Automatic approval of online comments with multiple-encoder networks

Vu Dang[1]

FPT Technology Research Institute, FPT University, Hanoi, Vietnam.
vudh5@fpt.com.vn

Abstract. In modern online publishing, user comments are an integral part of any media platform. Between the high volume of generated comments and the need for moderation of inappropriate content, human approval becomes a serious bottleneck with negative consequences for both operating cost and user experience. To alleviate this problem we present a text classification model for automatic approval of user comments on text articles. With multiple textual input from both the comment in question and the host article, the model uses a neural network with multiple encoders. Different choices for encoder networks and combination methods for encoder outputs are investigated. The system is evaluated on news articles from a leading Vietnamese online media provider, and is currently on a test run with said newspaper.

Keywords: neural network · text classification · online media

1 Introduction

For the vast majority of modern-day media platforms, the social component has become more important than ever. User discourse is now central to drawing and maintaining user interest, sometime even more so than the media content itself. On the other hand the proliferation of fake news and social predation has placed online media under intense scrutiny, and content providers are expected to moderate their own users. As the amount of user-generated content for the typical provider far outstrips the capacity of its editorial staff, human moderation puts great stress on editorial resources while unable to maintain timely approval of user content.

In this work we describe the work done on automatic approval of comments on news articles for VnExpress, a leading online media provider in Vietnam. The task is formulated as a supervised text classification problem using examples of accepted and rejected comments during actual operation of the editorial staff. Since the acceptability of a comment is dependent on the topic and content of the host article, the model input includes such information as the title, summary and category of the article in addition to the comment itself. The model is trained to predict a binary label for each comment and article, corresponding to acceptance or rejection.

The problem of text classification has seen much progress thanks to recent developments in deep learning. Instead of applying generic machine learning models to traditional text features such as n-grams or POS, more and more specialized neural networks have been designed to work directly on words and characters from the input text. Recurrent neural networks, in particular the Long Short-Term Memory (LSTM) [5], have long proved their suitability to sequential data such as text. On the other hand convolutional neural networks (CNNs), initially popularized in computer vision applications by such seminal works as [10, **?**], were later adapted to handle text input with considerable success. [3, **?**] introduced relatively simple CNNs for a variety of natural language processing tasks including text classification. Further works went into specialized nuances of CNNs for text classification; for example [7] explores CNNs with direct convolutions over one-hot word encodings, while [9] proposes a new network structure combining recurrent and convolutional connections. [14] stacked LSTM and convolutional layers for speech recognition, and [17] uses essentially the same architecture for text classification. More recently attention mechanisms have also found success with both image [13, **?**] and text [1]. Self-attention networks [12, **?**] offer relatively simple yet effective models to represent sequential input, while the Transformer model [15] set new standards for neural machine translation using purely attentional elements.

In this work we propose a multi-encoder network for classifying the acceptability of each comment in its context, with an encoder for each text input channel. We consider various design choices for the overall model as well as different encoder architectures, namely the CNN, LSTM-CNN with attention, LSTM with self-attention, and Transformer encoder. Our model is undergoing a trial run at VnExpress with a view to replacing most of the current manual moderation effort. The remainder of this paper is structured as follows: In Section 2 we further elaborate the problem setting and the input data. In Section 3 we describe the proposed model. Section 4 presents a number of variants to the model and Section 5 reports the result of comparative evaluation between them. Finally Section 5 concludes the paper.

2 Problem description

This work aims to automate the moderation process for comments on online news articles. In the current process, user comments can be submitted to the article page at any time but will only be displayed once a human editor has manually approved the content. The moderation follows a set of guidelines where a rejection can be issued for a variety of reasons such as:

- Political incorrectness.
- Vulgarity, pornography or other offensive content.
- Baseless accusation or defamation.
- Advertisement or PR.
- Antisocial content.
- Personal attacks.

As such approval depends on not just the comment itself but also the context. For example a comment accusing a suspect can be approved if the host article has already established that said suspect was found guilty in court, but has to be rejected if the crime is still under investigation. Similarly a comment praising a particular product can be displayed in an article about said product, but will be rejected as advertisement in the context of an unrelated article. Thus we frame the automation task as a classification problem using both the comment and its context as input, where the output label indicates approval or rejection. The classifier is trained on past comments and articles using actual editors' approvals and rejections as ground truth. Its output will be used to decide the approval of a comment if the confidence meets a certain threshold, to be established by the editors after a trial run. On the other hand if the confidence is low, the comment and provisional decision will be passed to an editor to review.

In particular the following input features are available:

- A text string representing the content of the comment.
- A text string representing the title of the host article.
- A text string representing the summary of the host article. This summary is produced by an editor for preview purposes as part of the standard operating procedure.
- A discrete variable representing the top-level category of the host article, for example *politics*, *sport* or *travel*.

Dataset statistics are presented in Table 1. The text inputs are segmented into words using a word tokenizerl. This step is necessary since modern Vietnamese language is written as a sequence of syllables with no specific notation for word boundaries.

Table 1. Dataset statistics

Accepted comments	274154
Rejected comments	186474
Unique articles	16087
Unique categories	16
Average comment length (words)	30.6
Average title length (words)	12.6
Average summary length (words)	22.1

3 Proposed model

3.1 Model structure

The general structure of our proposed model is illustrated in Figure 1, with one encoder network for each text input channel and a dense layer with softmax on

top. The encoder outputs are concatenated, then element-wise multiplication is performed with a category-specific weight vector before going to the dense layer.

Let $\mathbf{x}$ be the word sequence representing the comment itself, and $\mathbf{t}$ and $\mathbf{s}$ represent the title and summary of the host article respectively. Let F_x, F_t and F_s be the corresponding encoder functions with the same output dimension d, and denote by $\oplus$ the concatenation operator. Then the combined encoder output is a $3d$-dimensional vector

$$E(\mathbf{x}, \mathbf{t}, \mathbf{s}) = F_x(\mathbf{x}) \oplus F_t(\mathbf{t}) \oplus F_s(\mathbf{s})$$

Let c be the category of the host article and e_c be an $3d$-dimensional vector corresponding to c (which can be regarded as an embedding from the set of categories to $\mathbb{R}^{3d}$), and denote by $\circ$ the Hadamard or element-wise product. Furthermore let W and b be the weight and bias of the final dense layer, then the final model output is

$$\text{softmax}(W(E(\mathbf{x}, \mathbf{t}, \mathbf{s}) \circ e_c) + b)$$

The most natural approach to incorporate category information would be to append an embedding vector or a one-hot vector to the combined encoder output. However this approach only allows the category information to influence the next layer additively, while we have observed that the nature of the moderation process is highly dependent on the category of the host article. For example comments on politics are generally rejected for completely different reasons than comments on lifestyle articles. Thus we opted to use a Hadamard product of embeddings, which allows the input category to have direct influence over all features coming into the last classifier layer.

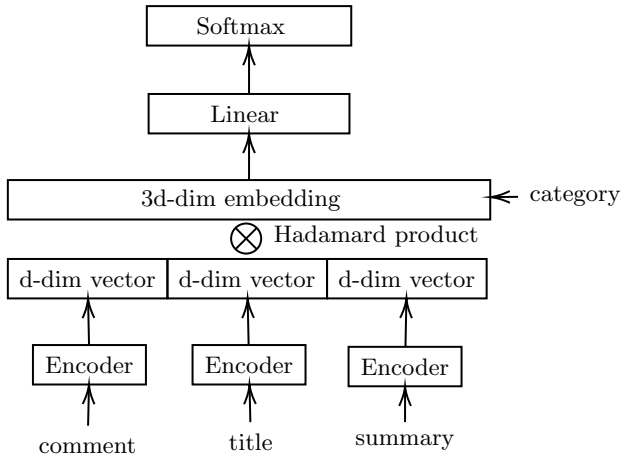

Fig. 1. Proposed classification model for comment approval

3.2 Self-attention encoder

For the encoders we use the self-attention network in [12]. The network consists of a bi-directional LSTM layer followed by a self-attention mechanism. Let $H = (h_1 \dots h_n)$ be the hidden states of the LSTM layer, then the embedding M of the whole input sequence is a linear combination of these states, namely $M = AH$. The combination weights are computed as in Figure 2 (taken from [12]). In short the weight matrix A is the output of a two-layer perceptron without bias (using ReLU activation instead of tanh as in [12]), with a softmax applied along the second dimension (i.e. along the sequence). The rows of A act like different filters which allow the mechanism to attend to multiple parts of the input sequence, and the embedding M is a $r \times 2u$ matrix where r is the number of filters and $2u$ is the output dimension of the bi-directional LSTM. Finally we use a fully connected layer to reduce M to a d-dimensional embedding vector. For regularization we apply dropout to the input and output layers.

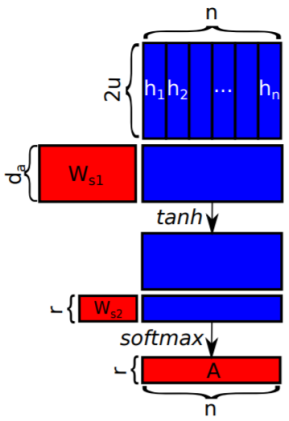

Fig. 2. Structure of the self-attention weights

4 Model variants

In this section we present various options to the model and evaluate them on the collected dataset.

4.1 Baselines

As baseline we evaluate a simple application of logistic regression, SVM and gradient boosted trees. Each input text channel is represented by a bag-of-words from unigrams, bigrams and trigrams, with one-hot encoding for article category.

These representations are concatenated into a long vector to feed into the classifier. The results for these classifiers are compared to the results of our proposed model with different encoders in Section 5.

4.2 Encoders

Here we list the different networks which were used as encoders in our model:

- **CNN**: We implement a simple convolutional network in the style of [8]. The network consists of multiple convolutional layers applied in parallel to the model input. The convolutional outputs go through adaptive pooling before concatenation, with a dense layer on top.
- **LSTM-CNN with Attention**: The first stage of this network is a bi-directional LSTM layer with a soft attention mechanism. The second stage has the same structure as the aforementioned CNN encoder and acts on the output of the first stage.
- **Self-attention**: This is the proposed encoder as detailed in Section 3.
- **Transformer**: The Transformer network is an influential architecture using purely attentional elements, first proposed in [15]. The core of this network is the Multi-Head Attention mechanism (Figure 3), with added positional embeddings to account for the sequential ordering of input tokens. Our encoder consists of a single Transformer layer with dropout, where residual input is added to the output of the attention mechanism before layer norm is applied. In our preliminary experiment this is much faster to train and less prone to overfitting than stacking multiple Transformer layers as in [15], which makes sense as our model contains three encoders and only has to predict a binary label. The output sequence is averaged along the sequential dimension to produce the final encoder output.
- **ConvS2S**: This encoder is derived from ConvS2S [4], a purely convolutional network for machine translation. Each convolutional block consists of a convolution with Gated Linear Units and a residual connection (Figure 4). The output sequence is averaged along the sequential dimension to produce the final encoder output. In a similar situation to the Transformer encoder, we found that using a single block for each encoder resulted in faster training and less overfitting than stacking multiple blocks per encoder.

4.3 Model configurations

We experimented with various ways to put the encoder outputs together with the category information as below:

- **Input concatenation**: As a baseline we simple concatenate all input channels (title, summary and comment) with the one-hot representation of the category to form a single sequence. The model is reduced to a simple classifier consisting of a single encoder which feeds into a dense layer with softmax.

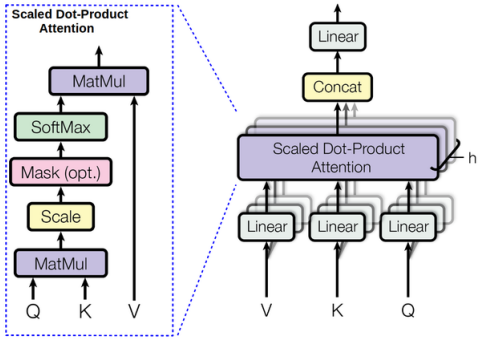

Fig. 3. Multi-head attention from the Transformer architecture

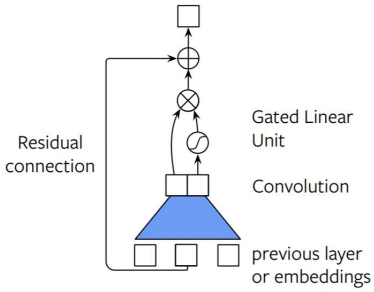

Fig. 4. Convolutional block from the ConvS2S architecture

- **Feature concatenation**: We concatenate the encoder outputs and the one-hot representation of the category into a single sequence before the dense layer with softmax.
- **Shared encoder**: This is the same configuration as the proposed model, except that all the encoders share the same weights.
- **Feature multiplication**: In stead of concatenating the encoder outputs, we take the Hadamard product of all the encoder outputs and an embedding of the category (all being of the same dimensionality). This product is the input to the final dense layer with softmax.

4.4 Pretrained embeddings

We evaluated our model with both randomly initialized word embeddings and pretrained embeddings. The pretrained embeddings were obtained from [16], in particular we used the 300-dimensional fastText [2] word vectors. When using the pretrained embeddings we initialize all pretrained words to their corresponding embedding vectors and the remaining words to random embeddings. Then we experimented with two options:

- Fixing the embedding vectors of pretrained words to their initial pretrained values and only fine-tuning the randomly initialized embeddings.
- Fine-tuning all embedding vectors during training.

5 Evaluation

The model variants are evaluated by 5-fold cross validation, using 10% of the training set for validation and hyperparameters are chosen by a simple grid search. The best configuration for our proposed models uses independent self-attention encoders for each input text channel, with a hidden dimension of 200 and dropout rate of 0.2. The best result comes with finetuning 300-dimensional pretrained word embeddings; however with randomly initialized word embeddings we found empirically that reducing the embedding dimension to 100 achieves the best result. We report macro-F1 score over all five folds in percentage points.

Table 2. Evaluation of different encoders and baselines

	F1 score
Logistic regression	73.07
SVM	72.42
Gradient boosted trees	68.59
CNN	76.58
LSTM-CNN with Attention	78.28
Self-attention	**78.34**
Transformer	77.93
ConvS2S	77.12

Table 2 shows the F1 score for different encoders. The best results by a considerable margin belong to the Self-attention and LSTM-CNN with Attention, and thus we perform further evaluations with these two encoders. The results are reported in Figure 5. All model variants outperform the baseline with input concatenation by a large margin, with the proposed version clearly achieving the best results for both encoder options.

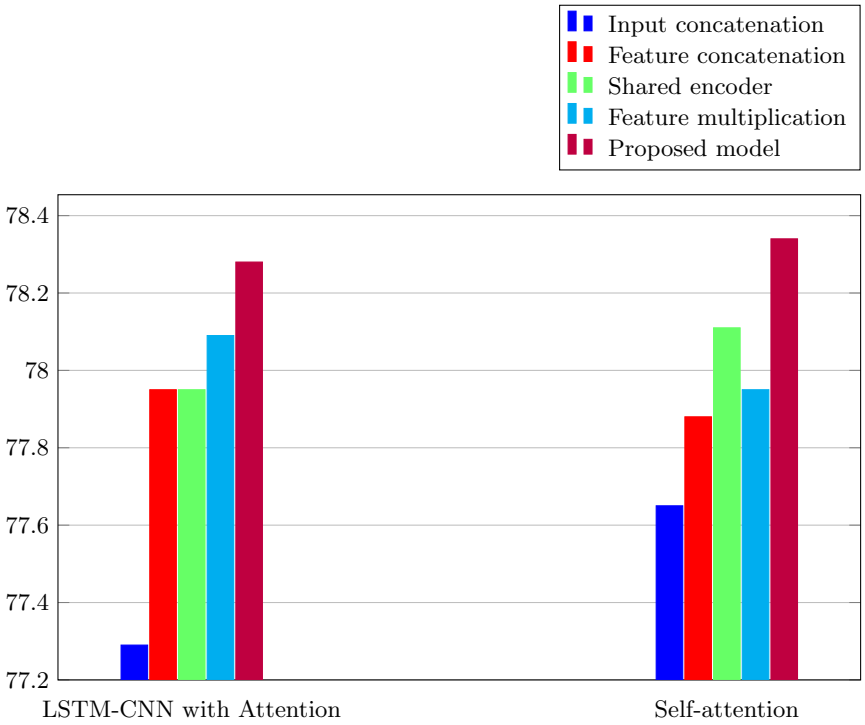

Fig. 5. Evaluation of model configurations

Table 3 reports our experiment with pretrained embeddings for the two best encoders. Fixed embeddings fall behind by a large margin with both encoders, while for the LSTM-CNN with Attention the random embeddings actually outperformed fine-tuned pretrained embeddings. This is because the optimal embedding dimension found in our experiments for this encoder is 200 while the embedding vectors are 300-dimensional, and raising the dropout rate could not compensate for the increased model capacity. On the other hand the Self-attention encoder achieves the best result by fine-tuning pretrained embeddings.

Table 3. Evaluation of pretrained embeddings

Configuration	Embedding	F1 score
Self-attention	Random	78.34
Self-attention	Fixed	75.00
Self-attention	Fine-tuned	**78.42**
LSTM-CNN with Attention	Random	78.28
LSTM-CNN with Attention	Fixed	75.16
LSTM-CNN with Attention	Fine-tuned	77.85

5.1 Performance considerations

In practical use we are mainly interested in the inference time since training can be scheduled while the system has no control over when new comments arrive. Figure 6 show the average inference time over 100 batches for different batch sizes from 1 to 1000, run on a single Tesla V100-SXM2 GPU. While the average batch time for the Self-attention encoder stays essentially constant, the LSTM-CNN with Attention encoder slows down considerably with larger batches as it uses up the GPU memory.

In principle this means the Self-attention encoder scales much better with the volume of incoming comments. However in practice it requires thoughtful implementation to make use of this large-batch advantage: the system cannot just wait for a full batch before inferencing since the wait time might be too long, but running each single comment through the model as soon as they come in would be highly inefficient. Currently we implement a fixed waiting time where comments which arrive within this time will be processed as a batch. The waiting time is chosen empirically; in fact setting it close to 0.1 second makes sure that the system has time to finish a batch before the next one arrives, while the impact on user experience is negligible. In the rare case that there are too many incoming comments during the waiting time, we simply create extra model instances on other worker machines to distribute the load.

6 Conclusion

In this paper we formulated the task of automatic moderation of online comments to news articles as a classification problem. We proposed a neural network model for the task and explored various design choices for our model including different encoders, configurations and the use of pretrained word embeddings. Evaluation on operational data shows the consistent superiority of the LSTM with self-attention as the encoder for our model and the importance of article category. The best-performing model variant is currently being trialed by the online newspaper VnExpress, to be optimized for deployment. We also discussed some practical consideration regarding batching, which arose during the trial run.

Since the proposed model uses one encoder per input text channel, it can be easily extended to richer formulations of the task, for example by adding related

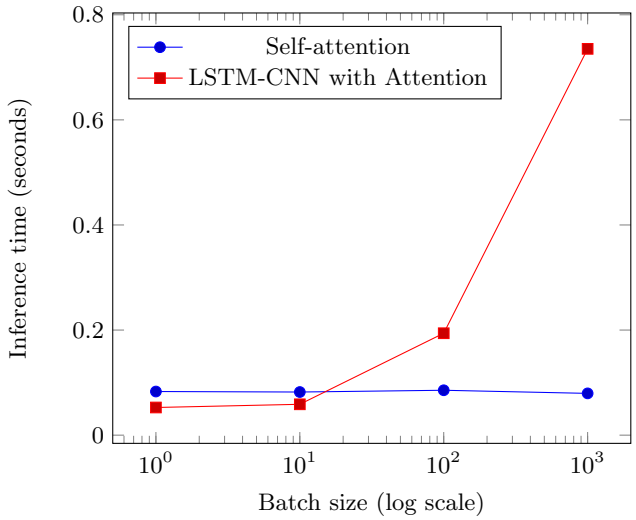

Fig. 6. Inference time by batch size

articles or other comments in the same conversation. Given the multimodal nature of digital media, the natural next step in our work is to incorporate images and videos where applicable and we are investigating specific choices of encoders for each mode of input. A more challenging problem is the need to justify the decisions taken by the automatic system, which will speed up the process of manual review considerably in case the model's prediction does not meet the confidence threshold. Attention mechanisms have shown some potential in explaining neural networks even though their practical usefulness in this regard is being debated (for example see [6]). In addition certain neural network architectures have been specifically designed to provide evidence for their predictions such as [11, **?**] as part of the growing interest in explainable machine learning. These advances provide promising directions for a deeper approach to the problem of automatic content moderation.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv e-prints **abs/1409.0473** (Sep 2014), https://arxiv.org/abs/1409.0473
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
3. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**, 2493–2537 (Nov 2011), http://dl.acm.org/citation.cfm?id=1953048.2078186

4. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. pp. 1243–1252. ICML'17, JMLR.org (2017), http://dl.acm.org/citation.cfm?id=3305381.3305510
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (Nov 1997). https://doi.org/10.1162/neco.1997.9.8.1735, http://dx.doi.org/10.1162/neco.1997.9.8.1735
6. Jain, S., Wallace, B.C.: Attention is not explanation. CoRR **abs/1902.10186** (2019)
7. Johnson, R., Zhang, T.: Effective use of word order for text categorization with convolutional neural networks. In: NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015. pp. 103–112 (2015), http://aclweb.org/anthology/N/N15/N15-1011.pdf
8. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL. pp. 1746–1751 (2014), http://aclweb.org/anthology/D/D14/D14-1181.pdf
9. Lai, S., Xu, L., Liu, K., Zhao, J.: Recurrent convolutional neural networks for text classification. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. pp. 2267–2273. AAAI'15, AAAI Press (2015), http://dl.acm.org/citation.cfm?id=2886521.2886636
10. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE. pp. 2278–2324 (1998)
11. Lei, T., Barzilay, R., Jaakkola, T.: Rationalizing neural predictions. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 107–117. Association for Computational Linguistics, Austin, Texas (Nov 2016). https://doi.org/10.18653/v1/D16-1011, https://www.aclweb.org/anthology/D16-1011
12. Lin, Z., Feng, M., dos Santos, C.N., Yu, M., Xiang, B., Zhou, B., Bengio, Y.: A structured self-attentive sentence embedding (2017)
13. Mnih, V., Heess, N., Graves, A., kavukcuoglu, k.: Recurrent models of visual attention. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27, pp. 2204–2212. Curran Associates, Inc. (2014), http://papers.nips.cc/paper/5542-recurrent-models-of-visual-attention.pdf
14. Sainath, T.N., Vinyals, O., Senior, A., Sak, H.: Convolutional, long short-term memory, fully connected deep neural networks. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 4580–4584 (April 2015). https://doi.org/10.1109/ICASSP.2015.7178838
15. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 5998–6008. Curran Associates, Inc. (2017), http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf
16. Vu, X., Vu, T., Tran, S.N., Jiang, L.: ETNLP: A toolkit for extraction, evaluation and visualization of pre-trained word embeddings. CoRR **abs/1903.04433** (2019)
17. Zhou, C., Sun, C., Liu, Z., Lau, F.C.M.: A c-lstm neural network for text classification. CoRR **abs/1511.08630** (2015)