test set.

4.1.2 Evaluation Method

The stance detection task was more inclined to achieve a better prediction for “Favor” and “Against” labels. Therefore, the average of F1-score was chosen to evaluate the effect of the stance detection task, which is defined as follows:

$$F_{avg} = \frac{F_{FAVOR} + F_{AGAINST}}{2} \quad \textbf{(10)}$$

where F_{FAVOR} and $F_{AGAINST}$ are the F1-score that support and do not support the target, respectively.

In addition, in order to evaluate the prediction effect of multiple targets in a sub dataset, the average of F_{avg} of all targets in one dataset is used to evaluate the effectiveness in the corresponding dataset, which is called “Avg” for short. To evaluate the effectiveness of the model on different sub-datasets, the macro-averaged F1-score [21] that is the average of “Avg” in all sub-dataset is selected to evaluate the performance of the model, which is called “F-Macro” for short. In the following section of this paper, the sign “Avg” and “F-Macro” are used to represent the above two evaluation indicators.

4.1.3 Other Methods

To demonstrate the performance of our algorithm, some methods of multi-target stance detection were compared, which is described as follows.

Sequence-to-sequence (Seq2Seq) [21]: Recently, the Seq2seq model has achieved good performance when dealing with timing problems. Therefore, Sobhani et al. applied this model to the multi-target stance detection problem. In this method, text is used as the input of the model, and the stance labels representing different targets are the output. The advantage of this algorithm is that it not only mine the stance related to the target from the text, but also consider to the relationship between multiple targets.

Attention-Based Hierarchical LSTM (AH-LSTM) [24]: In 2018, this method was proposed by Liu et al. for multi-target stance detection. First, based on the position of the target in the text, the text containing multiple targets is split into four parts. The corresponding target and text word embeddings are then spliced based on these parts, and they are processed with four separate LSTMs. After this, the outputs of the four LSTMs are input into a Bi-LSTM network with an attention mechanism. Finally, the stance of the tweet is detected by a softmax classifier that accepts the output of the Bi-LSTM.

4.2 Results and Discussion

To evaluate the effectiveness of the proposed method, it was compared with the methods introduced in Section 4.1.3 first. After this, the effects of using a Bi-LSTM and GRU to mine the unstructured features were compared. The impact of the position-weight vector on the system was then discussed. Finally, the effects of using different methods to complete feature re-extraction were analyzed.

4.2.1 Comparison Between Our Method and Related Methods

The comparison with other methods (as described in Section 4.1.3) are shown in Table 2.

Table 2. Performances of our approach and other methods (Unit: %)

Methods	Clinton-Sanders			Clinton-Trump			Cruz-Trump			F-Macro
	Clinton	Sanders	Avg	Clinton	Trump	Avg	Cruz	Trump	Avg	
Seq2Seq	55.59	53.86	54.72	54.46	58.74	55.60	47.02	59.21	53.12	54.81
AH-LSTM	52.26	58.38	55.32	60.63	61.98	61.31	52.88	50.30	51.59	56.07
GRU-PWV-CNN	55.45	58.65	57.13	60.03	66.17	63.10	52.88	60.02	56.45	58.89
Improvement with our method (%)	-	-	1.81	-	-	1.89	-	-	3.33	2.82

Note. The items in columns “Clinton”, “Sanders”, and “Trump” are all the F_{avg} and directly cited from the experimental results in the corresponding references.

By comparing the Avg and the F-Macro of the different methods, the performances is ranked as follows: Seq2Seq < AH-LSTM < GRU-PWV-CNN. From this result, we found the following results.

1. Seq2Seq method placed third in terms of performance. Although this method has the ability to detect positions with reference to different targets, it does not take into account the effect of the positional relationship between the text and the target. This may be the reason why its effect is lower than AH-LSTM and our method.

2. The performance of the AH-LSTM placed second in terms of performance. Although it uses position information to reduce the influence between different text passages describing different targets, this method has some problems. This method attempts to segment the text describing different targets into multiple sub-texts describing one target by the positions of the words. However, the inherent logic of natural language is flexible, and the method cannot flexibly segment text describing different targets, which leads to a poor stance detection effect. In addition, segmenting one text passage into four parts would cause the structure of the text segmentation to be destroyed. This may be the main reason for its poor performance.

In the GRU-PWV-CNN model, the position

information is fused into the model in the form of a vector, which ensures the integrity of the text while providing the model with position information. This may have been the reason for the superior performance of our method.

3. Our method achieved the best results, with a 2.82% increase in the Macro F_{avg}, which proved its advantages over the other methods. There are three main components of this method as follows: 1. the Bi-GRU layer for mining text features, 2. the position-weight fusion layer for mining position information and fusing it into the model, and 3. the CNN layer for secondary extraction of information from the fused information. The impact of the three parts will be discussed separately below.

4.2.2 Comparing Effects of Bi-LSTM and GRU

The unstructured features are extracted by the Bi-GRU, as described in Section 3.2. However, the LSTM can also achieve the same effect. Thus, two alternative networks were designed, and their effects were compared using the same encoding method, model structure, and three datasets. The model with the Bi-GRU is denoted as GRU-PWV-CNN, and the model with the Bi-LSTM is denoted as LSTM-PWV-CNN. The experimental results on the development set and test set are shown in Figure 6, where the y-axis refers to the Avg.

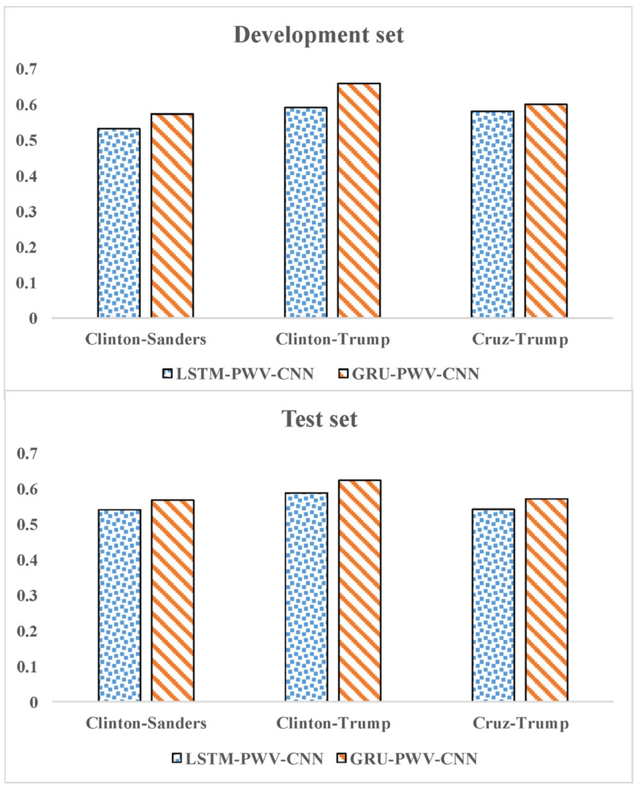


Figure 6. Avg of GRU-PWV-CNN and LSTM-PWV-CNN

By comparing the Avg of the two methods for the three datasets, the feature extraction performance of the GRU was better than that of the LSTM. In addition, because the GRU simplified the number of gates while ensuring the implementation of the LSTM function, the GRU-based model also had a slightly smaller time complexity than the LSTM. Thus, the GRU was used to complete the feature extraction of unstructured text.

4.2.3 Comparison of Different Coefficients in Position-weight Fusion Layer

In this paper, the position-weight vector is used to map the positional relationship between each word and the target into a vector. This vector is then embedded into the model to reduce the impact between different targets. To compare the effect of this vector on the model, the coefficient μ is used to scale each element in the vector proportionally. In addition, since the memory of the GRU fused with this vector passes through the tanh function, the memory tensor fused with this vector is between -1 and 1. Therefore, expanding the position-weight vector can expand the impact on the entire system. Although the optimal coefficient μ can be achieved through training, we still want to show the influences of the different fixed coefficients on the prediction results. So the effect of models with different μ values on the development set and test set, as shown in Figure 7.

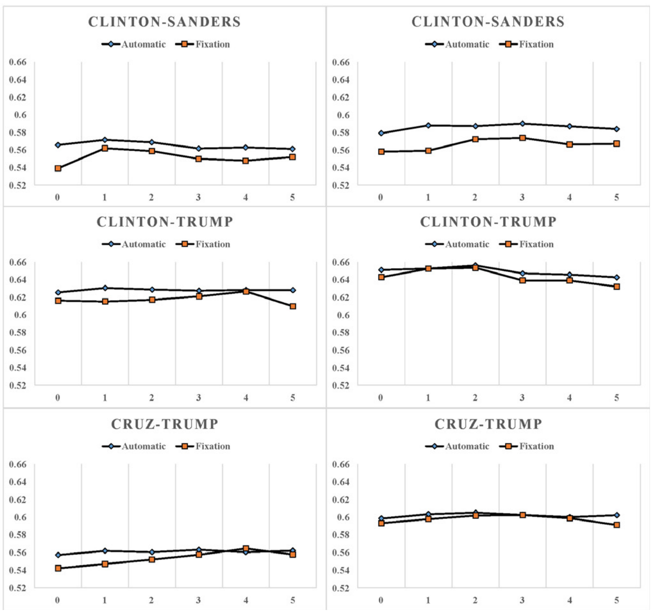


Figure 7. Avg for different fixed μ and automatically μ in the proposed method

In this picture, the x-axis denotes the μ size, and the y-axis refers to the Avg. Where automatic at the line with dots is the Avg obtained by the algorithm of using the μ as a trainable variable and initialized by 0, 1, 2, 3, 4, and 5. Fixation at the line with squares is the average of F_{avg} obtained by the algorithm when μ is 0, 1, 2, 3, 4, and 5.

The description in Figure 7 shows three important points: (1) it is found that the results of the model have obvious changes with the adjustment of the coefficient μ, and usually the performance when $\mu \neq 0$ is better than the effect when $\mu = 0$. This proves that PWV can help the algorithm to obtain better results in the stance detection task. (2) In most cases, the effect of automatically adjusting the coefficient μ is better than the fixed coefficient μ, because the coefficient obtained by training is easy to reach to the global optimal value compared to the one by adjusting manually. Therefore, this proves that the automatic acquisition of the coefficient μ not only omits the process of adjusting the hyper-parameters, but also has a better effect. (3) The effect of the algorithm in the development set is better than the one in the test set. Because the development set is used to adjust the parameters of our model to get the best effect, but the test set is not.

4.2.4 Comparing Effects of Different Feature Re-extraction Methods

After completing the final fusion of the position-weight vector with the extraction results from the unstructured text, the re-extraction should be processed, as mentioned in Section III.D. To find a suitable model to re-extract the fused features, three models were compared—a CNN, LSTM, and GRU. The results on the development set and test set are shown in Figure 8.

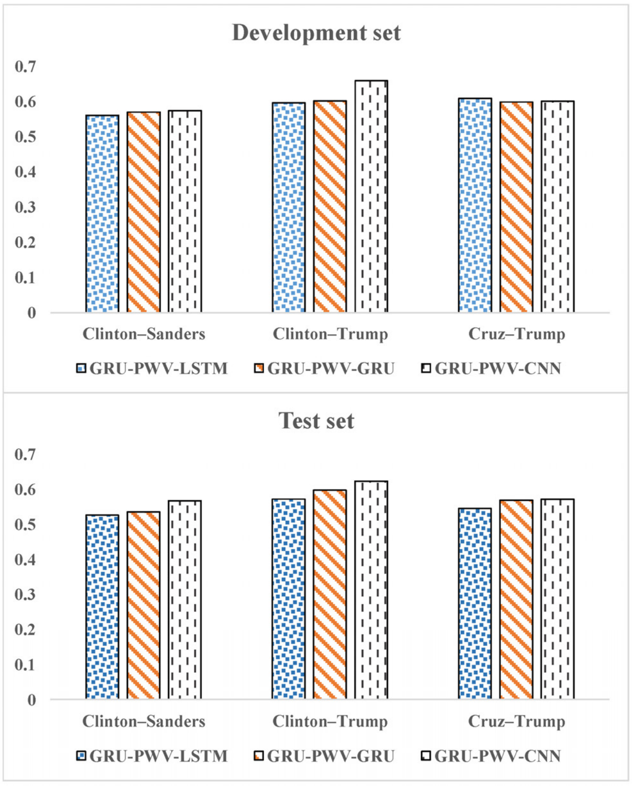


Figure 8. Average of F_{avg} of GRU-PWV-GRU, GRU-PWV-LSTM and GRU-PWV-CNN. The y-axis refers to the average of F_{avg}

From this picture, it is found that the overall performance of the CNN model is always better than the GRU and LSTM models in the development set and test set. Therefore, it is used to complete the re-extraction process. This may also prove that the fusion of RNN and CNN models can achieve better results. Because different network structures have their own advantages, the combination of them can solve the more complex problems.

5 Conclusion

People publish text containing various attitudes about other people, events, and other goals on social media. Mining this information has become an important channel for decision makers to obtain feedback of the public opinion. The purpose of stance detection technology is to automatically analyze the text in social media to obtain an attitude toward a specific target, which has played an important role in public opinion analysis. Because input text containing multiple targets has a complex structure and large amount of information, the multi-target stance detection task is separately defined as a sub-task of stance detection.

In this study, a GRU-PWV-CNN network for multi-target stance detection is proposed. First, a Bi-GRU is used to extract the original features from the unstructured input text. The corresponding position of the target in the input text is then calculated using the final position-weight vector. After this, the above two features are merged into the position weighted fusion layer. Subsequently, a CNN is used to re-extract features from the merged information. Finally, the softmax classification is used to determine the stances of different targets. The stance detection corpus for the US 2016 general election was used to validate our method. The experimental results showed that this method achieved and a 2.82% improvement in F-Macro.

To use deep learning to extract data features, adding multi-target position information to expand the tolerance of the input differences is a good method for improving the effectiveness of multi-target stance detection. In the future, more non-English multi-target stance detection data will be used, such as the Chinese corpus for the Taiwan 2020 election, for further experiments. This will expand the research scope on stance detection and allow stance detection technology to serve the Chinese corpus.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grants 61972040, and the Premium Funding Project for Academic Human Resources Development in Beijing

Union University under Grant BPHR2020AZ03. In addition, we thank Parinaz Sobhani, Diana Inkpen, and Xiaodan Zhu for providing the data used for our experiments, and Ai studio for providing us with a computing platform.

References

[1] P. Sobhani, *Stance Detection and Analysis in Social Media*, Ph. D. Thesis, University of Ottawa, Ottawa, Canada, 2017.

[2] S. M. Mohammad, S. Kiritchenko, P. Sobhani, X. D. Zhu, C. Cherry, Semeval-2016 Task 6: Detecting Stance in Tweets, *The 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, CA, USA, 2016, pp. 31-41.

[3] R. Xu, Y. Zhou, D. Wu, L. Gui, J. Du, Y. Xue, Overview of NLPCC Shared Task 4: Stance Detection in Chinese Microblogs, *The 24th International Conference on Computer Processing of Oriental Languages*, Kunming, China, 2016, pp. 907-916.

[4] S. M. Mohammad, P. Sobhani, S. Kiritchenko, Stance and Sentiment in Tweets, *ACM Transactions on Internet Technology*, Vol. 17, No. 3, pp. 1-23, July, 2017.

[5] R. Lehmann, Stance Detection in Danish Politics, Ph. D. Thesis, IT University of Copenhagen, Copenhagen, Denmark, 2019.

[6] M. Umer, Z. Imtiaz, S. Ullah, A. Mehmood, G. S. Choi, B. W. On, Fake News Stance Detection Using Deep Learning Architecture (CNN-LSTM), *IEEE Access*, Vol. 8, pp. 156695-156706, August, 2020.

[7] P. Sobhani, D. Inkpen, X. D. Zhu, A Dataset for Multi-Target Stance Detection, *The 15th Conference of the European Chapter of the Association for Computational Linguistics*, Valencia, Spain, 2017, pp. 551-557.

[8] J. J. Lin, Q. C. Kong, W. J. Mao, L. Wang, A Topic Enhanced Approach to Detecting Multiple Standpoints in Web Texts, *Information Sciences*, Vol. 501, pp. 483-494, October, 2019.

[9] C. Liu, W. Li, B. Demarest, Y. Chen, S. Couture, D. Dakota, N. Haduong, N. Kaufman, A. Lamont, M. Pancholi, K. Steimel, S. Kübler, IUCL at SemEval-2016 Task 6: An Ensemble Model for Stance Detection in Twitter, *The 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, CA, USA, 2016, pp. 394-400.

[10] Y. Igarashi, H. Komatsu, S. Kobayashi, N. Okazaki, K. Inui, Tohoku at SemEval-2016 Task 6: Feature-based Model versus Convolutional Neural Network for Stance Detection, *The 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, CA, USA, 2016, pp. 401-407.

[11] I. Augenstein, T. Rocktäschel, A. Vlachos, K. Bontcheva, Stance Detection with Bidirectional Conditional Encoding, *The 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, TX, USA, 2016, pp. 876-885.

[12] K. Dey, R. Shrivastava, S. Kaushik, Topical Stance Detection for Twitter: A Two-Phase LSTM Model Using Attention, *The 40th European Conference on Information Retrieval Research*, Grenoble, France, 2018, pp. 529-536.

[13] N. Yu, D. Pan, M. S. Zhang, G. H. Fu, Stance Detection in Chinese MicroBlogs with Neural Networks, *The 24th International Conference on Computer Processing of Oriental Languages*, Kunming, China, 2016, pp. 893-900.

[14] Q. Y. Sun, Z. Q. Wang, Q. M. Zhu, G. D. Zhou, Stance Detection with Hierarchical Attention Network, *The 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA, 2018, pp. 2399-2409.

[15] W. F. Li, Y. L. Xu, G. M. Wang, Stance Detection of Microblog Text Based on Two-Channel CNN-GRU Fusion Network, *IEEE Access*, Vol. 7, pp. 145944-145952, September, 2019.

[16] S. Chopra, S. Jain, J. M. Sholar, *Towards Automatic Identification of Fake News: Headline-Article Stance Detection with LSTM Attention Models*, The Course of Stanford CS224d: Deep Learning for Natural Language Processing, December, 2017.

[17] Q. Y. Sun, Z. Q. Wang, S. S. Li, Q. M. Zhu, G. D. Zhou, Stance Detection Via Sentiment Information and Neural Network Model, *Frontiers of Computer Science*, Vol. 13, No. 1, pp. 127-138, February, 2019.

[18] J. Ebrahimi, D. Dou, D. Lowd, A Joint Sentiment-Target-Stance Model for Stance Classification in Tweets, *The 26th International Conference on Computational Linguistics*, Osaka, Japan, 2016, pp. 2656-2665.

[19] G. Zarrella, A. Marsh, MITRE at SemEval-2016 Task 6: Transfer Learning for Stance Detection, *The 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, CA, USA, 2016, pp. 458-463.

[20] L. J. Sun, X. T. Li, B. W. Zhang, Y. M. Ye, B. X. Xu, Learning Stance Classification with Recurrent Neural Capsule Network, *The 8th CCF International Conference on Natural Language Processing and Chinese Computing*, Dunhuang, China, 2019, pp. 277-289.

[21] P. Sobhani, D. Inkpen, X. D. Zhu, Exploring Deep Neural Networks for Multitarget Stance Detection, *Computational Intelligence*, Vol. 35, No. 1, pp. 82-97, February, 2019.

[22] P. H. Wei, J. J. Lin, W. J. Mao, Multi-Target Stance Detection via a Dynamic Memory-Augmented Network, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, Ann Arbor, MI, USA, 2018, pp. 1229-1232.

[23] U. A. Siddiqua, A. N. Chy, M. Aono, Tweet Stance Detection Using Multi-Kernel Convolution and Attentive LSTM Variants, *IEICE Transactions on Information and Systems*, Vol. E102.D, No. 12, pp. 2493-2503, December, 2019.

[24] H. Liu, S. S. Li, G. D. Zhou, Two-Target Stance Detection with Target-Related Zone Modeling, *The 24th China Conference on Information Retrieval*, Guilin, China, 2018, pp. 170-182.

[25] X. J. Zhou, X. J. Wan, J. G. Xiao, Attention-based LSTM Network for Cross-Lingual Sentiment Classification, *The 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, TX, USA, 2016, pp. 247-256.

[26] A. Graves, J. Schmidhuber, Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures, *Neural Networks*, Vol. 18, No. 5-6,

pp. 602-610, July–August, 2005.

[27] W. P. Yin, K. Kann, M. Yu, H. Schütze, *Comparative Study of CNN and RNN for Natural Language Processing*, arXiv:1702.01923, February, 2017.

[28] Y. Jiang, *Using Machine Learning for Stance Detection*, Master's Thesis, The University of Texas at Austin, Austin, USA, 2019.

[29] M. Patel, A. Patel, R. Ghosh, *Precipitation Nowcasting: Leveraging bidirectional LSTM and 1D CNN*, arXiv:1810.10485, October, 2018.

[30] P. Chen, Z. Q. Sun, L. D. Bing, W. Yang, Recurrent Attention Network on Memory for Aspect Sentiment Analysis, *The 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark, 2017, pp. 452-461.

[31] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching Word Vectors with Subword Information, *Transactions of the Association for Computational Linguistics*, Vol. 5, pp. 135-146, June, 2017.

[32] X. Glorot, Y. Bengio, Understanding the Difficulty of Training Deep Feedforward Neural Networks, *Journal of Machine Learning Research*, Vol. 9, pp. 249-256, January, 2010.

Biographies

Wenfa Li was born in the Henan Province, China, in 1974. He received a Ph.D. degree in computer software and theory from the Institute of Computing Technology, Chinese Academy of Sciences, China, in 2009. He is currently a Professor in the Software Engineering Department, College of Robotics, Beijing Union University, China. His research interests include big data, information security, and machine learning.

Yilong Xu was born in Beijing, China, in 1993. He received a B.S. degree in computer science and technology from Beijing Union University. He is currently a master's student at Beijing Union University. His research interests include natural language processing and machine learning.

Gongming Wang was born in Henan Province, China, in 1981. He received a Ph.D. degree in computer system architecture from the Chinese Academy of Sciences and worked in a postdoctoral position at Tsinghua University. He is currently a researcher at Beijing Tianyuan Network Co., Ltd. (the subsidiary corporation of the Inspur Software Group Co., Ltd.). His research interests include big data and machine learning.