Handling negative mentions on social media channels using deep learning*

Khuong Vo, Tri Nguyen, Dang Pham, Mao Nguyen, Minh Truong, Dinh Nguyen and Tho Quan

YouNet Group, Ho Chi Minh City, Vietnam; Ho Chi Minh City University of Technology, Vietnam National University – HCMC, Ho Chi Minh City, Vietnam

ABSTRACT

Social media channels such as social networks, forum or online blogs have been emerging as major sources from which brands can gather user opinions about their products, especially the negative mentions. This kind of task, popular known as sentiment analysis, has been addressed recently by many deep learning approaches. However, negative mentions on social media have their own language characteristics which require certain adaptation and improvement from existing works for better performance. In this paper, we propose a new architecture for handling negative mentions on social media channels. As compared to the architecture published in our previous work, we expose substantial change in the combination manner of deep neural network layers for better training and classification performance on social-oriented messages. We also propose the way to re-train the pre-trained embedded words for better reflect sentiment terms, introducing the resultant sentimentally-embedded word vectors. Finally, we introduce the concept of a penalty matrix which incurs more reasonable loss function when handling negative mentions. Our experiments on real datasets demonstrated significant improvement.

ARTICLE HISTORY

Received 7 March 2018
Accepted 4 January 2019

KEYWORDS
Sentiment analysis; crisis management; deep learning; word embedding; convolutional neural network; recurrent neural network; loss function

1. Introduction

With the revolution of information and communication technologies through the Internet, social media has emerged as an efficient channel to handle the social crisis in real time manner (Middleton, Middleton, & Modafferi, 2014; Osborne et al., 2014). Especially, social media offers a promising way for emergency awareness and response. To the social media channels like the social networks of Facebook, Twitter or electronic newspapers, the information is updated continuously from the user as streamline of feeds.

There are more and more brands relying on such information to detect brand crisis, i.e. when a brand is being suffered from an unexpectedly high frequency of negative
comments on online channels. Toyota\(^1\) and Domino Pizza\(^2\) are the typical case studies where online channels offer effective means allowing negative information to spread quickly as a viral. However, this environment also allows the brand to counterattack effectively with a crisis via the various techniques: (i) early detection or prediction of crisis (Zhao, Resnick, & Mei, 2015) and (ii) opinion detection of users on dynamic environment of social media (Maynard, Gossen, Funk, & Fisichella, 2014).

In this paper, we focus on the latter, which is considered as a case of the well-known task of sentiment analysis. Recently, with the advances of deep learning technique (Young, Hazarika, Poria, & Cambria, 2017), there are many architectures proposed to handle this issue, notably the Convolutional Neural Network (CNN) (Kalchbrenner, Grefenstette, & Blunsom, 2014; Kim, 2014; Zhang & Wallace, 2015) and variants of the Recurrent Neural Network (RNN) (Cho et al., 2014; Hochreiter & Schmidhuber, 1997; Jozefowicz, Zaremba, & Sutskever, 2015), e.g. Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997). Whereas CNN is useful to extract hidden features from text data, LSTM is suitable to handle long sequence data. In our previous work (Nguyen et al., 2017), we proposed an architecture which combines CNN and LSTM to handle sentiment analysis of news articles on social media data. However, when applied with a huge amount of real mentions collected from social media channels, our architecture suffers from the following issues.

- The LSTM part in our architecture requires long training time, especially when the size of the dataset is huge. The advantage of LSTM is the capability of memorizing information in long documents. However, in the environments of social media, people do not tend to post relatively long messages. It motivates us to consider other lightweight versions of RNN, which are still keeping flowed information on sequence data, but requiring a less computational cost.
- The traditional approach of the word embedding method, which is usually used for text processing with deep learning, did not well reflect the difference between positive and negative terms. This is because the traditional training method of word embedding is based on the statistical information of the co-occurrence between words in documents. For example, let us consider a case that users express opinions about a certain product, e.g. smartphone. Users can give comments like ‘I think that Mobile A is a good smartphone’ (positive comment) or ‘IMHO, Mobile B is a clear example of a bad smartphone’ (negative comment). Thus, theoretically, the co-occurrence of (‘good’, ‘smartphone’) is quite equal to that of (‘bad’, ‘smartphone’). This causes the word-embedded vectors of two terms ‘good’ and ‘bad’ to be quite similar, which would heavily affect the accuracy of the subsequent sentiment classification task.
- In most deep neural network systems, a loss function is used to evaluate the error of the learning process. However, the typical loss function employed in the standard deep learning models does not reflect the seriousness of various cases of misclassification. Currently, we observe that most approaches using deep learning for sentiment analysis adopt loss functions that assign the same error rate for different error cases. For example, if a training sample is expected as a negative case, the loss function will produce the same error value if the classifier produces the result as positive or neutral case. Whereas, misclassification from negative to positive should be intuitively treated more serious than from negative to neutral.
Thus, in this paper, we introduce a new deep architecture for sentiment analysis, specifically aim at handling negative mentions in social media. Compared to the previous work, our new architecture presents the following novel features, which are also the contributions of this paper.

- In our model, the LSTM part in the previous work is replaced by the Gated Recurrent Unit (GRU) (Cho et al., 2014) to enjoy better performance. In comparison between two upgraded versions of RNN, LSTM has been proven more effective on long documents (Chung, Gulcehre, Cho, & Bengio, 2014). However, as our data of social-oriented messages are not often too long as previously discussed, the possible accuracy sacrifice endured by using GRU instead may be worthy, considering the computational improvement enjoyed.

- The process of training word embedding is integrated into the whole architecture, not separated as the previous work. Thus, the result of sentiment classification process can be used to re-train the word-embedding vectors as well. By doing so, the word-embedding vectors can better reflect the difference between sentiment terms resulted from the sentiment classification process. As a result, we introduce a novel model of sentimentally-embedded word vectors, in which the words are encoded by not only the statistical co-occurrence information but also their sentiment semantics.

- We also apply a penalty matrix with the cross entropy function to be used as the loss function of the combined deep architecture. Enhanced by the penalty matrix, the new loss function yields more reasonable loss value when encountering misclassification from negative cases.

Those contributions have been confirmed by our experimental results on two real data-sets, especially when handling negative cases. The rest of the paper is organized as follows. In Section 2, we recall some background knowledge required for this study. Section 3 captures the related works. Our proposed deep architecture is presented in Section 4, with a comparison to our own previous work. Section 5 presents our experiment, and Section 6 finally concludes the paper.

2. Preliminaries

2.1. Word embedding

The word embedding basic model is often used to perform weighting vector. Basically, this is a weight vector, for example, 1-of-N (one-hot vector), used to encode a word in the dictionary of $M$ words into a vector of length $M$. As presented in Figure 1 are one-hot vectors representing two words in a dictionary.

With such a simple representation, we cannot evaluate the similarity between words since the distance between any two vectors are always the same (e.g. if we use cosine distance). Moreover, the dimensionality of the vector space is huge when applied with the real dictionary.

The word embedding model using Word2vec technique (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) represents the form of a distribution relationship of a word in a
dictionary with the rest (known as Distributed Representation). In this model, words are embedded in a continuous vector space where semantically similar words are represented in adjacent points. This is based on the idea that words sharing semantic meaning are in the same contexts. As presented in Figure 2, each word now is represented as a \( K \)-dimensional vector, where \( K << M \), and each element of the vector is represented by new learning value.

Word2vec comes in two techniques, the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model. These are shallow three-layer neural networks which map words to the target words for learning weights which are word vector representations. CBOW model predicts a target word from the given context words, while the Skip-gram model predicts the context words from a given word.

### 2.2. Convolutional neural networks

The Convolutional Neural Network (CNN) is one of the most popular deep learning models. Given in Figure 3 is the general architecture of CNN system. The first layer builds the vectors from the words in the sentence. Input documents are transformed
into a matrix, each row of which corresponds a word in a sentence. For example, if we have a sentence with 10 words, each word was represented as a *word-embedding* vector of 100 dimensions, the matrix will have a size of $10 \times 100$. This is similar to an image with $10 \times 100$ pixels. The next layer performs *convolution* on these vectors with different filter sets and then *max-pooling* is performed for the set of filtered features to retain the most important features. Then, these features are passed to a fully connected layer with softmax function to produce the final probability output. *Dropout* (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) technique is used to prevent overfitting.

In Kim (2014), the basic steps of using a CNN in sentiment analysis were detailed in the process by which one feature is extracted from one filter as follows.

Given a sentence with $n$ words, let $x_i \in \mathbb{R}^k$ be a $k$-dimensional word vector corresponding to the $i$th word in the sentence. The sentence can be represented as:

$$x_{i:n} = x_1 + x_2 + \cdots + x_n.$$  

Here, ‘+’ denotes vector concatenation. Generally, $x_{i:i+j}$ represents the vector from index $i$ to $i+j$. A convolution operator with filter $w \in \mathbb{R}^{h \times k}$ for $h$ words produces the feature:

$$c_j = f(W \cdot x_{i:i+h-1} + b).$$

Here, $b$ is the bias and $f$ is a nonlinear function. By applying the filter on all windows of the sentence, we obtain the feature map:

$$c = [c_1, c_2, c_3, \ldots, c_{n-h+1}].$$

as the feature corresponding to this filter. Alternately, the max-pooling can be applied through local parts of the feature map to get local maximum values for this filter.

### 2.3. Recurrent neural networks

#### 2.3.1. Long short-term memory

The *Long Short-Term Memory* (LSTM) model introduced by Hochreiter & Schmidhuber. This model is a modification of the *Recurrent Neural Network* (RNN) model. As depicted in
Figure 4, an RNN is arranged in a linear pattern, called state, corresponding to a data entry of input data. For example, to handle a text document, a state corresponds to a word in the text at timestep $t$. Each state received input including the corresponding data entry $x_t$ and the previous state $s_{t-1}$ to output the new state $h_t$. RNN shares the same parameters ($U, W$) across all steps.

In the 1990s, the RNN faced two major problems of Vanishing and Exploding Gradients. During the gradient back-propagation, the gradient signal can be multiplied many times as the number of timesteps by the weight matrix. If the weight is too small, the information learned will almost be eliminated when the number of states becomes too large (which is our case of processing long mentions). Conversely, if the weight is large, this will lead to the case known as the Exploding Gradients when the gradient signal increasingly distracted during training, causing the process not converged.

LSTM has a similar structure to RNN. However, it uses a more complex way to compute the hidden state, which introduces a new structure called a memory cell as shown in Figure 5. LSTM prevents the above problems through a gating mechanism in memory cells. It uses gates which are the values between 0 to 1 to control how much of the information to let through at each timestep. Moreover, at each timestep, there is another input $c_{t-1}$ and another output $c_t$ which are the previous internal memory and the exposed internal memory, respectively. The equations below describe

Figure 5. Illustration of an LSTM memory cell.
how an LSTM memory cell is updated at every timestep $t$:

\[
\begin{align*}
    i_t &= \sigma(W_i x_t + U_i h_{t-1}) \\
    f_t &= \sigma(W_f x_t + U_f h_{t-1}) \\
    o_t &= \sigma(W_o x_t + U_o h_{t-1}) \\
    g_t &= \tanh(W_g x_t + U_g h_{t-1}) \\
    c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
    h_t &= o_t \odot \tanh(c_t),
\end{align*}
\]

where $\sigma$ denotes the logistic sigmoid function, $\odot$ is the elementwise multiplication, $i$ is the input gate, $f$ is the forget gate, $o$ is the output gate, $g$ is the candidate hidden state.

2.3.2. Gated recurrent unit

The Gated Recurrent Unit (GRU) was proposed by Cho et al. (2014), which is a variation of the LSTM. The GRU also has gating mechanisms that adjust the information flow inside the unit. However, it does not have a separate memory cell and combines the forget and input gates into a single update gate. These changes make GRU a simpler model than LSTM.

A GRU memory cell, as illustrated in Figure 6, is updated at every timestep $t$ as the following equations:

\[
\begin{align*}
    r_t &= \sigma(W_r x_t + U_r h_{t-1}) \\
    z_t &= \sigma(W_z x_t + U_z h_{t-1}) \\
    \hat{h}_t &= \tanh(W_x x_t + U (r_t \odot h_{t-1})) \\
    h_t &= (1 - z_t) h_{t-1} + z_t \hat{h}_t,
\end{align*}
\]

where $\sigma$ denotes the logistic sigmoid function, $\odot$ is the elementwise multiplication, $r$ is the reset gate, $z$ is update gate and $\hat{h}_t$ denotes the candidate hidden layer.

3. Related works

3.1. Sentiment analysis

Sentiment analysis (Nasukawa & Yi, 2003) or opinion mining (Dave, Lawrence, & Pennock, 2003) is the task that aims to infer the sentiment orientation in a document (Padmaja &
There are three levels of sentiment: (i) document-based level; (ii) sentence-based level; and (iii) aspect-based level. In document-based and sentence-based sentiment analysis, it is implicitly assumed that the analysed document or sentence only discusses a single object. Recently, Hu and Liu (2004) proposed to perform sentiment analysis at aspect-level (Liu, 2012). In this direction, apart from rating positive/negative sense of a text, the objects targeted by the mention (it may be a brand, a product or a feature) must also be identified.

Various works have been reported to handle this problem. **Topic modelling** was commonly applied in this case (Qiu, Liu, Bu, & Chen, 2011). Mei, Ling, Wondra, Su, and Zhai (2007) proposed using **Probabilistic Latent Semantic Analysis** (PLSA), while most recent works were based on **Latent Dirichlet Allocation** (LDA) (Li, Huang, & Zhu, 2010; Mukherjee & Liu, 2012; Sauper, Haghighi, & Barzilay, 2011; Zhao, Jiang, Yan, & Li, 2010). Insight analysis (Pang & Lee, 2005) improves the performance by inferring features of the analysed aspect.

### 3.2. Deep learning for sentiment analysis

Recently, using deep learning for NLP task has become an emerging trend. Back to 1989, LeCun did propose an architecture of shared weights in neural networks (LeCun, 1989). Subsequently, suggested by Hinton (1990), the idea of feeding a neural network with inputs through multiple-way interactions, parameterized by a tensor have been proposed for relation classification (Jenatton et al., 2012).

Word embedding and CNN were soon adopted for sentiment analysis using deep learning. Yu, Wang, Lai, and Zhang (2017) proposed to refine pre-trained word embedding with a sentiment lexicon. The work of Kim (2014) can be considered as a ‘standard approach’ using CNN for sentiment analysis. Meanwhile, and Kalchbrenner et al. (2014) constructed a hierarchical model by interleaving the max pooling layers with the convolutional layers.

Recently, **Recursive Neural Tensor Network** (RNTN) has been applied on Stanford Sentiment Treebank (SST) and obtained remarkable performance (Socher et al., 2013). Beside the SST, a new dataset, known as CMUMOSI, is introduced for sentiment analysis in a multi-model setup (Zadeh, Zellers, Pincus, & Morency, 2016).

To capture the relationship made by the appearance order of features, RNN model such as LSTM model has been used in combination with convolution processing to sentiment analysis for short text (Wang, Jiang, & Luo, 2016). Another enhanced version of LSTM, known as tree-LSTM (Tai, Socher, & Manning, 2015) is introduced, which is claimed to have better performance.

### 3.3. Sentiment analysis approaches for Vietnamese

Sentiment analysis has also attracted much attention in the research community in Vietnam. Various methods have been proposed. Kieu and Pham (2010) suggested using rule-based approach. Meanwhile, Duyen, Bach, and Phuong (2014) proposed a model combining SVM and maximum entropy to perform the classification task. Development of a sentiment dictionary for a certain domain and using other lexicon-based approaches were suggested in Trinh, Nguyen, Vo, and Do (2016) and Nguyen, VanLe, Le, and Pham (2014).
Using deep learning is also a considerable approach. Word embedding combined with multi-layer neural networks to handle various layers of semantic was suggested in Pham, Le, and Le (2016). In Vo, Nguyen, Le, and Nguyen (2017), a model combining CNN and LSTM, which is similar to our previous work (Nguyen et al., 2017), was proposed.

4. The deep architecture for negative-oriented sentiment analysis

4.1. The previous architecture

Figure 7 presents an overview of our previous work (Nguyen et al., 2017) of using deep architecture for sentiment analysis on news articles. The system includes the following modules.

**Word Embedding Module.** Using Skip-gram model, this is a three-layer neural network which maps words to the target words, to learn weights which act as word vector representations. The first layer consists of M nodes where M is the number of words in the dictionary. Each word w is fed to the input layer as a one-hot vector. The hidden layer consists of K neurons, where K<<M. The last layer also has M nodes which is a softmax layer. The target words, which are one-hot vectors, are the words that appear around the input word in a context window. The errors between outputs and targets are propagated back to re-adjust the weights.

After the word embedding layer is trained, the weights $w_{ij}$ on the connections from the $i$th nodes of input layer to $j$th nodes of the hidden layer form a $W_{M \times K}$ matrix for further usage.

Figure 7. The previous overall deep architecture.
Mention Input Module. This is a set of collected documents, each of which was previously labelled as positive, neutral, negative over an object. Originally, a document of \( N \) words is represented as a matrix \( D_{N \times M} \), in which the \( i \)th row is the one-hot vector of the \( i \)th word of the document. When performing the matrix multiplication \( D \times W \), we get an embedded matrix \( E_{N \times K} \) of the document. The matrix \( E \) is used as the input for the next Convolutional Neural Network module.

Convolutional Neural Network Module. At this stage, the convolution is performed between the matrix \( E \) with a kernel as a \( F_{d \times K} \) matrix. The meaning of the matrix \( F \) is to extract an abstract feature based on a hidden analysis of a \( d \)-gram from the original text. There are \( f \) matrices of \( F_{d \times K} \) used to learn \( f \) abstract features. As the convolution of two matrices \( E \) and \( F \) results in a column matrix of \( N \times 1 \), we eventually obtain the convoluted matrix \( C_{N \times f} \) by combining \( f \) column matrices together.

In the next step, the matrix \( C \) is pooled by a pooling window of \( q \times f \). The significance of this process is to retain the most important \( d \)-gram feature from \( q \) consecutive \( d \)-gram. Finally, we obtained the matrix \( Q_{p \times f} \) where \( p = N/q \).

At this point, instead of implementing a fully connected layer from the matrix \( Q \) as the CNN-based traditional method, the matrix \( Q \) continues to be used as input for the next LSTM Network Module.

LSTM Network Module. This layer consists of \( p \) contiguous states. State \( i \) receives external input as the \( i \)th row on the matrix \( Q \). Due to the characteristics of the LSTM network, if after a number of states, the current information is not relevant to the sentiment purpose, the output of the current state would not be passed to the next state. That is, LSTM starts predicting again from the next state.

The last state \( s_t \) is taken through a final fully connected layer to output the final result of classification (positive, neutral, negative). Error from the final result is propagated back to the beginning layer for the training process.

The dropout factors. As discussed, in order to avoid the overfitting, we use the Dropout factors (Srivastava et al., 2014). The application of the Dropout can be used for all systems with backpropagation learning. As shown in Figure 7, we used three dropout factors \( p_1, p_2, p_3 \), respectively for the input layer of the CNN, the state transitions on LSTM and the final fully connected layer.

4.2. The new architecture for negative mention detection

The previous architecture has been evolved into a new architecture as presented in Figure 8. This time, our purpose is to quickly detect negative mentions, in order to help a brand early counter-attack if any crisis occurs. Thus, in the new architecture, we only focus on negative mention detection. In other words, our classification system only classifies a new input data into two cases: negative case and non-negative case.

The new features introduced in this architecture are as follows.
Integrated Word Embedded Module. As one can observe, in our new architecture, the word embedding layer now is fully integrated into the sentiment classifier system, not separated as the previous model. In other words, the $E_{N \times K}$ matrix learned from the Word Embedding Module is now represented as a fully connected network, whose input is the one-hot vector representing the original word. Thus, the backpropagation process is performed from the sentiment classification result back to the embedding input layer, making the value of the $E_{N \times K}$ matrix re-trained accordingly based on the classification result. It would make the embedded vectors for the sentiment terms more reasonable. As shown in Figure 9, two embedded vectors for the term ‘good’ and ‘bad’ are represented in the new model. In the old architecture, the cosine similarity between those embedded vectors is approximately 0.7. In our new proposed system, this similarity decreases to around 0.3, which better reflects the difference between ‘good’ and ‘bad’. Also, Table 1 compares the some top similar words in the old and new architecture. Needless to say, those terms are significant to infer user opinions in text documents.

When looking into details of our architecture, one can observe that the cost used to encode the embedded vectors from words is not only based on the co-occurrence frequencies of words in the same context as in the original skip-gram model. In addition, the cost is also accumulated by the error of the sentiment classification process. By doing so, the encoded vectors of words frequently occurring in the same contexts would still be similar. However, encoded vectors of sentiment terms would be discriminately distinguished based on the sentiment polarity they imply (e.g. positive, negative, or neutral). Hence, we refer our integrated vector embedding model as the sentiment-
oriented word2vec model and the resultant embedded vectors as the sentimentally-embedded vectors.

We also would like to make a notice here regarding the time complexity of the new architecture when the word embedding module is integrated into the whole system. Thus, for each epoch, when the whole system is trained through the back-propagation mechanism, the word embedding module is also undergoing a training epoch. Let $O(P)$ be the complexity of the old system, the complexity of the new system is obviously $O(P + \pi)$ where $\pi$ is the time complexity of the weight update operations in the word embedding module. $\pi$ would not affect much to the overall complexity. We also present in our experiment the time comparison the word embedding is/ is not integrated into the whole system.

**Using GRU instead of LSTM.** In our new architecture, we replace the LSTM module by GRU module. By doing so, our new architecture enjoys the advantages. Generally, both LSTM and GRU yield comparable performance (Jozefowicz et al., 2015); however, GRU is computationally more efficient than LSTM due to a less complex structure as follows.

![Figure 9. New vectors are representing relationships between words.](image)

| Table 1. Top similar words for vectors trained in the separated process (left) and in the integrated process (right). |
|------------------------------------------------------------------------------------------------------------------|
| | Separated | Integrated |
|---|---|---|
| bad | good | poor |
| | cheap | awful |
| | fine | horrific |
| slow | fast | lag |
| | lag | complex |
| | delay | hung |
| *good | great | ok |
| | ok | bad |
| | fine | nice |

Note: Translated from Vietnamese.
• GRU has two gates while LSTM has three gates.
• GRU exposes the complete hidden state unlike LSTM which holds an internal memory.
• GRU has only one nonlinear computation (the \( \tanh \) function) as compared to LSTM, which has two, when computing output.

Thus, by intuition, the GRU units have fewer parameters and thus may enjoy faster training time or need less lengthy data to generalize. However, with long-length data, LSTM with higher expressiveness may lead to better results.

In our processing contexts, the data are of the kind of social-oriented massages, which are coming in vast numbers but not often too long. Thus, the improved training time enjoyed by GRU may be significant meanwhile the accuracy could be still maintained. Subsequent experimental results also confirm the benefit of this enhancement.

Penalty matrix. In neural network systems, one of the common methods for evaluating the loss functions is cross entropy (Bishop, 1995). Generally, a mention sample is labelled with a 2-dimensional vector \( y \). Each dimension respectively represents a value in \([\text{negative}, \text{non-negative}]\). For example, if a mention is labelled as negative, the corresponding \( y \) vector of this mention is \((1,0)\). After the learning process, a vector of a probability distribution over labels of 2-dimensional \( \hat{y} \) is generated, corresponding to the learning outcome of the system. The loss function is then calculated by the cross entropy formula as follows:

\[
H(y, \hat{y}) = - \sum y_i \ln \hat{y}_i.
\]

However, unlike standard classification task, the importance of each label in sentiment analysis (negative, non-negative) is different. Generally, in this domain, the data are imbalanced. That is, the number of non-negative mentions is very large, as compared to negative labels. Therefore, if a mention is classified as non-negative, the probability that it is a misclassified case is lower than the case it is classified as negative. The loss function is calculated by the default cross entropy function does not reflect those issues. Thus, we introduce a custom loss function, known as weighted cross entropy in which the cross entropy loss is multiplied by a corresponding penalty weight specified in a penalty matrix as shown in Table 2.

According to the penalty matrix in Table 2, one can observe that if a mention is expected to be non-negative but is predicted as negative, the corresponding penalty weight is 3. Meanwhile, for the case that a mention is expected to be negative and predicted as non-negative, the penalty weight is 1.5. In other words, the former case is considered more serious than the latter. It is obvious that if the prediction and the expectation match to each other, the penalty weight value is 1 (i.e. the loss function is minimized in this case).

We tentatively justify the weights for our penalty matrix as follows. Supposed that in our dataset, the mentions belonging to the sets of non-negative and negative are respectively \( X \) and \( Y \) per cent (obviously, \( X+Y =1 \)).

| Penalties | Predicted/expected | Non-negative | Negative |
|-----------|--------------------|--------------|----------|
| Non-negative | 1                  | 1.5          |
| Negative     | 3                  | 1            |
We consider a situation that the trained model achieves the accuracy of \( p \) per cent. Subsequently, we focus on two cases of misclassification as follows:

- (Case 1) All of misclassified cases are that the system predicts non-negative for the expected negative cases.
- (Case 2) All of misclassified cases are that the system predicts negative for the expected non-negative cases.

As our focus in this paper is to handling negative mentions, we consider the negative-case precision in each case. In Case 1, the negative-case precision is 1 (no wrong detection of negative case), meanwhile in Case 2, the negative-case precision is \( \frac{Y}{Y + (1 - p)} \). Thus, the negative-case precision of Case 2 is reduced as compared to Case 1. However, as the traditional loss function reflects the current accuracy of the trained model, which are now the same \( (p) \) for two cases, the learning process does not learn from the error caused by the loss of negative-case precision.

Urged by this observation, we intuitively set the weights on the penalty matrix to reflect this issue. The misclassified cases are punished with slightly heavier weights, and the misclassification in Case 2 gets a double penalty weight as compared to the one in Case 1. More reasonable weights can be deducted based on the distribution of observed data. However, we leave this direction for future work.

Example 4.1 If \( y = [1, 0] \) (negative) \( \hat{y} = [0.3, 0.7] \) (non-negative), then the default cross entropy results in 1.204, while the result of weighted cross entropy is 1.806.

Example 4.2 If \( y = [0, 1] \) (non-negative) \( \hat{y} = [0.8, 0.2] \) (negative), the default cross entropy also results in 1.609, while the result of weighted cross entropy is 4.828.

Examples 4.1 and 4.2 show that the weighted cross entropy function gives different loss values to different misclassification cases. Currently, we develop our penalty matrix based on observable intuition. However, in the future, we can rely on the distribution of data to construct this penalty matrix.

5. Experiments

5.1. Experiment settings

We have implemented a learning system based on the proposed deep architecture and conducted experiments to perform negative detection on two Vietnamese datasets YNM and VLSP. Table 3 shows the summarized characteristics of the datasets.

YNM is a large sentiment dataset which is labelled and provided by YouNet Media, a company specializing in online data analysis. YNM dataset consists of documents collected from the social media channels of Facebook, YouTube, Instagram, forum, e-newspapers and blogs in Vietnam. The data are centralized by SocialHeat platform, as illustrated in Figure 10. By manual effort, we filter and collect mentions which expressed ideas about real products. In Figure 10 are some customer reviews about the new smartphone product S9 of Samsung. To label data, a friendly UI has been developed for the curators to work, as shown in Figure 11. In this dataset, a negative label is assigned to a
mention if the mention contains only negative opinions or contains both negative and positive opinions.

VLSP dataset was provided by VLSP 2016 sentiment analysis contest. The dataset consists of real data collected from social media. In this dataset, a negative label is assigned to a mention if the mention contains only negative opinions. Neutral and positive mentions are merged into non-negative mentions.

For training model on YNM dataset, since there are many noisy terms such as misspellings, usernames, and hashtags, etc., we only keep words whose total frequency is greater than 100. The original dimension of our one-hot vectors is over 50,000, reduced to 320 after performing word embedding using Skip-gram model with a window size of 7. Each mention in the training set is truncated or padded to be the same length of 150. We also add two more dimensions to the embedding matrix for padding words and the words that are out-of-vocabulary words as zero vectors. Since the word embedding matrix is continued to be trained in classification task, these added dimensions are

| Dataset | Class | Training | Test |
|---------|-------|----------|------|
| YNM     | Negative | 160,000  | 40,000 |
|         | Non-negative | 800,000  | 200,000 |
| VLSP    | Negative | 1700     | 350  |
|         | Non-negative | 3400     | 700  |

Table 3. Summary of YNM and VLSP dataset characteristics.

Figure 10. SociaHeat interface.
kept untrainable (no gradient update). In the Convolutional Neural Network layer, we depict three filter region sizes: 3, 5 and 7, each of which has 64 filters. In our system, a max-pool of size 10 slides through feature maps. In Recurrent Neural Network layer, both LSMT and GRU use 128-unit cells. All dropout factors are set at 0.5.

For training model on VLSP dataset, it is hard to train models from scratch on such small dataset. Hence, we use the models trained on the YNM dataset, which generalized well on the large dataset, to continue to train on the VLSP dataset. This technique is a kind of transfer learning which is frequently used in computer vision (Yosinski, Clune, Bengio, & Lipson, 2014).

5.2. Classification models

In order to verify the performance of our proposed architecture, as compared to other existing work, we experiment various classification models, as depicted in Table 4, where various approaches have been employed, corresponding three contributions of our proposal: the integrated word embedding, using the GRU and using the penalty matrix.

In Table 4, the names of the methods are encoded to reflect the corresponding approaches. For example, method $W - LP+$ indicates not using the integrated word embedding, using LSTM instead of GRU and the penalty matrix is used on this method. Similarly, $W + GP-$ indicates that the integrated word embedding is used on this method, together with the GRU model where the penalty matrix is not employed.

5.3. Performance comparison

Tables 5 and 6 present the experimental results measured by our methods, using the standard metrics of recall, precision and F-measure on the YNM and VLSP datasets, respectively.
As our focus is to detect negative mention, we measure those metrics over the samples labelled as negative mentions.

In order to observe the individual impact of each contribution on two datasets, we also compare the F-measure gained when using/not using the integrated word embedding (Figures 12(a) and 13(a)), precision gained when using/not using penalty matrix (Figures 12(b) and 13(b)), and F-measure gained using LSTM/GRU (Figures 12(c) and 13(c)). All experimental results show that all of our proposed contributions remarkably improve the performance. The performance achieved when experimenting on the VLSP dataset is lower than that on the YNM dataset, maybe due to the fact that small volume of the VLSP dataset may not be sufficient for the deep learning approach. However, our ultimate model \((W + GP+)\) still got the best F-measure on this dataset.

To have a closer view, one can observe that the enhancements of the integrated word embedding and penalty matrix did improve the accuracy performance on both datasets, as seen in Figures 12(a), 13(a), 12(b) and 13(b). The replacement of LSTM by GRU did not introduce much improvement of accuracy, as illustrated in Figures 12(c) and 13(c). However, Figure 12(d) and presents another advantage of GRU over LSTM when training with a large dataset. In this figure, a training loss curve is presented, showing that GRU converges far faster than LSTM when experimented in our YNM dataset. It is also our aim when enhancing LSTM by GRU to enjoy faster convergence rate. Meanwhile, Figure 13(d) shows the convergence rate of GRU and LSTM are almost the same, it may be due to the small size of the training dataset of VLSP.

On the other hand, despite the complex architecture, the model can handle over 15,000 mentions/second on the machine using Nvidia GeForce GTX 1070 GPU. This proved that our system can be fruitfully applied in production. In addition, to measure the time spent by the integrated word embedding process, we also record the average training time for each iteration with the batch size of 256, incurred when the integrated word embedding was/was not used. Table 7 presents the result. It is observable that the time spent on the integrated word embedding process did not much affect the training duration, as previously discussed.

**Table 5.** Detailed negative mention detection results on YNM dataset.

| Method | Precision | Recall | F-measure |
|--------|-----------|--------|-----------|
| \(W - LP\) | 0.795     | 0.838  | 0.816     |
| \(W - LP\) | 0.829     | 0.81   | 0.819     |
| \(W - GP\) | 0.799     | 0.828  | 0.813     |
| \(W - GP\) | 0.836     | 0.794  | 0.814     |
| \(W + LP\) | 0.833     | 0.865  | 0.849     |
| \(W + LP\) | 0.844     | 0.848  | 0.846     |
| \(W + GP\) | 0.839     | 0.872  | 0.855     |
| \(W + GP\) | 0.864     | 0.852  | 0.858     |

**Table 6.** Detailed negative mention detection results on VLSP dataset.

| Method | Precision | Recall | F-measure |
|--------|-----------|--------|-----------|
| \(W - LP\) | 0.675     | 0.709  | 0.692     |
| \(W - LP\) | 0.714     | 0.684  | 0.699     |
| \(W - GP\) | 0.684     | 0.716  | 0.700     |
| \(W - GP\) | 0.718     | 0.69   | 0.704     |
| \(W + LP\) | 0.705     | 0.715  | 0.710     |
| \(W + LP\) | 0.715     | 0.707  | 0.711     |
| \(W + GP\) | 0.706     | 0.712  | 0.709     |
| \(W + GP\) | 0.722     | 0.718  | 0.720     |
6. Conclusion

Social media has become a major information channel today. Especially, for celebrities and brands, a large number of negative comments may cause a social crisis, as many cases have been witnessed recently in Vietnam and worldwide.

To address this issue, in this paper we extend our previous work of sentiment analysis using deep learning to focus on handling negative mentions from social media channels. In order to do so, we do not merely change the number of labels when training the system, but also introducing a new substantially enhanced architecture. In light of the limitations of the previous work, various techniques have been employed to improve the performance on the new architecture, which includes the followings.

- We integrated the process of word embedding with the training process of sentiment analysis. Thus, the embedded words also reflect the sentiment semantics from the training data. Our sentiment-oriented word2vec model, which captures both context and sentiment information of the words in corpus, thus generates the so-called sentimentally-embedded vectors for Vietnamese. This results can be inherited for further research in this area.
- We introduce the mechanism of penalty matrix to improve the precision of negative case detection.

![Figure 12: Performance comparison of negative mention detection on YNM dataset.](image-url)
We also replace the part of LSTM network in the old architecture with GRU network, a new upgraded version of RNN, to enjoy faster training time. This approach is proven effective when working with a large dataset of YNM, but not showing much impact on the small dataset of VLSP. Hence, it confirms again that our new proposed architecture is particularly fit to social-oriented messages, not for all textual data in general.

As a result, when experimented with real datasets obtained from online social channels, we observe significant improvements gained from our newly proposed system. The experimental results confirm our predicted advantages of the enhancement, as GRU helps the new system to converge faster, while the integrated training process of word embedding and the penalty matrix reveals better F-1 performance. We also notice the difficulty when settling the weights for the penalty matrix. Currently, we set the weights based on our intuition, but we cannot

Figure 13. Performance comparison of negative mention detection on VLSP dataset. (a) Comparison based on using the integrated word embedding (b) Comparison based on using penalty matrix (c) Comparison between LSTM and GRU and (d) Comparison of convergence between LSTM and GRU.

Table 7. Execution time per iteration.

| Method | Time (seconds) |
|--------|----------------|
| LW −   | 0.0227         |
| GW −   | 0.0223         |
| LW +   | 0.0248         |
| GW +   | 0.0245         |

- We also replace the part of LSTM network in the old architecture with GRU network, a new upgraded version of RNN, to enjoy faster training time. This approach is proven effective when working with a large dataset of YNM, but not showing much impact on the small dataset of VLSP. Hence, it confirms again that our new proposed architecture is particularly fit to social-oriented messages, not for all textual data in general.
confirm whether the weights are optimized. In the future, we intend to base on the data distributions, which can be determined by sampling, to determine the weights systematically and mathematically. We also intend to consider using the emerging attention-based model (Kumar et al., 2016) to further improve the accuracy of the classification system.

Notes
1. http://www.bsgco.com/work/cases/toyota-reputation-management
2. http://cssc.uscannenberg.org/cases/v1/v1art6/
3. http://vlsp.org.vn/vlsp2016/eval/sa
4. http://www.younetmedia.com/socialheat-social-monitoring
5. http://vlsp.org.vn/resources-vlsp2016

Acknowledgments
We are grateful to YouNet Media and the VLSP organizers for supporting real datasets for our experiment.

Disclosure statement
No potential conflict of interest was reported by the authors.

Notes on contributors
Khuong Vo received his B.Eng. (Honors) degree in Computer Science from Ho Chi Minh City University of Technology in 2017. His current research focuses on applying machine learning and deep learning to solve real-world problems, especially in social media and e-commerce fields.

Tri Nguyen received his B.Eng. (Honors) degree in Computer Science from Ho Chi Minh City University of Technology in 2017. His current research interests include natural language processing, image processing and deep learning.

Dang Pham received his B.Sc degree in Computer Science from Ho Chi Minh University of Science in 2014. Currently, he is pursuing a Master degree in the same university. His research interests include image processing, natural language processing and neuroscience.

Mao Nguyen received his B.Sc degree in Computer Science from Ho Chi Minh University of Science in 2011 and holds a Master degree in Computer Science from John von Neumann Institute in 2015. His current research interests include data mining/machine learning, computer vision and statistical inference methods.

Minh Truong received his B.Eng. degree in Computer Science from Ho Chi Minh City University of Technology in 2012. He won VEF fellowship in 2014 to study at Iowa State University, USA, and completed his Master degree in 2016 under the supervision of Prof. Srikantha Tirgupura. His primary focuses now are on natural language processing and large-scale distributed system based on Hadoop.

Dinh Nguyen received his B.Eng. degree in Computer Science from Ho Chi Minh City University of Technology in 2014 and holds a Master degree in Computer Science from the same university in 2018. His main research interests include deep learning, data mining, automation and intelligent systems development.

Tho Quan is an Associate Professor in the Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Vietnam. He received his B.Eng. degree in Information Technology from HCMUT in 1998 and received Ph.D degree in 2006 from Nanyang Technological
University, Singapore. His current research interests include formal methods, program analysis/verification, the Semantic Web, machine learning/data mining and intelligent systems. Currently, he heads the Department of Software Engineering of the Faculty. He is also serving as the Chair of Computer Science Program (undergraduate level).

References
Bishop, C. M. (1995). Neural networks for pattern recognition. Oxford: Oxford University Press.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. Preprint, arXiv:1406.1078.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. Preprint, arXiv:1412.3555.

Dave, K., Lawrence, S., & Pennock, D. M. (2003). Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In Proceedings of the 12th international conference on World Wide Web (pp. 519–528). Budapest, Hungary.

Duyen, N. T., Bach, N. X., & Phuong, T. M. (2014). An empirical study on sentiment analysis for Vietnamese. In 2014 international conference on Advanced technologies for communications (ATC) (pp. 309–314) Hanoi, Vietnam.

Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. Artificial Intelligence, 46(1–2), 47–75.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735–1780.

Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 168–177).

Jenatton, R., Bordes, A., Roux, N. L., Obozinski, G. R., Bordes, A., & Obozinski, G. R. (2012). A latent factor model for highly multi-relational data. In Advances in neural information processing systems 25 (Vol. 2, pp. 3167–3175). Lake Tahoe, NV: ACM Press.

Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In International conference on machine learning (pp. 2342–2350). Lille, France.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. Preprint, arXiv:1404.2188.

Kieu, B. T., & Pham, S. B. (2010). Sentiment analysis for vietnamese. In 2010 Second international conference on knowledge and systems engineering (KSE) (pp. 152–157).

Kim, Y. (2014). Convolutional neural networks for sentence classification. Preprint, arXiv:1408.5882.

Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., … Socher, R. (2016). Ask me anything: Dynamic memory networks for natural language processing. In International conference on machine learning (pp. 1378–1387) New York, USA.

LeCun, Y. (1989). Generalization and network design strategies. Connectionism in perspective (pp. 143–155). Zurich, Switzerland.

Li, F., Huang, M., & Zhu, X. (2010). Sentiment analysis with global topics and local dependency. In AAAI (Vol. 10, pp. 1371–1376). Atlanta, Georgia.

Liu, B. (2012). Sentiment analysis and opinion mining. Synthesis Lectures on Human Language Technologies, 5(1), 1–167.

Maynard, D., Gossen, G., Funk, A., & Fisichella, M. (2014). Should i care about your opinion? Detection of opinion interestingness and dynamics in social media. Future Internet, 6(3), 457–481.

Mei, Q., Ling, X., Wondra, M., Su, H., & Zhai, C. (2007). Topic sentiment mixture: Modeling facets and opinions in weblogs. In Proceedings of the 16th international conference on World Wide Web (pp. 171–180). Alberta, Canada.

Middleton, S. E., Middleton, L., & Modafferi, S. (2014). Real-time crisis mapping of natural disasters using social media. IEEE Intelligent Systems, 29(2), 9–17.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, Volume 2 of NIPS’13, (pp. 3111–3119). USA: Curran Associates Inc.
Mukherjee, A., & Liu, B. (2012). Aspect extraction through semi-supervised modeling. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, pp. 339–348. Jeju Island, Korea.

Nasukawa, T., & Yi, J. (2003). Sentiment analysis: Capturing favorability using natural language processing. In Proceedings of the 2nd international conference on Knowledge capture (pp. 70–77). Sanibel Island, FL, USA.

Nguyen, H. N., Van Le, T., Le, H. S., & Pham, T. V. (2014). Domain specific sentiment dictionary for opinion mining of vietnamese text. In International Workshop on Multi-disciplinary Trends in Artificial Intelligence (pp. 136–148) Bangalore, India.

Nguyen, D., Vo, K., Pham, D., Nguyen, M., & Quan, T. (2017). A deep architecture for sentiment analysis of news articles. In International conference on computer science, applied mathematics and applications (pp. 129–140) Berlin, Germany.

Osborne, M., Moran, S., McCreadie, R., Von Lunen, A., Sykora, M., Cano, E., … O’Brien, A. (2014). Real-time detection, tracking, and monitoring of automatically discovered events in social media. In Proceedings of 52nd annual meeting of the association for computational linguistics: System demonstrations (pp. 37–42) Baltimore, USA.

Padmaja, S., & Fatima, S. S. (2013). Opinion mining and sentiment analysis-an assessment of peoples’ belief: A survey. International Journal of Ad hoc, Sensor & Ubiquitous Computing, 4(1), 21.

Pang, B., & Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Proceedings of the 43rd annual meeting on association for computational linguistics (pp. 115–124).

Pham, D.-H., Le, A.-C., & Le, T.-K.-C. (2016). Learning semantic representations for rating vietnamese comments. In 2016 Eighth international conference on knowledge and systems engineering (KSE) (pp. 193–198) Hanoi, Vietnam.

Qiu, G., Liu, B., Bu, J., & Chen, C. (2011). Opinion word expansion and target extraction through double propagation. Computational Linguistics, 37(1), 9–27.

Sauper, C., Haghighi, A., & Barzilay, R. (2011). Content models with attitude. In Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies (Vol. 1, pp. 350–358).

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 conference on empirical methods in natural language processing (pp. 1631–1642) : Seattle, USA.

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

Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. Preprint, arXiv:1503.00075.

Trinh, S., Nguyen, L., Vo, M., & Do, P. (2016). Lexicon-based sentiment analysis of facebook comments in Vietnamese language. In Recent developments in intelligent information and database systems (pp. 263–276). Springer.

Vo, Q.-H., Nguyen, H.-T., Le, B., & Nguyen, M.-L. (2017). Multi-channel lstm-cnn model for vietnamese sentiment analysis. In 2017 9th international conference on knowledge and systems engineering (KSE) (pp. 24–29).

Wang, X., Jiang, W., & Luo, Z. (2016). Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers (pp. 2428–2437) Osaka, Japan.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In Advances in neural information processing systems (pp. 3320–3328) : Montreal, Canada.

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2017). Recent trends in deep learning based natural language processing. Preprint, arXiv:1708.02709.

Yu, L.-C., Wang, J., Lai, K. R., & Zhang, X. (2017). Refining word embeddings for sentiment analysis. In Proceedings of the 2017 conference on empirical methods in natural language processing (pp. 534–539) Copenhagen, Denmark.
Zadeh, A., Zellers, R., Pincus, E., & Morency, L.-P. (2016). Multimodal sentiment intensity analysis in videos: Facial gestures and verbal messages. *IEEE Intelligent Systems, 31*(6), 82–88.

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

Zhao, W. X., Jiang, J., Yan, H., & Li, X. (2010). Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of the 2010 conference on empirical methods in natural language processing* (pp. 56–65) Cambridge, USA.

Zhao, Z., Resnick, P., & Mei, Q. (2015). Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th international conference on World Wide Web* (pp. 1395–1405) Florence, Italy.