Feature-guided Neural Model Training for Supervised Document Representation Learning

Aili Shen Bahar Salehi Jianzhong Qi Timothy Baldwin
School of Computing and Information Systems
The University of Melbourne
Victoria, Australia
ailis@student.unimelb.edu.au baharsalehi@gmail.com
jianzhong.qi@unimelb.edu.au tb@ldwin.net

Abstract

With the advent of neural models, there has been a rapid move away from feature engineering, or at best, simplistically combining hand-crafted features with learned representations as side information. We propose a method that uses hand-crafted features to guide learning by explicitly attending to feature indicators when learning the relationship between the input and target variables. In experiments over two different tasks — quality assessment of Wikipedia articles and popularity prediction of online petitions— we demonstrate that the proposed method yields neural models that consistently outperform those that simply use hand-crafted features as side information.

1 Introduction and Background

Text classification/regression is a fundamental problem in natural language processing. Traditional methods make use of hand-crafted features, such as the length of a document, to represent a document. A classifier/regressor is built on top of such features to learn a model (Wang and Manning, 2012; Warncke-Wang et al., 2013, 2015; Dang and Ignat, 2016). Recently, neural models such as LSTMs (Hochreiter and Schmidhuber, 1997) and convolutional neural networks (CNNs: Kim (2014); Kalchbrenner et al. (2014)) have become the de facto for text classification/regression tasks, with one oft-cited advantage being that they are able to learn implicit features as part of the representation learning.

Studies employing neural models either eschew hand-crafted features or simplistically use hand-crafted features as side information. For example, Dang and Ignat (2017) propose to use a bidirectional LSTM ("bi-LSTM") to classify Wikipedia articles by their quality classes, and Shen et al. (2017) concatenate structural features (e.g., article length) and readability scores with bi-LSTM-learned document representations for the same task. Subramanian et al. (2018) hand-engineer a set of features (e.g., the ratio of indefinite and definite articles), and concatenate them with CNN-learned document representations to predict the popularity of online petitions. Wu et al. (2018) explore the utility of hand-crafted features in NER by concatenating these features with character representations learned via an CNN and word embeddings. These representations are then fed into a bi-LSTM to identify named entities (with the help of a CRF) and re-construct the hand-crafted features in the output simultaneously, which is achieved by combining an auto-encoder loss with the NER loss.

The motivation underlying this work is that when hand-crafted features are represented by numerical vectors and concatenated with neural network representations, there is no information on what kind of feature each value represents. To make better use of hand-crafted features, we propose a feature-guided neural training method that guides the network to map feature indicators onto (explicit or implicit) features in the document. We evaluate the effectiveness of the proposed method over two datasets for two different tasks: (1) quality assessment of Wikipedia articles, and (2) popularity prediction of online petitions. Taking state-of-the-art approaches for the respective tasks, we achieve consistent improvements when using our model.

The closest work to our approach is the label-guided model training of Wang et al. (2018). They embed words and labels in the same embedding space, and compute a label-based attention score between a word and all possible labels, which is used to weight word embeddings in obtaining document representations. Our work differs in two aspects: (1) Wang et al. (2018) capture direct associations between labels and words, while we use the proxy of (potentially much higher-level) hand-crafted features to guide network learning; and (2) our method does not rely on the target variable being closely related to the semantics of a document, leading to better generalisability.

2 Methodology

Figure 1 is an illustration of our proposed approach, in the context of a stacked bi-LSTM, where two bi-LSTMs are applied to obtain the sentence and document level representations, respectively. Note that our method is not limited to LSTMs, as we show in Section 3.

A hand-crafted feature can consist of multiple feature indicators. For example, having *level 3+ headings* is one of the features in quality assessment of Wikipedia articles. This hand-crafted feature consists of two feature indicators (tokens) {"===", "===="}. We embed the document and feature indicators into a shared space. Then, as indicated in Figure 1, for each feature indicator, we compute cosine similarity between the feature indicator and word embeddings, followed by average-pooling to obtain a sentence score:

$$\text{score} = \frac{1}{N}\sum_{j=1}^{N}\frac{\mathbf{F}\mathbf{V}_j}{\|\mathbf{F}\|_2\|\mathbf{V}_j\|_2}. \quad (1)$$

Here, $\mathbf{F}$ and $\mathbf{V}_j$ are embeddings of the feature indicator and the jth word in a sentence, respectively; $\|\mathbf{F}\|_2$ and $\|\mathbf{V}_j\|_2$ are the ℓ_2 norms of $\mathbf{F}$ and $\mathbf{V}_j$, respectively; and N is the number of words in a sentence. All scores based on the feature indicator are concatenated with sentence representations $\boldsymbol{Z}_1$, which are learned through a bi-LSTM layer (f_1). Then, another bi-LSTM layer (f_2) is applied to the concatenated sentence representations to obtain document representation $\boldsymbol{Z}_2$, which is followed by a dense layer (f_3) to compute $\boldsymbol{y}$.

The score computed in Equation 1 is for a single-token feature indicator. If a hand-crafted feature consists of multiple feature indicators (tokens, e.g., {"===", "===="}), the score becomes:

$$\text{score} = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{N}\sum_{j=1}^{N}\frac{\mathbf{F}_i\mathbf{V}_j}{\|\mathbf{F}_i\|_2\|\mathbf{V}_j\|_2}. \quad (2)$$

Here, M is the number of feature indicators in a hand-crafted feature, and $\mathbf{F}_i$ is the word embedding of the ith feature indicator in the feature.

For example, the feature *level 3+ headings* is one of structural features described in Shen et al. (2017), which consists of two feature indicators {"===", "===="}. To obtain the similarity score for the feature *level 3+ headings*, we first compute the similarity score between each feature indicator "==="/"====" and word embeddings in the sentence, then apply average-pooling to obtain the similarity score for each feature indicator. We obtain the similarity score of *level 3+ headings* by averaging similarity scores among the feature indicators at the sentence level. Finally, the feature score is concatenated with the sentence representation, which is fed into a latter layer.

While we don't experiment with this in this paper, it is also possible to first average feature indicator embeddings and then compute the sentence score by Equation 1. This way, we can efficiently reduce the computation of similarity scores for hand-crafted features with a large number of feature indicators. In this paper, we use Equation 2, as the maximum number of feature indicators in a given hand-crated feature is less than $1,000$, and less than 10 in most cases.

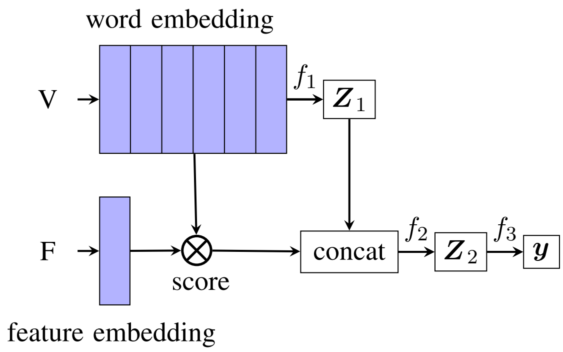

Figure 1: Illustration of our proposed method. Here, $\otimes$ denotes cosine similarity between the feature indicator and word embeddings; f_1 and f_2 denote bi-LSTM layers; f_3 denotes a dense layer; $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ denote sentence and document representations, respectively; $\boldsymbol{V}$ and $\boldsymbol{F}$ are the document input and feature indicators, respectively; and $\boldsymbol{y}$ is the target output.

3 Experiments

To test the effectiveness of our proposed method, we experiment with a Wikipedia document quality assessment task (Shen et al., 2019), and online petition signature prediction task (Subramanian et al., 2018), as detailed below. The reasons we chose these particular tasks are as follows. First, extensive domain-specific feature engineering had taken place in each case, that we could use as the basis of our feature indicators. Second, strong neural benchmarks have been established, based on extensive experimentation with both neural and non-neural models. Our experiments in this paper are based on the state-of-the-art.

We aim to explore the relative gains of our proposed method relative to the current state-of-the-art for the task, which in both cases is not based on contextualised embeddings. For BERT (Devlin et al., 2019) or other contextualised encoders, the same word in different contexts will end with different embeddings, leading to localized representations of feature indicators. As such, the proposed method is not directly applicable to models such as BERT, and novel research would be required to adapt the method to such models.

3.1 Wikipedia Document Quality Assessment

Dataset The Wikipedia dataset (Shen et al., 2019) consists of 29,794 English Wikipedia articles and their corresponding quality labels: Featured Article, Good Article, B-class Article, C-class Article, Start Article, and Stub Article, in descending order of document quality. The dataset is class-balanced and partitioned into training, development, and test splits (8:1:1). Documents are relatively long, and processed in a hierarchical manner, by constructing sentence representations, and composing these into a document representation.

Following Dang and Ignat (2017) and Shen et al. (2019), we formulate the quality assessment of Wikipedia articles as a multi-class classification problem, and all models are trained to minimise cross-entropy loss. We report average `accuracy` and standard deviation over 10 runs.

Hand-crafted features used to guide network learning here include: (1) references indicators; (2) links to other Wikipedia pages indicators; (3) citation templates indicators; (4) non-citation templates indicators; (5) categories linked in the article indicators; (6) image indicators; (7) infobox indicators; (8) level 2 headings indicators; and (9) level 3+ heading indicators. These features are from Dang and Ignat (2016) and Shen et al. (2017). Hand-crafted features in `Side-information` are based on counting the number of appearances of such feature indicators.

Model configuration We apply our proposed approach ("`Feature-guided`") over the four models detailed below. In each case, we contrast with two baselines: (1) `Vanilla`, makes no use of hand-crafted features; and (2) `Side-information` which uses the hand-crafted features as side information, by concatenating them with learned representations in the penultimate layer.

1. CNN_BILSTM: apply convolution kernels with width 2, 3, and 4 (32 for each width size) to word embeddings within a sentence, and a tanh activation function to each; pass the output of the filters through a bi-LSTM.

2. AVERAGE_BILSTM (Shen et al., 2017): average word embeddings to get the sentence representation, and run a bi-LSTM over the sequence of sentence representations.

3. STACKED_BILSTM: feed the word embeddings in a sentence through a bi-LSTM, and the output through a max-pooling layer; finally, apply another bi-LSTM over the sentence representations.

4. STACKED_BILSTM_ATT (Yang et al., 2016): use a hierarchical STACKED_BILSTM, except that an attention mechanism with a context size of 100 is applied to the output of each bi-LSTM to weight words/sentences based on their importance in the sentence/document.

A max-pooling layer is applied to the output of the bi-LSTM at the sentence level for all models except STACKED_BILSTM_ATT to get the document level representation, which is followed by two dense layers, one with a ReLU activation and one without any activation function. For all models, dropout layers are applied at both the sentence and document levels with a rate of 0.5 during training. For both CNN_BILSTM and AVERAGE_BILSTM, the bi-LSTM cell size is set to 256. For STACKED_BILSTM and STACKED_BILSTM_ATT, the cell size is set to 32 and 256 for the sentence and document level bi-LSTM, respectively.

We use 50-dimensional pre-trained word embeddings from GloVe (Pennington et al., 2014).[1] For OOV words, the word embeddings are randomly initialised based on sampling from a uniform distribution $U(-1, 1)$. All word embeddings

[1]We fine-tuned hyper-parameters over the development set for quality predictions of Wikipedia articles. 50-dimensional embeddings were chosen because `Vanilla` performs the best under this setting (meaning the baseline without features is as strong as possible).

Model	CNN_BILSTM	AVERAGE_BILSTM	STACKED_BILSTM	STACKED_BILSTM_ATT
`Vanilla`	57.12±0.58%	57.91±0.81%	57.60±0.65%	56.70±1.21%
`Side-information`	57.24±0.47%	59.04±0.33%	57.97±0.74%	57.44±0.62%
`Feature-guided`	**58.10±0.50%**†	**59.90±0.45%**†	**58.30±0.71%**	**58.30±0.65%**†

Table 1: `accuracy` over Wikipedia dataset. The best result is in **bold**, and marked with "†" if the improvement is statistically significant (based on a one-tailed Wilcoxon signed-rank test; $p < 0.05$).

are updated in the training process. We use a mini-batch size of 128 and a learning rate of 0.01. We train each model for 50 epochs. To prevent over-fitting, early stopping is adopted. All hyper-parameters are set empirically over the development data, and the models are optimised using Adam (Kingma and Ba, 2015).

Results The experimental results are presented in Table 1. We can see that our method outperforms both `Vanilla` and `Side-information` across all four network architectures, at a level of statistical significance for 3 out of the 4 models. It is worth noting that the performance of STACKED_BILSTM_ATT is worse than that of STACKED_BILSTM for both `Vanilla` and `Side-information`, due to attention in STACKED_BILSTM_ATT not being as effective as max-pooling on this task. However, the performance of STACKED_BILSTM_ATT is not worsened by incorporating the attention for our method, indicating that our feature-guided learning can guide the network to learn better.

3.2 Online Petition Signature Prediction

Dataset The online petitions dataset (Subramanian et al., 2018) consists of 10,950 UK petitions and their corresponding signature counts. Following Subramanian et al. (2018), we chronologically split the data into training, dev, and test (8:1:1).

We formulate signature count prediction as a regression problem, and all models are trained to minimise mean squared error. For evaluation, average mean absolute error ("`MAE`") and mean absolute percentage error ("`MAPE`") over 10 runs are reported. Here, `MAPE` is calculated as $\frac{100}{n}\sum_1^n \frac{|\hat{y}_i - y_i|}{y_i}$, where $\hat{y}_i$ and y_i are the predicted and ground-truth signature counts.

Hand-crafted features used to guide network learning here include: (1) indefinite vs. definite articles; (2) P1 singular and plural, P2, and P3 singular and plural pronouns; (3) subjective, positive, and negative words; and (4) biased words. These features are from Subramanian et al. (2018).

Model	`MAE`	`MAPE`
`Vanilla`	1.44	38.1
`Side-information`	1.45	39.0
`Feature-guided`	**1.42**†	**36.2**†

Table 2: Results over online petitions. The best result is indicated in **bold**, and marked with "†" if the improvement is statistically significant (based on a one-tailed Wilcoxon signed-rank test; $p < 0.05$).

Hand-crafted features in `Side-information` are based on counting the number of appearances of such feature words.

Model configuration We again compare our proposed `Feature-guided` approach with `Side-information` and `Vanilla`, in the form of a state-of-the-art CNN with convolution kernels of width 1, 2, and 3 (100 kernels for each width size) over word embeddings, with a ReLU applied to each. The outputs are passed through two dense layers, one with a tanh activation function and one with an ELU activation function, to obtain the final output.

A dropout layer is applied to the output of the convolution filters at a rate of 0.5 during training. We use a mini-batch size of 32, and a learning rate of $1e-4$. All other hyper-parameters are the same as in the Wikipedia setting, except that the early stopping is based on `MAE`.

Results Table 2 summarises our results. We observe that our approach benefits `Vanilla` and `Side-information` once again, at a level of significance in terms of both `MAE` and `MAPE`. It is worth noting that `Side-information` performs worse than `Vanilla`, as it over-fits to features only present in the training data. In comparison, our method improves the model performance even in this case, as it identifies words semantically related to the feature indicators. For example, the word *increase*, not in the predefined list of positive words, has a high similarity score (> 0.8) with positive words *help*, *hope*, *give*, and *allow*.

4 Conclusion and Future Work

We proposed a method to guide network learning by attending to feature indicators associated with hand-crafted features. Experimental results over two tasks (quality assessment of Wikipedia articles, and popularity prediction of online petitions) show that our approach consistently outperforms two baselines, across a range of neural architectures. For future work, we are interested in exploring hand-crafted features from external sources, such as editor comments of a Wikipedia article.

References

Quang-Vinh Dang and Claudia-Lavinia Ignat. 2016. Measuring quality of collaboratively edited documents: the case of Wikipedia. In *Proceedings of the 2nd IEEE International Conference on Collaboration and Internet Computing*, pages 266–275.

Quang Vinh Dang and Claudia-Lavinia Ignat. 2017. An end-to-end learning solution for assessing the quality of Wikipedia articles. In *Proceedings of the 13th International Symposium on Open Collaboration*, pages 4:1–4:10.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT 2019)*, pages 4171–4186.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *The 3rd International Conference on Learning Representations, (ICLR 2015)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 1532–1543.

Aili Shen, Jianzhong Qi, and Timothy Baldwin. 2017. A hybrid model for quality assessment of Wikipedia articles. In *Proceedings of the Australasian Language Technology Association Workshop*, pages 43–52.

Aili Shen, Bahar Salehi, Timothy Baldwin, and Jianzhong Qi. 2019. A joint model for multimodal document quality assessment. In *Proceedings of the 19th ACM/IEEE-CS on Joint Conference on Digital Libraries (JCDL 2019)*, pages 107–110.

Shivashankar Subramanian, Timothy Baldwin, and Trevor Cohn. 2018. Content-based popularity prediction of online petitions using a deep regression model. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, (ACL 2018)*, pages 182–188.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 2321–2331.

Sida I. Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 90–94.

Morten Warncke-Wang, Vladislav R. Ayukaev, Brent Hecht, and Loren Terveen. 2015. The success and failure of quality improvement projects in peer production communities. In *Proceedings of the 18th ACM Conference on Computer-Supported Cooperative Work and Social Computing*, pages 743–756.

Morten Warncke-Wang, Dan Cosley, and John Riedl. 2013. Tell me more: An actionable quality model for Wikipedia. In *Proceedings of the 9th International Symposium on Open Collaboration*, pages 8:1–8:10.

Minghao Wu, Fei Liu, and Trevor Cohn. 2018. Evaluating the utility of hand-crafted features in sequence labelling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, page 28502856.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NACCL-HLT 2016)*, pages 1480–1489.