Domain-Aware Sentiment Classification with GRUs and CNNs

Guangyuan Piao and John G. Breslin

Insight Centre for Data Analytics, Data Science Institute,
National University of Ireland Galway, Ireland,
IDA Business Park, Galway, Ireland
{guangyuan.piao}@insight-centre.org,
{john.breslin}@nuigalway.ie

Abstract. In this paper, we describe a deep neural network architecture for domain-aware sentiment classification task with the purpose of the sentiment classification of product reviews in different domains and evaluating nine pre-trained embeddings provided by the semantic sentiment classification challenge at the 15th Extended Semantic Web Conference. The proposed approach combines the domain and the sequence of word embeddings of the summary or text of each review for Gated Recurrent Units (GRUs) to produce the corresponding sequence of embeddings by being aware of the domain and previous words. Afterwards, it extracts local features using Convolutional Neural Networks (CNNs) from the output of the GRU layer. The two sets of local features extracted from the domain-aware summary and text of a review are concatenated into a single vector, and are used for classifying the sentiment of a review. Our approach obtained 0.9643 F1-score on the test set and achieved the 1st place in the first task of the Semantic Sentiment Analysis Challenge at the 15th Extended Semantic Web Conference.

1 Introduction

Sentiment analysis plays an important role in many domains such as predicting the market reaction in the financial domain [4]. Therefore, many approaches have been proposed for classifying sentiment labels as well as predicting sentiment scores. With the advance of deep learning [13] approaches such as Convolutional Neural Networks (CNNs) [14] for processing data in the form of multiple arrays, or Recurrent Neural Networks (RNNs) such as Long Short-Term Memory neural networks (LSTMs) [7] for tasks with sequential inputs, many research areas have made a significant progress. These research areas include computer vision [5, 12], natural language processing (NLP) [9, 10], and recommender systems [19].

Recently, many approaches [3, 16] leveraging deep neural networks (DNNs) have been proposed for sentiment analysis as well. For example, an ensemble approach using several DNNs such as CNNs and LSTMs proposed by [3] won one of the sentiment prediction tasks on Twitter[1] at the sentiment analysis challenge SemEval2017[2].

In this paper, we also propose a DNN architecture for the embedding evaluation and semantic sentiment analysis challenge held in conjunction with the 15th Extended Semantic Web Conference[3]. Our proposed approach achieved the best performance compared to the solutions from the other participants in the first task of the challenge.

The first task of the challenge has two main goals as follows:

1. Given nine pre-trained word embeddings and two baseline embeddings, the objective is to evaluate these embeddings in the context of sentiment classification for domain-specific reviews.

2. Propose a sentiment classification model and compare the classification performance on a test set with the models proposed by the other participants.

To this end, the challenge provides a training dataset which consists of one million reviews covering 20 domains. Each training instance consists of a summary, review text, the domain of the review, and its sentiment label (i.e., positive or negative).

Each participated system can train their systems based on the training set and predicts the sentiment labels for the reviews in the test set provided by the organizers. Those systems are evaluated by the F1-score on the test set, which can be defined as follows:

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{1}$$

$$precision = \frac{TP}{TP + FP} \tag{2}$$

$$recall = \frac{TP}{TP + FN} \tag{3}$$

where $tp \in TP$ (true positive) is defined when a sentence is correctly classified as positive, and $fp \in FP$ (false positive) denotes when a positive review is classified as negative. A true negative ($tn \in TN$) denotes when a negative sentence is correctly classified as negative, and a false negative ($fn \in FN$) denotes the case of a negative sentence is classified as positive.

2 System Description

In this section, we describe the architecture of our proposed system using domain-aware GRUs and CNNs with pre-trained word embeddings for the sentiment classification of reviews. Figure 1 provides an overview of our proposed model for the semantic sentiment analysis challenge.

[1] `https://twitter.com`

[2] `http://alt.qcri.org/semeval2017/index.php?id=tasks`

[3] `https://2018.eswc-conferences.org/`

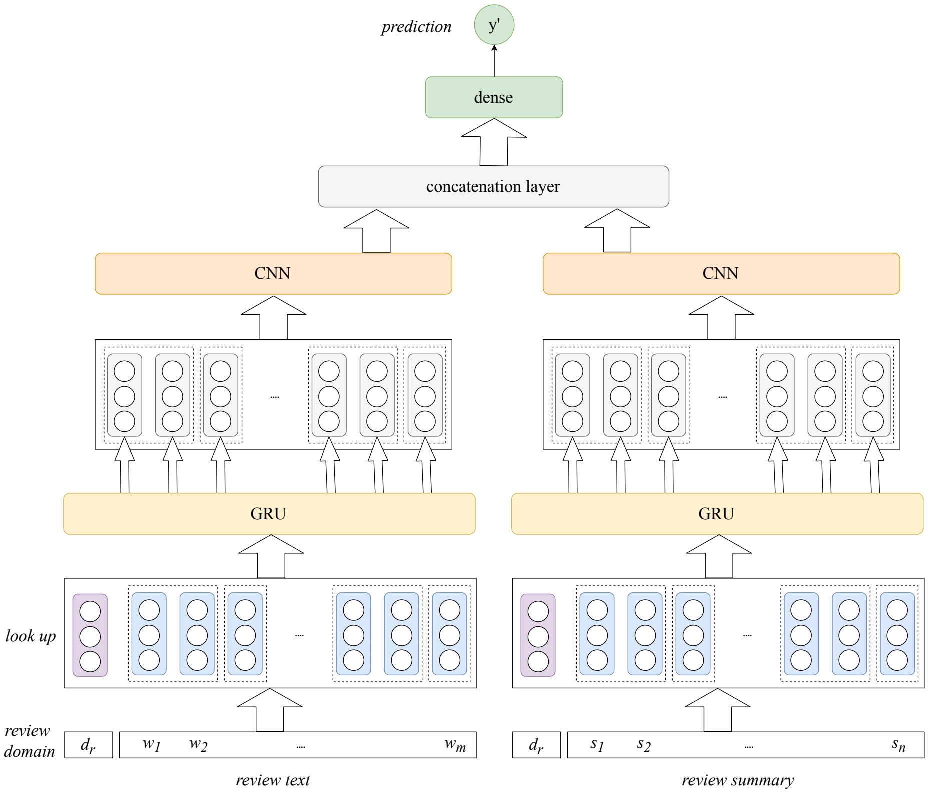

Fig. 1: The proposed system architecture for predicting the sentiment of a given review. A review is represented as $r = \{s, x, d_r\}$ where s, x, and d_r denote the summary, review text, and the domain of r, respectively. First, we represent the text x of a review as a sequence of embeddings: $\{E_{d_r}, E_{w_1}, \ldots, E_{w_m}\}$ where E_{d_r} denotes the domain embedding of r, and $E_{w_1}, \ldots, E_{w_m}$ denote the set of word embeddings for the sequence of words in x. Afterwards, the sequence of embeddings is feed into the GRU layer. The output of this GRU layer is a sequence of embeddings with the memory of the domain as well as previous words, and is used as an input to a CNN layer to extract local features of the text x. Similarly, we extract the local features of the summary of r. Finally, the local features of the text and summary of a review are concatenated, and fed into a dense layer in order to predict the sentiment label of r.

We represent each review r as two parts using the review text and summary. For example, the first part consists of the domain of r (d_r), and the sequence of words $\{w_1, \ldots, w_m\}$ in the text of r. The second part consists of the domain of r (d_r), and the sequence of words $\{s_1, \ldots, s_n\}$ in the summary of r. Each training instance can be represented as $t_i = \{r_i, y_i\}$ where y denotes the sentiment label (i.e., 1 or 0) of r. As we can see from Figure 1, there are mainly four layers in our proposed model. The first part of review with its domain and text goes through the following layers.

Look up. The look up layer transforms a review r into the domain and the sequence of word embeddings: $\{E_{d_r},\ E_{w_1},\ \ldots,\ E_{w_m}\}$ where E_{d_r} denotes the domain embedding of r, and $E_{w_1}, \ldots, E_{w_m}$ denote the set of pre-trained word embeddings for the sequence of words in r's text. The domain embeddings are also parameters of our model and will be learned through the training process.

GRU layer. The sequence of embeddings $\{E_{d_r}, E_{w_1}, \ldots, E_{w_m}\}$ for a review from the look up layer is used as an input to a Gated Recurrent Unit [2] (GRU) layer. This layer aims to transform the domain and word embeddings of a review r into a sequence of embeddings (hidden states in GRUs) for r where each embedding at position k is aware of the domain as well as the previous words. GRUs and LSTMs are RNNs that have been designed to resolve the issue of vanishing gradient problem in the vanilla RNNs. We opted to use GRUs instead of LSTMs as the former one has fewer number of parameters to tune, which is suitable for the limited computing resources with a large number of training instances. We provide more details about GRU in Section 2.1.

CNN layer. Recurrent Neural Networks such as GRU is good at capturing the long-term dependencies of words, e.g., capturing sarcasm of a text. In contrast, the CNN layer aims to extract local features which are useful for the sentiment classification. CNNs apply a set of convolutional filters to the input matrix with a filtering matrix $f \in \mathbb{R}^{h \times d}$ where h is the filter size. This filter size denotes the number of the output embeddings from the GRU layer it spans. In our proposed model, we used $\{3,\ 4,\ 5\}$ as the filter sizes and 200 filters for the first part of a review with its review text. For the second part of a review with its review summary, we used $\{1,\ 2,\ 3\}$ as the filter sizes and 100 filters in total. We use C_t and C_s to denote the local features extracted from the two parts of a review with its text and summary. We provide more details about CNNs in Section 2.2.

Concatenation layer. Similar to the first part of a review, the second part of it with its summary also goes through the abovementioned layers. In the concatenation layer, the outputs from both parts are concatenated together to represent the review: $C = [C_s, C_t]$. This vector of the review is used as an input to a dense layer.

Dense layer. Finally, there is a dense layer with 80 hidden nodes (D) to shrink the dimension of C to 80 dimensions. The dense layer is then used for classifying the sentiment of a review.

$$y' = W \cdot D + b \tag{4}$$

where W is the weight parameters for this dense layer and b denotes a bias term.

Loss function. We use the cross-entropy as our loss function, and the objective is to minimize the loss over all training examples.

$$L = \sum_i (y_i \times -log(\sigma(y'_i)) + (1 - y_i) \times -log(1 - \sigma(y'_i))) \tag{5}$$

where σ is the *sigmoid* function, $\sigma(y'_i)$ is the predicted sentiment score, and y_i is the ground truth sentiment label of i-th training instance.

2.1 Gated Recurrent Units

Gated Recurrent Units along with LSTMs are part of the RNN family, which is designed to deal with sequential data. For each element of the sequence (e.g., each word of a review), a vanilla RNN uses the current element embedding and its previous hidden state to output the next hidden state:

$$h_t = f(W_h \cdot x_t + U_h \cdot h_{t-1} + b_h) \tag{6}$$

where $h_t \in \mathbb{R}^m$ denotes the hidden state at t-th element (or at time step t), x_t is the current element embedding, $W_h \in \mathbb{R}^{m \times d}$ and $U_h \in \mathbb{R}^{m \times m}$ are weight matrices, b_h denotes a bias term, and $f(\cdot)$ is a non-linear activation function such as *tanh*. One main problem of this vanilla RNN is that it suffers from the vanishing gradient problem.

This problem has been solved in LSTMs with a memory cell instead of a single activation function. GRUs (see Figure 2) can be seen as a variation on the LSTMs but with a smaller number of parameters, and both GRUs and LSTMs produce comparative results in many problems. To resolve the vanishing gradient problem of a vanilla RNN, GRUs use *update* and *reset* gates, which control the information to be passed to the output. The formula for the output gate is as follows:

$$z_t = \sigma(W_z \cdot x_t + U_z \cdot h_{t-1}) \tag{7}$$

where W_z and U_z are weight matrices and σ denotes the *sigmoid* function. This gate determines how much information from the past can be passed along to the future. In contrast, the reset gate decides how much information from the past should be forgotten using the formula as below.

$$r_t = \sigma(W_r \cdot x_t + U_r \cdot h_{t-1}) \tag{8}$$

where W_r and U_r are weight matrices of the reset gate. The output of the reset gate is used to calculate the current memory content as follows:

$$h'_t = tanh(W_h \cdot x_t + r_t \odot U_h \cdot h_{t-1}) \tag{9}$$

where W_h and U_h are weight matrices. The element-wise product of r_t (which ranges from 0 to 1) and $U_h \cdot h_{t-1}$ determines how much information to remove from the previous time steps.

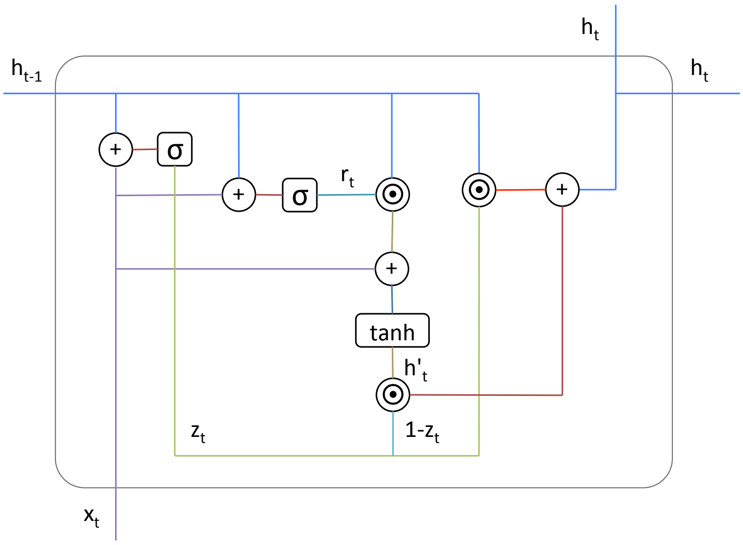

Fig. 2: The architecture of a Gradient Recurrent Unit based on the figure from https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be.

Finally, the current hidden state is determined by using the current memory content h'_t and the previous hidden state h_{t-1} with the update gate.

$$h_t = (1 - z_t) \odot h'_t + z_t \odot h_{t-1} \tag{10}$$

Based on these formulas, we can observe that the vanilla RNN can be see as a special case of a GRU when $z_t = 0$ and $r_t = 1$.

2.2 Convolutional Neural Networks

Here we give an overview of CNNs. A smaller version of the used model is presented in Figure 3. The input of CNNs in our model is the output of the GRU layer, which consists of a sequence of hidden states. We use GRU matrix $G \in \mathbb{R}^{m \times d}$ to denote this sequence of hidden states. In contrast to using the sequence of word embeddings as an input, using those hidden states obtained through the GRU layer embodies the information from previous words as well as the domain of a review.

We then apply a set of convolutional filters to the GRU matrix with a filtering matrix $f \in \mathbb{R}^{h \times d}$ where h is the filter size which denotes the number of hidden states it spans. The convolutional operation is defined as follows:

$$c_i = f\left(\sum_{j,k} w_{j,k}(G_{[i:i+h-1]})_{j,k} + b_c\right) \tag{11}$$

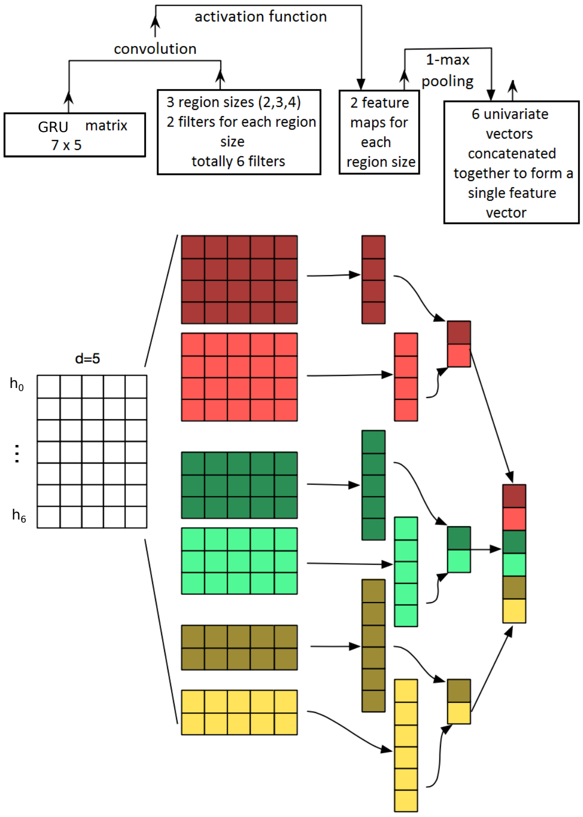
activation function
convolution
1-max pooling
GRU matrix 7 x 5
3 region sizes (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
d=5
h0
⋮
h6

Fig. 3: The architecture of a smaller version of the used CNNs. Figure is taken from [20] with some modifications.

where $b_c \in \mathbb{R}$ is a bias term, and $f(\cdot)$ is a non-linear function (we use the *ReLu* function in our model). The output $c \in \mathbb{R}^{m-h+1}$ is a concatenated vector which consists of c_i over all windows of hidden states in G. We can define the number of filtering matrices which can be applied to learn different features. As mentioned earlier, we used $\{3, 4, 5\}$ as the filter sizes and 200 filters for the first part of a review with its review text. For the second part of a review with its review summary, we used $\{1, 2, 3\}$ as the filter sizes and 100 filters in total.

In order to extract the most important feature for each convolution, we use the $max - pooling$ operation $c_{max} = max(c)$. Therefore, this operation extracts the most important feature with regardless of the location of this feature in the sequence of hidden states. Afterwards, the c_{max} of each filter is concatenated into a single vector $\mathbf{c}_{max} \in \mathbb{R}^p$ where p is the total number of filters. For example, the total number of filters for the first part of a review is $3 \times 200 = 600$, and that for the second part of the review is $3 \times 100 = 300$ in our model. These two vectors are concatenated together in the concatenation layer as we can see from the Figure 1 which is followed by a dense layer.

2.3 Training

This section provides some details about training our proposed model. The parameters to be learned through the training process include the domain embeddings, the parameters associated with GRUs and CNNs as well as the nodes in the dense layer.

Validation set. In order to train our proposed model, we used 10,000 out of the 1 million provided reviews which are evenly distributed over 20 domains as the validation set. Therefore, the validation set consists of 500 training instances for each domain where 250 of them are positive ones and the rest are negative ones.

Training. We train our proposed model nine times with each of the nine word embeddings provided. As we also aim to evaluate these pre-trained word embeddings, we do not further optimize the weights of these embeddings, i.e., the weights of word embeddings are fixed. To learn the parameters of our proposed approach for minimizing the loss, we use a mini-batch gradient decent with 128 as the batch size and use the Adam update rule [11] to train the model on the training set, and stop the training if there is no improvement on the validation set within the subsequent 10 epochs.

Regularization. We further use the dropout [18] for regularization with a dropout rate $p = 0.5$. Dropout is one of the widely used regularization approaches for preventing overfitting in training neural network models. In short, individual nodes are "disabled" with probability $1 - p$, and then the outputs with the set of disabled nodes of a hidden layer are 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.

3 Results

In this section, we discuss the results on the validation set using our model with different word embeddings, and the results on the test set compared to all the participated systems.

3.1 Results on the Validation Set

Table 1 shows the best classification accuracy ($accuracy = \frac{TP+TN}{TP+FP+TN+FN}$) on the validation set using the nine provided embeddings and the two baseline ones using our proposed architecture. The provided embeddings are denoted as emb_{dimensionality of word embeddings}_{epochs for training}.

Instead of using each model with different word embeddings, we further investigated an ensemble approach which uses a simple voting scheme for 11 variants of our proposed model with different word embeddings or training epochs. The final sentiment classification is determined by the votes from the 11 classification results where the sentiment label with a higher number of votes is selected. Based on the results, we have the following observations:

- The classification performance with the nine provided embeddings outperforms the one with the two baseline embeddings.
- The accuracy of sentiment classification increases with a higher dimensionality or number of epochs of the provided word embeddings.
- Overall, emb_512_50, which uses 512 as the dimensionality of word embeddings and was trained for 50 epochs, provides the best performance among all embeddings.
- The simple ensemble approach improves the classification accuracy further compared to using a single word embeddings.

Table 1: The accuracy of sentiment classification on the validation set with the nine provided embeddings.

embedding	**accuracy**
glove	0.9525
amazon_we	0.9122
emb_128_15	0.9578
emb_128_30	0.9560
emb_128_50	0.9544
emb_256_15	0.9576
emb_256_30	0.9576
emb_256_50	0.9578
emb_512_15	0.9568
emb_512_30	0.9598
emb_512_50	0.9605
ensemble approach	**0.9657**

3.2 Results on the Test Set

The final test set released by the challenge organizers consists of 10,000 examples, which are evenly distributed over 20 domains. Table 2 shows the sentiment classification results on the test set from all the participated systems in this year and those in the last year [17], which are released by the challenge organizers.

As we can see from the table, our proposed approach provides the best performance among all participants in terms of F1-score. In addition, we observe that the performance of the participated systems for this year is significantly improved compared to that for the previous year.

Table 2: The comparison of our approach and the other participated ones for the sentiment classification on the test set.

year	ranking	F1-score
2018	**#1 (our approach)**	**0.9643**
	#2	0.9561
	#3	0.9356
	#4	0.9228
	#5	0.8823
	#6	0.8743
	#7	0.7153
	#8	0.5243
2017	#1 [1]	0.8675
	#2 [6]	0.8424
	#3 [8]	0.8378
	#4 [15]	0.8112

4 Conclusions

In this paper, we presented our system architecture for the Semantic Sentiment Analysis challenge at the 15th Extended Semantic Web Conference. Our goal was to use and compare pre-trained word embeddings with proposed architecture, which combines the advantages of both GRUs and CNNs for classifying the sentiment labels of domain-specific reviews. For future work, the current model can be improved by using bi-directional GRUs to learn the hidden states of GRUs based on both previous and post information. In addition, the ensemble approach with a hard voting scheme can be replaced by another ensemble approach such as a soft voting scheme.

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. Atzeni, M., Dridi, A., Reforgiato Recupero, D.: Fine-Grained Sentiment Analysis on Financial Microblogs and News Headlines BT - Semantic Web Challenges. pp. 124–128. Springer International Publishing, Cham (2017)

2. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
3. Cliche, M.: BB_twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs. CoRR abs/1704.0 (2017), http://arxiv.org/abs/1704.06125
4. Cortis, K., Freitas, A., Daudert, T., Huerlimann, M., Zarrouk, M., Handschuh, S., Davis, B.: 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). pp. 519–535 (2017)
5. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning Hierarchical Features for Scene Labeling. IEEE Transactions on Pattern Analysis and Machine Intelligence 35(8), 1915–1929 (2013)
6. Federici, M., Dragoni, M.: A Knowledge-Based Approach for Aspect-Based Opinion Mining BT - Semantic Web Challenges. pp. 141–152. Springer International Publishing, Cham (2016)
7. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation 9(8), 1735–1780 (nov 1997), https://doi.org/10.1162/neco.1997.9.8.1735
8. Iguider, W., Reforgiato Recupero, D.: Language Independent Sentiment Analysis of the Shukran Social Network Using Apache Spark BT - Semantic Web Challenges. pp. 129–132. Springer International Publishing, Cham (2017)
9. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. In: The 52nd Annual Meeting of the Association for Computational Linguistics (2014)
10. Kim, Y.: Convolutional neural networks for sentence classification. In: Conference on Empirical Methods on Natural Language Processing (2014)
11. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet Classification with Deep Convolutional Neural Networks. In: Advances in Neural Information Processing Systems. pp. 1097–1105 (2012)
13. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553), 436–444 (2015)
14. LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E., Jackel, L.D.: Handwritten Digit Recognition with a Back-propagation Network. In: Advances in Neural Information Processing Systems. pp. 396–404 (1990)
15. Petrucci, G., Dragoni, M.: The IRMUDOSA System at ESWC-2017 Challenge on Semantic Sentiment Analysis BT - Semantic Web Challenges. pp. 148–165. Springer International Publishing, Cham (2017)
16. Piao, G., Breslin, J.G.: Financial Aspect and Sentiment Predictions with Deep Neural Networks: An Ensemble Approach. In: Financial Opinion Mining and Question Answering Workshop at The Web Conference (WWW). ACM (2018)
17. Reforgiato Recupero, D., Cambria, E., Di Rosa, E.: Semantic Sentiment Analysis Challenge at ESWC2017 BT - Semantic Web Challenges. pp. 109–123. Springer International Publishing, Cham (2017)
18. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15(1), 1929–1958 (2014)
19. Zhang, S., Yao, L., Sun, A.: Deep Learning based Recommender System: A Survey and New Perspectives. CoRR abs/1707.0 (2017), http://arxiv.org/abs/1707.07435

20. Zhang, Y., Wallace, B.C.: A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. CoRR abs/1510.0 (2015), http://arxiv.org/abs/1510.03820