Financial Aspect and Sentiment Predictions with Deep Neural Networks: An Ensemble Approach

Guangyuan Piao
Insight Centre for Data Analytics
Data Science Institute
National University of Ireland, Galway
Galway, Ireland
guangyuan.piao@insight-centre.org

John G. Breslin
Insight Centre for Data Analytics
Data Science Institute
National University of Ireland, Galway
Galway, Ireland
john.breslin@nuigalway.ie

ABSTRACT

In this paper, we describe our ensemble approach for sentiment and aspect predictions in the financial domain for a given text. This ensemble approach uses Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) with a ridge regression and a voting strategy for sentiment and aspect predictions, and therefore, does not rely on any handcrafted feature. Based on 5-cross validation on the released training set, the results show that CNNs overall perform better than RNNs on both tasks, and the ensemble approach can boost the performance further by leveraging different types of deep learning approaches.

KEYWORDS

Sentiment Analysis; Financial Aspect Prediction; Deep Learning

1 INTRODUCTION

Deep learning [10] techniques such as Convolutional Neural Networks (CNNs) [11] for processing data in the form of multiple arrays, or Recurrent Neural Networks (RNNs) such as Long Short-Term Memory neural networks (LSTMs) [5] for tasks with sequential inputs, have been widely adopted in various research domains such as computer vision [4, 9], natural language processing (NLP) [6, 7] including sentiment analysis [3], recommender systems [14] etc.

More recently, [2] proposed an ensemble approach using several deep neural networks (DNNs) such as CNNs and LSTMs for one of the tasks at the sentiment analysis challenge SemEval2017[1], and their approach outperforms other methods for the sentiment analysis task on Twitter[2]. Deep learning approaches have also been applied to the sentiment analysis in the financial domain which plays a significant role in predicting the market reaction [3]. Motivated by the state-of-the-art results on different tasks including sentiment analysis, we introduce our ensemble approach with different types of DNNs for tackling the first task at the Financial Opinion Mining and Question Answering (FIQA) challenge[3], which is co-located with the Web Conference 2018.

[1] http://alt.qcri.org/semeval2017/index.php?id=tasks
[2] https://twitter.com
[3] https://sites.google.com/view/fiqa

WWW '18 Companion, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5640-4/18/04.
https://doi.org/10.1145/3184558.3191829

1.1 Task1: Sentiment and Aspect Predictions in the Financial Domain

Given a text t and a target as an input, the task is to predict its sentiment score y_s and its aspect labels y_a at the level two of an financial aspect tree. An example of a training instance is shown as below:

```
"55": {
    "sentence": "Tesco Abandons Video-Streaming
                 Ambitions in Blinkbox Sale",
    "info": [
      {
        "snippets": "['Video-Streaming Ambitions']",
        "target": "Blinkbox",
        "sentiment_score": "-0.195",
        "aspects": "['Corporate/Stategy']"
      },
      {
        "snippets": "['Tesco Abandons Video-Streaming
                     Ambitions ']",
        "target": "Tesco",
        "sentiment_score": "-0.335",
        "aspects": "['Corporate/Stategy']"
      }
    ]
  }
```

Table 1 shows the details of the dataset for task 1. There are two types of text; one is Twitter posts and the other is news headlines in the financial domain. Overall, there are 28 distinct aspects for all posts and headlines in the released training dataset.

Table 1: Dataset statistics for task 1.

	Posts	Headlines
# of examples	675	436
# of multi-labeled	0	30
max length	40	19
distinct aspects	28	

2 DEEP NEURAL NETWORKS

In order to feed into DNNs such as CNNs, we first convert each text as a sequence of words, and map each word to its corresponding embedding. Therefore, a text is represented as a matrix of size $m' \times d$, where m' is the number of words in the text and d is the dimension of the word embedding space. We used the zero-padding strategy in order to make all texts have the same length m. The final matrix of a given text $T \in \mathbb{R}^{m \times d}$ ($m = 50$ for our approach) is used as an input to one of the DNNs which we will describe below.

2.1 Convolutional Neural Networks

CNNs apply convolutional filters to the input matrix with a filtering matrix $f \in \mathbb{R}^{h \times d}$ where h is the filter size which denotes the number of words it spans. This operation can be defined as follows:

$$c_i = a(\sum_{j,k} f_{j,k}(T_{[i:i+h-1]})_{j,k} + b) \tag{1}$$

where $b \in \mathbb{R}$ denotes the bias term, and $a(\cdot)$ is a non-linear activation function. We used the well-known ReLu activation here. Different filter sizes such as [1, 2, 3] or [3, 4, 5] can be used for CNNs, and there can be multiple filters for each filter size.

Next, CNNs can apply a pooling operation to each convolution with the hope to extract the most important feature for each convolution. For example, the max-pooling $c_{max} = max(c)$ retains the maximum value for each convolution. Finally, the output from each convolution is concatenated into a single vector which can be seen as the text embedding for a given text learned by CNNs.

For CNNs, we used the following settings for their hyperparameters for training:

Table 2: Hyperparameter settings for CNNs.

Hyperparameter	Settings
pooling	max pooling
filter sizes	[1, 2, 3], [3, 4, 5] or [5, 6, 7]
# of filters	200 - 300

2.2 Long-Short Term Memory Networks

The first step in LSTM is going through a *forget gate layer*. This layer decides what information to keep from the cell state by looking at h_{t-1} and x_t, as shown in Equation 2:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2}$$

where $[h_{t-1}, x_t]$ denotes the concatenated vector of h_{t-1} and x_t, σ is is a sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$, and b_f denotes a bias term. W_f is a weight vector to be learned for the forget gate layer.

Next, LSTM decides what new information to store in the cell state, which consists of two parts. The first part is an *input gate layer*, which is defined as below:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i). \tag{3}$$

The second part is a tanh layer (Equation 4), which creates a vector of new candidate values, $\tilde{C}_t$, that could be added into the cell state.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4}$$

Finally, the new cell state C_t will be created based on linear interactions of the previous cell state C_{t-1}, f_t, i_t, and $\tilde{C}_t$ as follows.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{5}$$

The last step is filtering the cell state to generate the final output, which can be formulated as below:

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \\ h_t &= o_t * \tanh(C_t) \end{aligned} \tag{6}$$

Here, o_t decides what parts of the cell state to keep for the final output, and the cell state goes through a tanh layer before multiplying by o_t.

Another RNNs we used for Gated Recurrent Unit (GRU) [1] is a variant of LSTMs which has less parameters to tune.

For DNNs, we used the following hyperparameter settings for training:

Table 3: Hyperparameter settings for training DNNs.

Hyperparameter	Settings
embedding dimension	200
dropout rate	0.5
batch size	40
epoches	100 - 300

2.3 Regularization and Training

For regularization, we use the dropout [12] in the same way as [13]. Dropout, which refers to dropping out units in a neural network, is one of the widely used regularization techniques for preventing overfitting in training neural networks. Individual nodes are either "disabled" with probability $1 - p$ or kept with probability p. The "thinned" outputs of a hidden layer are then used as an input to the next layer. In this way, it prevents units from co-adapting and forces them to learn useful features individually.

We also constrain l_2-norms of the weight vectors to a threshold ϵ as below, which normalizes a word vector w so that its l_2-norm is equal to ϵ, and will be performed whenever the l_2-norm of w is bigger than ϵ ($\epsilon = 3$ for our approach).

$$\|w\|_2 = \epsilon, \ if \ \|w\|_2 > \epsilon. \tag{7}$$

To learn the parameters for minimizing the loss, we use a Stochastic Gradient Descent (SGD) with the Adam update rule [8] to train the model until the loss has converged.

3 PROPOSED APPROACH

In this section, we describe our proposed approach for the Financial Aspect and Sentiment Prediction task with Deep neural networks (Deep-FASP).

Figure 1 shows an overview of our proposed approach for predicting the aspects and the sentiment score of a given text. It consists of five steps from an input to the predicted output.

- **Input.** Each headline and post is a sequence of words.

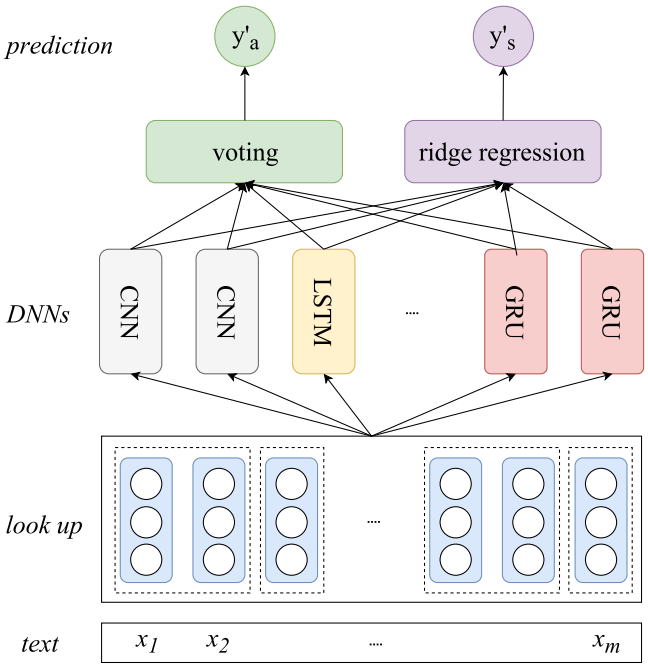

Figure 1: An overview of our proposed approach Deep-FASP.

- **Look up.** Those words are represented as word embeddings.
- **DNNs**. The sequence of word embeddings is used as an input to multiple DNNs such as CNNs, LSTMs, and GRUs. Each DNN outputs their predicted labels for the aspect classification task, and outputs their measured sentiment scores for the task of sentiment prediction.
- **Prediction.** Finally, the outputs from multiple DNNs are aggregated for predicting the final results. The output of aspect classification is the predicted label with the highest votes based on the votes from all DNNs. For sentiment score prediction, we use a ridge regression to combine the outputs from different DNNs to produce the final sentiment score.

For both tasks, each DNN model (e.g., CNNs or LSTMs) has the same archietecture for retrieving the representation of a given text. Given the text representation learned by a DNN, we design a customized fully connected and output layer for each task, which will be described in the following.

3.1 Sentiment Prediction

Figure 2 illustrates the model architecture for predicting the sentiment score of a given text. The fully connected layer for this task has 30 units as the settings in [2], and the final sentiment score is predicted with a linear regression on those 30 units.

We use the mean squared error as below for the loss function of sentiment prediction.

$$L_s = \frac{\sum_{i=1}^{n}(y'_s(i) - y_s(i))^2}{n} \quad (8)$$

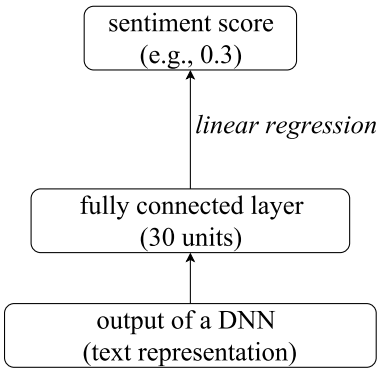

Figure 2: The Deep-FASP architecture for sentiment prediction.

where n is the number of training instances, $y'_s(i)$ and $y_s(i)$ denote the predicted and ground truth sentiment scores for i-th instance, respectively.

3.2 Aspect Prediction

Figure 3 shows the architecture for predicting aspect labels. As we can see form the figure, the fully connected layer for this task has 80 units, and the final output layer has the same number of units as the distinct aspect labels.

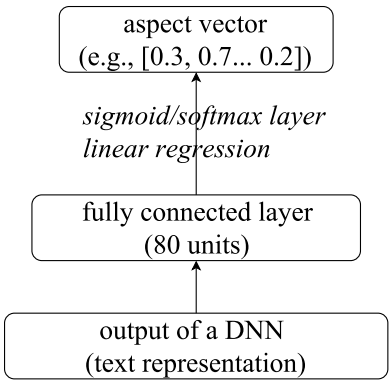

Figure 3: The Deep-FASP architecture for aspect prediction.

The *sigmoid* or *softmax* function is applied at the end in order to transform the output layer into a boolean vector where ones denote corresponding aspects. Applying the sigmoid or softmax function denotes that we treat the problem as multi-label or multi-class problem, respectively. Although the first released dataset was a multi-labeled dataset, the final one has only one multi-labeled instance which is closer to a multi-class problem. In this regard, we tested both sigmoid and softmax layers for the final prediction.

For the aspect prediction task, we use the cross-entropy loss as our loss function to optimize, which is defined as below:

$$L_a = -\frac{1}{n}\sum_{i=1}^{n}[y_a(i)\log y'_a(i) + (1 - y_a(i))log(1 - y'_a(i))] \quad (9)$$

where $y'_a(i)$ and $y_a(i)$ denote the predicted and ground truth aspect vectors for i-th instance, respectively.

Note that when the final layer is a sigmoid layer for predicting aspects on the test set, the values in the final aspect vector are rounded to 0 or 1 (e.g., [0, 1, ..., 0]). In contrast, when the final layer is a softmax layer, only the label with the highest value will be remained.

3.3 Pre-processing Texts

We pre-process the raw data (headlines and posts) as follows before the training stage.

- Converts both texts and targets as lowercase.
- Targets are replaced by the $target$ token.
- URLs are replaced by the URL token.
- @mentions are replaced by the $mention$ token.
- Any letter repeated more than two times in a text is replaced by two repetitions of that letter, e.g., "fooooo" is replaced by "foo".

3.4 Pre-trained Word Embeddings

Considering the training set is not large for the given task, we used a Twitter corpus[4] for sentiment analysis to pre-train word embeddings. We used a CNN with the filter sizes [1, 2, 3] to pre-train these word embeddings.

Figure 4 shows nearest words of the word "good" based on pre-trained word embeddings. As we can see from the figure, similar words in terms of sentiment are nearby each other in the latent space. These embeddings were used to initialize the word embeddings for both sentiment and aspect prediction tasks.

4 RESULTS

In this section, we describe the experimental results based on 5-cross validation on the released training set for sentiment and aspect prediction tasks.

4.1 Sentiment Prediction

Table 4 shows the results for predicting sentiment scores based on DNNs and two simple baselines. The results are obtained by using 5-cross validation. PredictZero always predicts zero for any given text, and SVR is a SVM model for regression where each word is denoted as a boolean feature.

As we can see from the table, CNN [1, 2, 3] provides the best performance as a single predictor followed by CNN [3, 4, 5] and CNN [5, 6, 7]. Overall, CNNs perform better than RNNs in our experiment with 5-cross validation. The ensemble approach using a ridge regression (the regularization parameter is tuned to 95 based on the results with 5-cross validation) for different models with different settings in the table except Bi-GRU outperforms any single model, with 0.0837 and 0.4683 for MSE and R2 score respectively.

Finally, we trained this ensemble model based on the whole training dataset in order to predict the sentiment scores for the texts in the test dataset.

Table 4: The results of financial sentiment prediction based on 5-cross validation in terms of MSE and R2 score. The values in [] denotes filter sizes used for CNNs.

No.	Approach	MSE	R2
1	**SVR**	0.1566	0.0115
2	**PredictZero**	0.1737	-0.0984
3	**CNN [1, 2, 3]**	0.0973	0.3841
4	**CNN [3, 4, 5]**	0.0996	0.3705
5	**CNN [5, 6, 7]**	0.1064	0.3278
6	**Bi-GRU**	0.1110	0.2985
7	**Bi-LSTM**	0.1076	0.3196
8	**GRU**	0.1067	0.3262
9	**Deep-FASP (Ensemble)**	**0.0926**	**0.4144**

4.2 Aspect Prediction

Table 5 shows the results for aspect prediction based on DNNs and two simple baselines using Ridge Regression and Random Forest for aspect classification where each word is denoted as a boolean feature. The results are obtained by using 5-cross validation.

Similar to the results for sentiment prediction, CNNs perform better than RNNs in the aspect prediction task as well. The voting strategy with those models in the table did not yield better performance compared to using CNN[1, 2, 3]. We also observe that using the sigmoid layer (CNN[1, 2, 3] - ML), i.e., treat the problem as a multi-label classification task, did not improve the performance but decrease the performance significantly. Therefore, we assume the aspect prediction task as a multi-class problem, and used several CNN[1, 2, 3] models with different settings for aspect prediction without incorporating other models. As a result, the voting strategy has slightly better performance with 0.6530 for the accuracy.

Finally, we trained this voting approach based on the whole training dataset in order to predict the aspect labels for the texts in the test dataset.

5 CONCLUSIONS AND FUTURE WORK

This paper descried Deep-FASP which is an ensemble approach for sentiment and aspect prediction tasks in the financial domain using deep learning approaches such as CNNs without handcrafted features. The results based on 5-cross validation on the training dataset show that CNNs perform better than RNNs such as LSTMs or GRUs, and the ensemble approach provides the best performance in both tasks compared to any single model. As the final results of the challenge is not available at the time of writing, more details of

[4] http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/

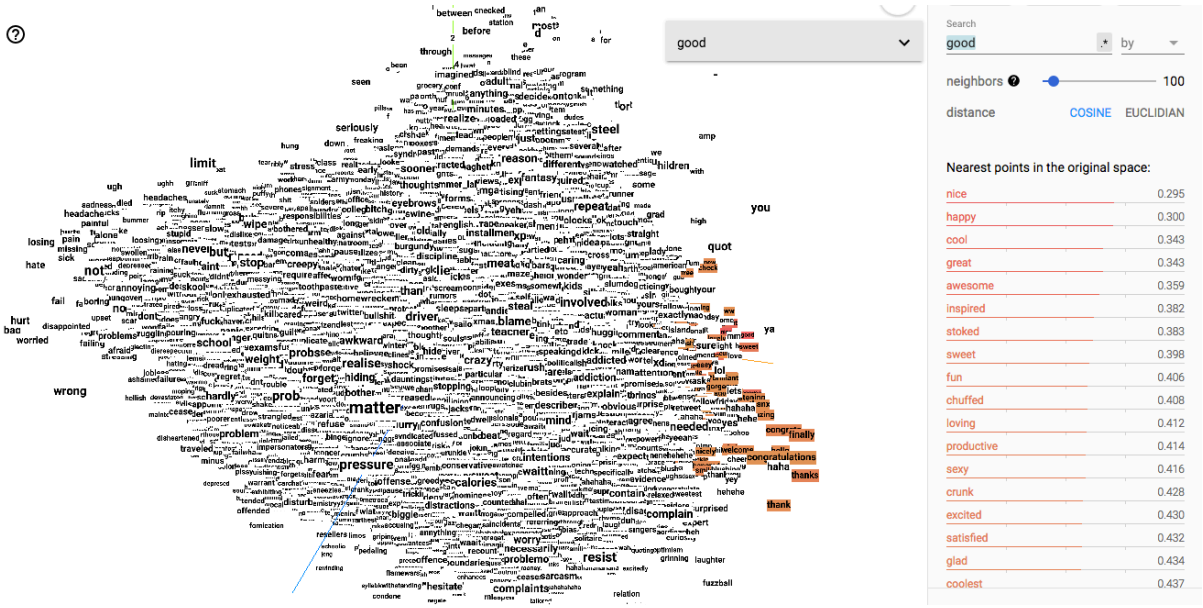

Figure 4: Nearest words of the word "good" from the pre-trained word embeddings.

Table 5: The results of financial aspect prediction based on 5-cross validation in terms of accuracy. The values in [] denotes filter sizes used for CNNs. ML denotes multi-label.

No.	Approach	Accuracy
1	**Ridge Regression**	0.3634
2	**Random Forest**	0.2749
3	**CNN [1, 2, 3]**	0.6436
4	**CNN [3, 4, 5]**	0.6180
5	**CNN [5, 6, 7]**	0.5940
6	**Bi-GRU**	0.5085
7	**Bi-LSTM**	0.4915
8	**GRU**	0.5274
9	**CNN [1, 2, 3] - ML**	0.5581
10	**Voting**	**0.6530**

our approach and the performance compared to other participated teams will be updated on https://github.com/parklize/FIQA.

6 ACKNOWLEDGMENTS

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289 (Insight Centre for Data Analytics).

REFERENCES

[1] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[2] Mathieu Cliche. 2017. BB_twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs. *CoRR* abs/1704.0 (2017). arXiv:1704.06125 http://arxiv.org/abs/1704.06125

[3] Keith Cortis, André Freitas, Tobias Daudert, Manuela Huerlimann, Manel Zarrouk, Siegfried Handschuh, and Brian Davis. 2017. Semeval-2017 task 5: Fine-grained sentiment analysis on financial microblogs and news. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. 519–535.

[4] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. 2013. Learning Hierarchical Features for Scene Labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1915–1929.

[5] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (nov 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[6] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *The 52nd Annual Meeting of the Association for Computational Linguistics*.

[7] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Conference on Empirical Methods on Natural Language Processing*.

[8] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*. 1097–1105.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[11] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. 1990. Handwritten Digit Recognition with a Back-propagation Network. In *Advances in Neural Information Processing Systems*. 396–404.

[12] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[13] Qi Zhang, Yeyun Gong, Jindou Wu, Haoran Huang, and Xuanjing Huang. 2016. Retweet Prediction with Attention-based Deep Neural Network. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management - CIKM '16 (CIKM '16)*. ACM, New York, NY, USA, 75–84. https://doi.org/10.1145/2983323.2983809

[14] Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep Learning based Recommender System: A Survey and New Perspectives. *CoRR* abs/1707.0 (2017). arXiv:1707.07435 http://arxiv.org/abs/1707.07435