other, we can say that the term is neutral, bearing little sentiment information.

There are five layers in Figure 1, and from top to bottom, they are: lookup, convolutional, linear, HardTanh and linear. Using the message in this figure as an example, the words of this post are the input of this feed-forward neural network. In this example message, we assume there is no phrase identified, so there are four input terms. If there is a phrase, let's say *all-time high* is detected as a phrase, then these two words will be treated as one input term. The top layer is the lookup table for term polarity values. Because the training input is message, which varies in length, we use a convolutional layer to extract features that can be fed to standard affine layers. There are different ways to generate the representation of text segments with different lengths. In this study, we use the concatenation convolutional layer, which concatenates the layers of max, min and average of the two polarity values of all terms in the input message. This layer gives the best performance, based on our pilot experiments. The concatenation layer is expressed as follow:

$$Z(m) = [Z_p(m), Z_n(m)] \quad (2)$$

$$Z_p(m) = [Z_{p,max}(m), Z_{p,min}(m), Z_{p,ave}(m)] \quad (3)$$

$$Z_n(m) = [Z_{n,max}(m), Z_{n,min}(m), Z_{n,ave}(m)] \quad (4)$$

Where $Z(m)$ is the representation of message m, $Z_p(m)$ is for the positivity values of all the terms in this message, and $Z_n(m)$ is for negativity values of the terms. Given the convolutional layer, we can get the output of the first linear layer:

$$f^1(t) = w_1 Z(m) + b_1 \quad (5)$$

The HardTanh layer:

$$\alpha = HardTanh(f^1(t)) \quad (6)$$

And the second linear layer, whose output, $f^2(t)$, is the sentiment score for input message m:

$$f^2(t) = w_2\alpha + b_2 \quad (7)$$

Where w_1, w_2, b_1, b_2 are the parameters of the linear layers. The non-linear HardTanh layer is to extract highly non-linear features. Without the HardTanh layer, the network would be a simple linear model. Because the hard version of the hyperbolic tangent is slightly cheaper to compute and still keep the generalization performance unchanged, it is chosen as the non-linear layer.

The *HardTanh(x)* function is defined as:

$$HardTanh(x) = \begin{cases} -1, \text{ if } x <= -1 \\ x, \text{ if } -1 <= x <= 1 \\ 1, \text{ if } x > 1 \end{cases} \quad (8)$$

Since we have just two labels for the output, negative and positive, the dimension of the second linear layer is 2. If the polarity of a StockTwits message is positive, the predicted positive score is expected to be larger than the predicted negative score, and vise versa.

The hinge loss of this model is defined as:

$$loss(m) = max(0, 1- g(m) f_p(m) + g(m) f_n(m)) \quad (9)$$

Where $g(m)$ is the gold value of message m (positive or negative), $f_p(m)$ is the predicted positive score, and $f_n(m)$ is the predicted negative score.

Model Training: The data set used for training this model is already described in previous section. To train this model, we take the derivative of the loss by back-propagation with respect to the whole set of parameters, and use AdaGrad to update the parameters (Collobert et al., 2011; Duchi et al., 2011). Each term is first initialized by randomly choosing a negative and positive value less than 0.2. The same neural network and parameters setting are used to learn the sentiment polarity for both words and phrases. A validation data set was used to tune the model hyper-parameters.

Baseline Methods for Performance Comparison: We compare our method to three other methods: TF.IDF, PMI and Vo & Zhang from (Vo and Zhang, 2016), which is based on a simple neural network. PMI and TF.IDF are the two most successful approaches building lexicons based on statistic measures. The Vo & Zhang method is the state-of-the-art approach utilizing machine learning technology. We described them briefly below.

TF.IDF is usually used for calculating the weight of a term in text analysis tasks, and it has been used in previous studies for lexicon construction (Oliveira et al. 2014; Oliveira et al., 2016; Al-Twairesh et al., 2016). To use it for computing a term's sentiment score, we first created two documents composed by all the messages of each class (bullish document and bearish document). Then, for each term, we compute its TF.IDF value for the bullish and bearish classes, respectively. And finally we can compute