A Comprehensive Verification of Transformer in Text Classification

Xiuyuan Yang, Liang Yang(✉), Ran Bi, and Hongfei Lin

Dalian University of Technology, Dalian 116023, Liaoning, China
yxy815754134@mail.dlut.edu.cn,
{liang,biran,hflin}@dlut.edu.cn

Abstract. Recently, a self-attention based model, named Transformer, is proposed in Neural Machine Translation (NMT) domain, and outperforms the RNNs based seq 2seq model in most cases, hence it becomes the state-of-the-art model for NMT task. However, some studies find that the RNNs based model integrated with the Transformer structures could achieve almost the same experiment effect as the Transformer on the NMT task. In this paper, following the previous researches, we intend to further verify the performance of Transformer structures on the text classification task. Based on RNNs model, we gradually add each part of the Transformer block and evaluate their influence on the text classification task. We carry out the experiments on NLPCC2014 and dmsc_v2 datasets, and the experiment results show that multi-head attention mechanism and multiple attention layers could improve the performance of the model on the text classification task. Furthermore, the visualization of the attention weights also illustrates that multi-head attention outperforms the traditional attention mechanism.

Keywords: RNNs · Transformer · Text classification

1 Introduction

Since Deep Learning methods are widely implemented in Natural Language Processing (NLP) tasks, great improvements have been achieved in recent years. Initially, Recurrent Neural Networks (RNNs) [1], such as Long Short-Term Memory (LSTM) [2] networks and the Gated Rectified Unit (GRU) [3], which can better capture the long-range context information, are commonly used in various research domains.

For example, in the text classification task, RNNs are used for encoding text information. Bi-RNNs aggregate information from both directions of sentence to get a comment on the word and merge it into the text vectors. For Lexical analysis, including word segmentation, part-of-speech tagging (POS) [4], named entity recognition (NER) [5], etc., RNNs can carry more deep information such as syntax structure and dependencies. In addition, RNNs are skilled in dealing with variable-length input sentences and can be used as a language model to generate sentences, thus it is a natural choice for using them as encoder and decoder in the neural machine translation (NMT) systems.

M. Sun et al. (Eds.): CCL 2019, LNAI 11856, pp. 207–218, 2019.
https://doi.org/10.1007/978-3-030-32381-3_17

Recently, Vaswani et al. [6] propose Transformer as a novel feature extractor for NMT task and it outperforms RNNs in most cases. Tang et al. [7] hypothesize that the reasons, why the Transformer has a strong performance, are the abilities of semantic feature extraction and capturing long-range dependencies. Their experimental results show that Transformer outperforms distinctly better than RNNs on word sense disambiguation. Meanwhile, Transformer model and RNNs have a close performance in modeling subject-verb agreement over long distance. Unlike the above mentioned works, Radford et al. [8] evaluate the performance of models by introducing different feature extractors into different tasks. The experiments show that on eight different NLP tasks with large datasets, the feature extractor is only changed from Transformer to LSTM and the average of eight tasks scores drop points under the same conditions.

Due to these analyses, we can find that the performance of Transformer has obvious advantages over native RNNs. Does it mean that we can just simply replace RNNs with Transformer in most tasks? Actually, it is not the case. As we know, the Transformer block makes Transformer work well, which is a small system composed of several components such as multi-head attention, layer norm, and feed-forward network. Domhan et al. [9] find that RNNs can also make close achievements to Transformer by gradually adding each part of Transformer block on NMT task.

Inspired by this, following the researches from Domhan et al. [9], we intend to further verify the performance of Transformer block on the text classification task, by gradually adding each part of Transformer block to RNNs and evaluate their impact in this paper.

Our observations and analysis conclusions in this paper are as follows:

- Transform not always performs well on text classification.
- Unlike NMT, adding feed-forward to RNN based models has no obvious effect on the text classification task.
- The contributions of multi-head attention and multiple attention layer are crucial in different classification tasks.
- Attention visualization demonstrates the strong effects of multi-head attention.

2 Related Work

Before the Transformer is proposed, RNNs with attention are the state-of-the-art models for most cases in NLP. Bahdanau et al. [10] firstly introduce the single layer attention-like mechanism for RNNs based NMT models, to solve the problem of long-term gradient disappearance in the RNNs. Luong et al. [11] further perform different single layer attention mechanisms for RNNs based NMT models. Yang et al. [12] propose a hierarchical attention mechanism for RNNs, which is a two levels of attention structure, capturing the word and sentence representation on the document classification.

Recently, Vaswani et al. [6] propose Transformer as a novel feature extractor for NMT task and it outperforms RNNs in most cases. The Transformer is based solely on attention mechanisms, dispensing with recurrence entirely. Their experiments show

that the Transformer models to be superior in quality while being more parallelizable and requiring significantly less time to train.

However, Domhan et al. [9] find that RNNs can make close achievements to Transformer by applying some parts of the Transformer block into RNNs on NMT tasks. Chen et al. [13] demonstrate that several modeling improvements (including multi-head attention and layer normalization) as well as optimization techniques are applicable across different model architectures. Moreover, they further propose new model architectures that combine components from the RNMT+ (an enhanced version of RNNs on NMT) and the Transformer model, and achieve better results than both individual architectures.

Different from the above research on NMT, this paper proposes new improvements to RNNs on text classification, based on the strategy adopted by Domhan et al. [9]. The effect for each part of Transformer block is different, which complements the relationship between RNNs and Transformer on text classification.

3 Model Architecture

The basic architecture of the model is based on RNNs. To get better results, we choose to use GRU and LSTM alternately instead of using a single RNNs model. The baseline model consists of several parts as shown in the solid line in Fig. 1, which contain an input embedding layer, an encoder layer, an attention-pooling layer and an output layer.

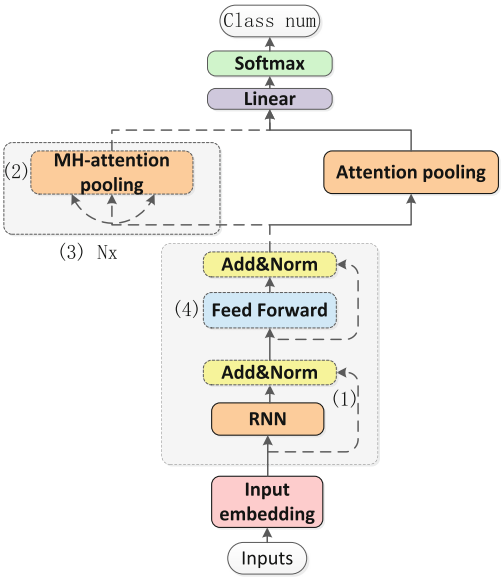

Fig. 1. The architecture of our model

In order to clearly explain how we add structural components to the basic model, we describe each structural component as shown in the dotted line in Fig. 1: (1) A residual [14] connection around each of the two sub-layers, followed by layer normalization. (2) MH-attention pooling. (3) Multiple-attention layer. (4) A feed-forward layer.

3.1 RNN Encoder Layer

Through word embedding layer, we get the input text with words $x_t, t \in [1, T]$. After that, the text representations are constructed by using Bi-Recurrent Neural Networks (Bi-RNNs), which combine a Long Short-Term Memory (LSTM) network [15] and a Gated Rectified Unit (GRU) [16], showed as following:

$$\begin{aligned} h_t, ce_t &= \overrightarrow{\overleftarrow{LSTM}}(x_t, h_{t-1}, ce_t) \\ h_t^* &= \overrightarrow{\overleftarrow{GRU}}(h_{t-1}, h_{t-1}^*) \end{aligned} \tag{1}$$

Where h_t is the hidden state and ce_t is cell state for Bi-LSTM, and the h_t^* is the output hidden state for Bi-GRU.

3.2 Attention Mechanism

All attention mechanisms take a set of query vectors $Q \in \mathbb{R}^{M \times d}$, key vectors $K \in \mathbb{R}^{N \times d}$, and value vectors $V \in \mathbb{R}^{N \times d}$, in order to produce one context vector $C \in \mathbb{R}^{M \times d}$, which is a linear combination of the value vectors.

$$\text{Attention}(Q, K, V) = \text{softmax}(QWK^T)V \tag{2}$$

Attention-Pooling for Text Classification

In the baseline model, we construct the attention-pooling mechanism like [11], A context vector $u_w \in \mathbb{R}^{1\text{x}d}$ to extract such words that are important to the meaning of the text and aggregate the representation of those informative words to form a text vector.

$$\begin{aligned} u_i &= \tanh(W_s h_t^* + b_s) \\ a_i &= \frac{\exp(u_i^{\mathrm{T}}, u_w)}{\sum_i \exp(u_i^{\mathrm{T}}, u_w)} \\ c &= \sum\nolimits_i a_i u_i \end{aligned} \tag{3}$$

where the h_t^* is the output state of RNNs encoder, the $\mathrm{c} \in \mathbb{R}^{1 \times d}$ is the text vector. It is noteworthy that the context vector $u_w \in \mathbb{R}^{1 \times d}$ can be randomly initialized and jointly learned during the training process.

Multiple Head Attention-Pooling for Text Classification
Vaswani et al. [6] propose multiple head attention, as follows:

$$\begin{aligned} head_i(Q_i, K_i, V_i) &= \text{softmax}(\frac{Q_i K_i^T}{\sqrt{d_k}})V_i \\ \text{MutltiHead}(Q, K, V) &= \text{Concat}(head_1, \ldots, head_h) \end{aligned} \tag{4}$$

In the multi-head attention, the query, key, and value will be separated into parts ($Q_i \in \mathbb{R}^{M \text{x} d_i}, K_i \in \mathbb{R}^{N \text{x} d_i}, V_i \in \mathbb{R}^{N \text{x} d_i}$). Then, model conducts dot product attention for each part ($head_i$) independently. Finally, the multiple head attention vector will be produced by contacting $head_i$.

For text classification, we will formulate the query, key, and value as following:

$$\begin{aligned} Q = W_q u_w, K = W_k h_t^*, V = W_v h_t^* \\ c = MultiHead(Q, K, V) \end{aligned} \tag{5}$$

Where the W_q, W_k, W_v are the learnable weights and $u_w \in \mathbb{R}^{1 \text{x} d}, c \in \mathbb{R}^{1 \text{x} d}$ is the context vector and text vector. In this paper, we employ $h = 8$ parallel attention heads, which means that we will separate them into 8 parts. For each part, $d_i^q = d_i^k = d_i^v = d_{model}/h$.

3.3 Layer Normalization

Layer normalization [17] fixes the mean and the variance of the accumulated inputs within each layer. It computes as:

$$\begin{aligned} norm(h_t) &= \frac{g}{\sigma_t} \otimes (h_t - \mu_t) + b \\ \mu_j = \frac{1}{d}\sum_{i=1}^{d} h_{t,j}, \sigma_t &= \sqrt{\frac{1}{d}\sum_{i=1}^{d}(h_{t,j} - \mu_j)^2} \end{aligned} \tag{6}$$

Where g, b are learned scale and shift parameters with the same dimension as h_t. In this paper, we will apply this structure to each output of sub RNNs layers.

3.4 Feed-Forward Layer

The feed-forward layer can be taken as a 1-D convolution layer, which includes two linear transformations in between with a ReLU activation.

$$ff(h_t, d_o) = dropout(\max(0, Wh_t + b))) \tag{7}$$

Where h_t is the input, and $W \in \mathbb{R}^{d_o \times d_{\text{model}}}$ is the learnable weight. In this paper, we will apply it between each sub RNNs layers.

3.5 Classification Layer

The classification layer includes a linear layer and a softmax layer. The text vector c can be used as features for text classification.

$$p = \text{softmax}(W_c c + b_c) \tag{8}$$

We use the negative log likelihood of the correct labels as training loss.

$$L = -\sum_{d} \log p_{dj,} \tag{9}$$

Where j is the label of the text d.

4 Experiments and Analysis

The following is an extensive empirical analysis of models with different parts of the Transformer block, and how different certain parts affect the performance.

4.1 Experiment Setup

We conduct experiments on corpus with different sizes to get more typical results. For this goal, a small size dataset of NLPCC2014 and dmsc_v2 dataset with large samples are selected. The former small dataset is collected from Chinese product review web site. This task of NLPCC2014 aims to predict the polarity of each review in the data set and the polarity of each review is binary, either positive or negative. We use 10 thousand training sentences and 2500 sentences as test data. Detailed data statistics are shown in Table 1.

Table 1. NLPCC2014 dataset related information

Dataset	Positive	Negative
NLPCC2014-train	5000	5000
NLPCC2014-test	1250	1250

The latter is a big dataset, with more than 2 million ratings from Douban movies. This task of dmsc_v2 aims to analyze the level of each rating and the scores are divided into five levels. We use 1.68 million training examples and 0.42 million examples as test data. Detailed data statistics are shown in Table 2:

Table 2. Dmsc_v2 dataset related information

Dataset	Total_num	Level-1	Level-2	Level-3	Level-4	Level-5
Train	1.68 million	505 K	151 K	142 K	376 K	508 K
Test	0.42 million	126 K	38 K	35 K	93 K	127 K

For both datasets, we process them using the Spacy tokenizer and collect the vocabulary by filtering the word with a frequency less than 5. We also use pre-trained word embedding [18], learning upon Baidu Encyclopedia with SGNS (skip-gram model with negative sampling) [19].

In this paper, the same set of hyperparameters is shared across models. We make a 300 dimensions word embedding layer, and in order to facilitate adding residual connections, all sub-layers in the model produce outputs with 512 dimensions. Adam optimizer with default setting is used for training [20], except that learning rate is 0.0001. Negative Log Likelihood Loss is selected as the loss function, and clip normal in this paper is range from −5 to 5. Each training experiment runs three more times, and then the output is recorded, producing the mean of the Accuracy, Precision, Recall, and F1-score.

4.2 Results and Analysis

Different from the RNNs, the Transformer is more than just self-attention. The differences between RNNs and Transformer also include multi-head attention, multiple attention layers and norm layer. That is why we study the contribution of each part of the Transformer block and try to find which one plays an important role. The details of the models are as follows:

(1) SVM: Traditional SVM is used to classify datasets.
(2) CNN+1 h: Using CNN for text classification, which kernel sizes are the set of [3–5] + attention mechanism.
(3) Transformer: Transformer has the same hyperparameters of transformer block with the follow models.
(4) RNNs + 1 h: Basic model (RNNs + single-attention pooling + classifiaction layer).
(5) RNNs + mh: Replace single-attention pooling with multi-head attention pooling based on (4).
(6) RNNs + mh + norm: Adding a residual connection around each of RNNs sub layers, followed by layer normalization, based on (5).
(7) RNNs + multi-att-1 h/mh + norm: Appling attention pooling (1 h/mh) at each output of RNNs sub layers and contacting multiple attention states together as text vector.
(8) RNNs + multi-att-1 h/mh + norm + ffn: Adding a feed-forward layer based on (7).

Results and Analysis on the NLPCC2014 Dataset
The results of different models on the small dataset are shown in Table 3.

Table 3. Transforming RNNs into a Transformer-style architecture on the small dataset.

Models	Accuracy	F1-Score	Precision	Recall
(1) SVM	0.7316	0.7365	0.7249	0.7539
(2) CNN + 1 h	0.7576	0.7638	0.7447	0.7840
(3) Transformer	0.7584	0.7626	0.7496	0.7760
(4) RNN + 1 h	0.7815	0.8037	0.7312	0.8997
(5) RNN + mh	0.7940	0.8101	0.7524	0.8822
(6) RNN + mh + norm	0.7811	0.8079	0.7202	0.9222
(7) RNN + multi-att-1 h + norm	0.7848	0.8000	0.7502	0.8687
RNN + multi-att-mh + norm	**0.7948**	**0.8249**	0.7196	0.9689
(8) RNN + multi-att-1 h + norm + ffn	0.7856	0.8041	0.7400	0.8807
RNN + multi-att-mh + norm + ffn	0.7892	0.8084	0.7427	0.8959

By comparing (1) with (2, 3, 4), we can find that the basic model is significantly better than the traditional model, which proves that deep networks can reduce artificial feature extraction and make better use of text feature.

By comparing (2, 3) with (4), we know that RNN models are better than CNN or Transformer on this task, which demonstrate that RNNs can better capture the long-range context information.

By comparing (3) with other, we may find that Transformer not always work well on text classification task. Meanwhile, adding each part of Transformer block to RNNs model is an effective way to improve model on text classification work.

By comparing (4) with (5), we can see that the model benefits from the multi-head attention mechanism, which illustrates that Multi-head attention allows the model to jointly attend information from different representation sub spaces at different positions.

By comparing (5) with (6), we can demonstrate that layer normalization on residual inputs blocks has a small drop on evaluation metrics.

By comparing (4) and (5) with (7), we can confirm that the model gains better performance when adding multiple attention layers to both model (4) and model (5) (1 h/mh). It means that multiple attention layers can get more context information.

Compared (7) with (8), the experiments show that adding a feed-forward layer has a slight effect on the models. Adding a feed-forward layer leads to large and consistent performance boost on NMT task [9] while it is not strictly necessary on the text classification task, which confirms that text classification task is different from the NMT task, so we need to make different structures of the model on them.

In the end, we can easily find that the biggest benefits for model on the small dataset come from **multi-head attention** and **multiple attention layers.**

Results and Analysis on the Dmsc_v2 Dataset

The results of different models on the large dataset are shown in Table 4.

Firstly, Most of comparisons are consistent with that on the small dataset, such as (1) with (2, 3, 4) (the traditional model and the neural network), (2, 3) with (4) (the comparison between CNN, Transformer and RNN), (3) with the rest (the comparison between Transformer and enhanced RNNs), (4) with (5) (the effect of adding mh-head attention pooling), and (5) with (6) (the influence of adding layer-norm).

Table 4. Transforming RNNs into a Transformer-style architecture on the large dataset.

Models	Accuracy	F1-Score	Precision	Recall
(1) SVM	0.4986	0.4619	0.4887	0.4594
(2) CNN + 1 h	0.5643	0.5337	0.5547	0.5268
(3) Transformer	0.5437	0.5068	0.5253	0.5030
(4) RNN + 1 h	0.5786	0.5442	0.5701	0.5382
(5) RNN + mh	**0.5845**	0.5526	0.5704	0.5462
(6) RNN + mh + norm	0.5831	0.5455	0.5723	0.5372
(7) RNN + multi-att-1 h + norm	0.5780	0.5447	0.5738	0.5318
RNN + multi-att-mh + norm	0.5837	**0.5584**	0.5776	0.5460
(8) RNN + multi-att-1 h + norm + ffn	0.5795	0.5466	0.5702	0.5355
RNN + multi-att-mh + norm + ffn	0.5839	0.5500	0.5743	0.5397

In the rest of Table 4, by comparing (4), (5) with (7), we can see that adding multiple attention layers has a slight effect on (4) (1 h-attention pooling), but it improves F1-Score on (5) (mh-attention pooling). In addition, adding a feed-forward layer improve slightly performance on both single attention pooling and the multi-head attention pooling. It means that whether the size of the dataset is big or not, adding a feed forward layer has no obvious effect on text classification tasks.

In the end, we can clearly find that the main benefits on the large dataset come from multi-head attention and multiple attention layers like the small dataset.

4.3 Visualization of Different Attention Mechanisms

We visualize the attention weights of examples from test set in Fig. 2. Each part is the same example, which uses both the single-head attention and the multi-head attention. The color shade denotes the weight of words from attention pooling. The higher the weight, the darker the color. For visualization purposes, we normalize the word weight and display the weight higher than threshold 0.1 to make sure that only important words are emphasized.

From Fig. 2, the results show that the single-head attention model can select the words carrying some strong sentiment like “很好 (very good)”, “不 (no)”. However, we can see the words just like the first sentence, whether they are the keywords or not, marked in single-head attention have almost the same color shadow. It shows that single-head attention is not effective all the time. Fortunately, we observe that for multi-head attention model, that is to say, eight attention operations, most of them can find

the keywords in the same example. After integrating this information, it will recognize all keywords in the sentence, which outperform than the single-head attention mechanism. That is the reason why the multi-head attention mechanism has greatly improved the experimental results in the above experiments.

Fig. 2. Visual the same examples with single-head attention and multi-head attention.

5 Conclusion

In this paper, we aim to explore and evaluate how much specific parts of the Transformer block can successfully improve RNNs' performance, especially on the text classification task. Hence, we conduct extensive evaluation experiments on different datasets, such as NLPCC2014 and dmsc_v2 datasets. The scales of there two datasets are different, and the former is a small one, while the other is large.

Based on the experiments and results, we find that RNNs based model in both the small dataset and the large dataset on classification task can benefit from multi-head attention mechanisms and multiple attention layers. In addition, layer normalization on residual inputs blocks drop the performance on both datasets. Moreover, adding a feed-forward layer leads to large and consistent performance boost on NMT task [9] while it is not strictly necessary on the text classification task, which confirms that text classification task is different from the NMT task, so we need to make different structures of the model on them. On the other hand, by comparing the attention visualization results between single-head attention and multi-head attention, we obtain some conclusions which confirm that the multiple head attention mechanism could optimize

attention weights, and improve the performance of models. In a word, the RNNs basic model can greatly improve the performance by continuously adding each part of the Transformer block.

In the future work, considering the strong performance of Transformer block, we will replace the self-attention from Transformer block with more feature extractors on the text classification task, like TextRCNN [21], to further discover the influence of Transformer block.

Acknowledgments. This work is partially supported by a grant from the National Key Research and Development Program of China (No. 2018YFC0832101), the Natural Science Foundation of China (No. 61702080 and 61632011), the Fundamental Research Funds for the Central Universities (No. DUT19RC(4)016), and Postdoctoral Science Foundation of China (2018M631788).

References

1. Elman, J.L.: Finding structure in time. Cogn. Sci. **14**(2), 179–211 (1990)
2. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
3. Cho, K., et al.: Learning Phrase Representations using RNNs encoder–decoder for statistical machine translation. In: EMNLP, pp. 1724–1734. ACL, Doha (2014)
4. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991 (2015)
5. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. In: NAACL-HLT, pp. 260–270. NAACL, San Diego (2016)
6. Vaswani, A., et al.: Attention is all you need. In: NIPS, pp. 5998–6008. Curran Associates, Inc., Long Beach (2017)
7. Tang, G., Müller, M., Rios, A., Sennrich, R.: Why self-attention? A targeted evaluation of neural machine translation architectures. In: EMNLP, pp. 4263–4272. ACL, Brussels (2018)
8. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018). https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/languageunderstandingpaper.pdf
9. Domhan, T.: How much attention do you need? A granular analysis of neural machine translation architectures. In: ACL, vol. 1, pp. 1799–1808. ACL, Melbourne (2018)
10. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: ICLR, San Diego, USA (2015)
11. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: EMNLP. ACL, Lisbon (2015)
12. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: NAACL-HLT, pp. 1480–1489. NAACL, San Diego (2016)
13. Chen, M.X., et al.: The best of both worlds: combining recent advances in neural machine translation. In: ACL. ACL, Melbourne (2018)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR, pp. 770–778. IEEE (2016)
15. Graves, A., Mohamed, A.R., Hinton, G.: Speech recognition with deep recurrent neural networks. In: ICASSP, pp. 6645–6649. IEEE, Vancouver (2013)
16. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS Workshop on Deep Learning (2014)

17. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
18. Li, S., Zhao, Z., Hu, R., Li, W., Liu, T., Du, X.: Analogical reasoning on chinese morphological and semantic relations. In: ACL, vol. 2, pp. 138–143. ACL, Melbourne (2018)
19. Soutner, D., Müller, L.: Continuous distributed representations of words as input of LSTM network language model. In: Sojka, P., Horák, A., Kopeček, I., Pala, K. (eds.) TSD 2014. LNCS (LNAI), vol. 8655, pp. 150–157. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10816-2_19
20. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR. Scottsdale, Arizona, USA (2015)
21. Lai, S., Xu, L., Liu, K., Zhao, J.: Recurrent convolutional neural networks for text classification. In: AAAI, pp. 2267–2273. AAAI, Austin (2015)