Bidirectional Transformer Based Multi-Task Learning for Natural Language Understanding

Suraj Tripathi(✉), Chirag Singh(✉), Abhay Kumar(✉), Chandan Pandey(✉), and Nishant Jain(✉)

Samsung R&D Institute, Bengaluru, India
{suraj.tri, c.singh, abhayl.kumar, chandan.p, nishant.jain}@samsung.com

Abstract. We propose a multi-task learning based framework for natural language understanding tasks like sentiment and topic classification. We make use of bidirectional transformer based architecture to generate encoded representations from given input followed by task-specific layers for classification. Multi-Task learning (MTL) based framework make use of a different set of tasks in parallel, as a kind of additional regularization, to improve the generalizability of the trained model over individual tasks. We introduced a task-specific auxiliary problem using the k-means clustering algorithm to be trained in parallel with main tasks to reduce the model's generalization error on the main task. POS-tagging was also used as one of the auxiliary tasks. We also trained multiple benchmark classification datasets in parallel to improve the effectiveness of our bidirectional transformer based network across all the datasets. Our proposed MTL based transformer network improved state-of-the-art overall accuracy of Movie Review (MR), AG News, and Stanford Sentiment Treebank (SST-2) corpus by 6%, 1.4%, and 3.3% respectively.

Keywords: Bidirectional transformer · Sentiment classification · Multi-task learning · Unsupervised learning

1 Introduction

The learning of representation is the cornerstone of every task of machine learning. Deep learning became widely popular due to the very effective learning of representation through error backpropagation. The main issue with deep learning based methods is that they require a large amount of labeled data to generalize well on unseen data but in many Natural Language Processing (NLP) tasks, labeled data is scarce, so usually, pre-training for a language model on unsupervised data is used for learning universal language representations and transfer learning [1, 2].

Another widely used approach for feature learning is Multi-Task learning [3]. The learning behavior of humans also inspires MTL since we are capable of capturing general idea across tasks and easily transfer knowledge acquired from one task to

A. Kumar, C. Pandey and N. Jain—Equal Contribution.

E. Métais et al. (Eds.): NLDB 2019, LNCS 11608, pp. 54–65, 2019.
https://doi.org/10.1007/978-3-030-23281-8_5

another task. This is because human learning generalizes well and is not focused on learning specific patterns of a task too well. It is useful for multiple tasks to be jointly trained so that the features learned in one task can benefit other related tasks. Recently, there is a lot of interest in applying MTL for representation learning especially using deep neural networks [4, 5]. MTL helps by augmenting the dataset of all the tasks involved since we are training multiple tasks in parallel. It also helps in reducing the generalization error by preventing overfitting to a specific task. In an MTL setting, it is crucial to select the relevant and related task, but to encourage cross-task learning, diversity is also essential. We contend that if we combine the language pre-training with MTL, both can help to learn even better representation for general Natural Language Understanding (NLU) tasks. Following that, we decided to demonstrate the effectiveness of our proposed networks on the tasks of Sentiment analysis and Topic modeling, which are among the widely used tasks of NLP.

2 Related Work

Existing systems on Sentiment analysis are mostly deep learning based or some other supervised learning approaches like Support Vector Machines (SVMs) over a carefully constructed feature set. In spite of being one of the most explored tasks, it is still very challenging due to the inherent ambiguity of natural language and the complexity of human emotions. Work on Sentiment analysis can be broadly classified into Traditional approaches and Deep learning based methods.

Traditional approaches [6, 7] are based on engineering features like Bag of words model or a combination of words and their sentiment strength scores. These sentiment scores are assigned to words by an algorithm and some manual engineering as well. As expected these methods are cumbersome and error-prone. Manually covering all relevant features is very difficult in practice. Also, a slight modification in dataset or problem definition requires repetition of the whole process all over again.

Feature learning through back-propagating errors is one of the key strengths of deep neural networks. Emergence of Deep networks like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) and word embedding methods like Word2Vec, GloVe, etc. marked the new era of machine learning techniques where there is no need for manual feature engineering. Word embeddings along with deep neural network methods have outperformed traditional methods as shown in [8]. CNN based methods capture the N-Gram features via convolutions whereas RNNs capture sequential information and dependencies. Architectures based on BiLSTM-CRF also have been employed for NLU task of sentiment analysis [9]. McCann et al. [10] makes use of contextualized word vectors to capture sentiment present in text utterances whereas Radford et al. [11] utilized byte-level recurrent language models for sentiment analysis.

Deep learning methods are widely employed, but most of them contain either just sequential information or structural information. Even after that, they are not able to capture the complete meaning and context of the sentence. Also, deep networks lack generalization over unseen input instances because of model complexity. Another problem with word embeddings is that they capture only distributional information but

not the polarity. Sentiment analysis is heavily dependent on polarity information. The lexicon-based polarity information has been integrated by Shin et al. [12] with word embedding which is an improvement that shows in their result but again their network architecture is not general and suffers from specific task limitations. Also, most of these models are shallow and unidirectional. Bi-directionality, if any is at a very shallow level.

Recent work on topic classification [13–15] is based on a variant of Long-short term memory (LSTM) or CNN network. Howard et al. [13] have used fine-tuning over a pre-trained language model whereas Johnson et al. [15] employs a deep pyramid convolutional neural network for text categorization. Our proposed MTL based Transformer encoder outperforms the state-of-the-art architectures of sentiment and topic classification by a significant margin for three benchmarks NLU datasets. Our model and motivation behind it are described below.

3 Bidirectional Transformer Network

3.1 Motivation

Every computational step in a deep neural network involves an approximation, and that is the primary source of error propagation. Longer the computational path more the error introduced. That is where self-attention [16] is helpful. Transformer model is first introduced by Google [16] which makes use of self-attention layers that helps in reducing computational path length in a deep network. The Transformer network is essentially an Encoder-Decoder architecture for seq2seq learning tasks like Machine Translation, Named-entity recognition (NER) tagging, etc. We are using just the truncated (12 layers) encoder part of the Transformer since ours are classification tasks only and hence do not require a decoder. Encoder stack in our model consists of 12 layers of encoders arranged in a stacked manner. Each Encoder layer, refer Fig. 1, in turn, consists of two sublayers:

- Bidirectional Self-Attention layer
- Fully Connected layer

Self-attention layer enables context-aware learning for the encoder. Encoder layer makes use of a self-attention mechanism which looks at the whole sequence of words in a sentence while learning the embedding for a single word. This way encoder can capture the sentence level context in embedding for each word. In contrast to RNNs, encoder employs direct short circuit peep into the whole sentence context which relieves the network of the burden of approximating long-term path dependency. It also makes the attention deeply learn bidirectional context since the encoder is looking in both directions. The bi-directionality in other models based on CNNs or LSTMs [17] is shallow because they are just adding/concatenating forward and backward unidirectional context representations rather than utilizing a single bidirectional attention process.

The Transformer encoder is pre-trained for general language modeling task on a huge corpus such as Wikipedia and publicly available book-corpus. It is an

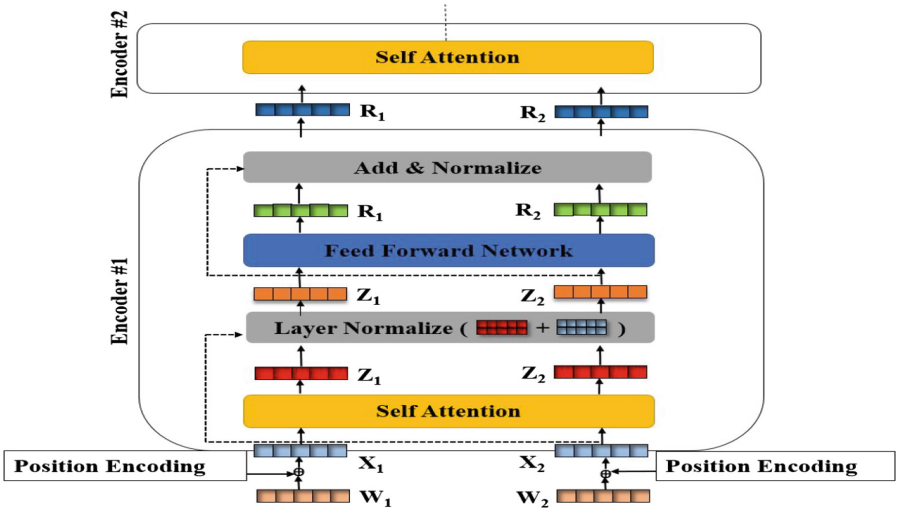

Fig. 1. Encoder layer of transformer network

unsupervised training for a masked language model task. This pre-training is important because language model is the most general form of language understanding and therefore the knowledge acquired is easily transferred across multiple tasks, which is very important for multi-task learning. We now describe the Transformer based model in detail.

3.2 Word Embedding Layer

Embedding layer is before the first encoder layer. Each word token in the sentence is a vector W, which is obtained by indexing a word-embedding matrix. After that W is transformed into vector X, refer Eq. 1, by adding position encoding to it. The dimensionality of X, W, P is 1024, which is the same as the hidden size H of an encoder. We experimented with different values of hidden size H and finalized it to be 1024 based on the performance on the validation set. We utilize position vectors to encode the relative sequential ordering of words in a sentence. For word embedding W, we used sub-word tokens as done by [18], with the vocabulary of 30,000 tokens. It is representative of that word for the current sentence, which captures bi-directional context information due to self-attention. So, the model learns different word embedding W for a word depending on the different context it appears in.

$$X = W + P \tag{1}$$

Where W is a sub-word vector, P is position vector, and X is the final representation of a single token and is input to the first encoder layer. The first token in any training example is always a CLS token of the size of H; it is essentially a special classification token containing the pooled (sum along the sequence) representation of the sequence.

CLS token in the final hidden state (i.e., the output of Transformer) is the sequence representation used in classification tasks.

3.3 Encoder Stack

The sequence of tensors X is fed into the first encoder layer, where self-attention layer projects it into three vectors, Query vector, a Key vector, and a Value vector. These vectors are obtained using three separate attention matrices W^Q, W^K, and W^V which are learned during training, and together they form an attention head. The Q, K, and V vectors are a different abstraction of the same word, each playing a different role during self-attention. Attention score of the current word with respect to each word in the sentence is calculated using these vectors in the following manner:

$$Q = X \cdot W^Q; K = X \cdot W^K; V = X \cdot W^V \tag{2}$$

$$Z = softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right)V \tag{3}$$

Query representation of current word W_1 (*i.e.,* Q_1) dot product with Key vector of some other word say W_2 (*i.e.*, K_2), gives an importance score of W_2 with respect to current word W_1. We then make use of softmax normalization to get the attention scores of all words in the sentence with respect to the current word W_1.

These attention scores are further multiplied with the value vectors to give weighted importance to each word and then summed along the sentence length to get Z vector, which is the output of self-attention layer for the current word/token. This score determines how much attention/focus to give other parts of the sentence while encoding the current word. Residual connection and layer normalization are done after each self-attention and feedforward layer.

$$Z = LayerNorm(X + Z) \tag{4}$$

Z is fed to feed-forward layer, which further processes it and forwards it to next encoder layer. The feed-forward layer is of size 4H (H is hidden layer size of Encoder as described in Sect. 3.2). Dropout is used for regularization after each sublayer before adding it to the original input to the sublayer by a residual connection. The dropout rate is a hyper-parameter and trained as part of the training process.

3.4 Combining the Multi-headed Attention

Each self-attention layer has multiple attention heads. The number of attention heads is a hyperparameter. We used 12 heads in each self-attention layer for our tasks. Each attention head of a self-attention layer calculates its representation Z, as explained above, which are concatenated and multiplied with a weight matrix W^O, to get the final output for the fully connected output layer. Each head focusses on different parts of the sentence to learn different dependencies and context rather than learning a particular

peculiarity, which is then combined to form the complete embedding of the word/token for better generalization.

3.5 Feed-Forward Layer

The output of self-attention layer, Z is input to this layer which is nothing but one layer fully connected network with hidden size 4H and output size H.

$$R = feedforward(Z) \tag{5}$$

$$R = LayerNormalize(R + Z) \tag{6}$$

R is the final output of one encoder layer which is fed to the encoder stacked above.

4 Multi-Task Learning Approach

A simple technique for performing MTL is to train the target and auxiliary tasks simultaneously. The model parameters are shared between tasks in this setting, pushing the model to learn the representation of features that generalize better across tasks.

4.1 Auxiliary Task Definition

We defined a task-specific auxiliary problem to be used in parallel during training time with our main natural language processing task of sentiment, and topic classification. We employ unsupervised k-means algorithm in conjunction with input n-grams for the auxiliary sub-task definition. Following subsections will briefly discuss the auxiliary sub-task definition for different main NLU tasks.

Movie Review Auxiliary Task-1 Definition: Movie review (MR) corpus is composed of text sequences with either positive or negative polarity. Each text sequence represents a single sentence consisting of words from the English dictionary. The polarity of any text sequence is highly dependent on the consisting words and context of those words, where the context of a word is defined by the content before and after the input word. To explicitly model individual words meaning and its context, we make use of word-based n-grams in conjunction with k-means algorithm, where n is a hyperparameter which is tuned as part of the training process.

For an input data instance $X = \{x_1, x_2, \ldots, x_l\}$, where x_i represents word at the i^{th} position and l denotes the length of the input text, we generate a set of n-grams. For example, if $n = 2$, the generated set will consist of all unigrams $\{x_i, i = 1, 2, \ldots, l\}$ and bigrams $\{[x_i, x_{i+1}], i = 1, 2, \ldots, l-1\}$. Similarly, this process is repeated for all data instances present in the MR corpus which results in an exhaustive list of n-grams. The list of n-grams generated from the training part of MR dataset is being used as an input to the k-means clustering algorithm.

For MR corpus, we assumed that polarity of each n-gram could be categorized into 3 clusters named as positive, negative and neutral as utterances in MR corpus belongs to either positive or negative sentiment. We make use of the K-means clustering

algorithm to generate an assignment of each n-gram to a particular cluster. Embeddings extracted from Google's pre-trained word2vec model is used as input to the clustering algorithm. A 300 dimension embedding replaces each n-gram, and n-grams with more than one words are replaced by the average of the individual word embeddings. To initialize the centroids of 3 clusters (positive, negative and neutral), we selected m words relevant to each cluster, where m is a tunable parameter, and used the average of the word embeddings obtained using word2vec as centroid values.

We will briefly describe the k-means algorithm used for generating the assignment of individual n-grams. As mentioned above, we make use of a vector of 300 dimensions to represent data and centroid points. After initialization of all the points, the algorithm works in two phases: The first phase includes using a distance metric like Euclidean distance ($d(x, y)$) to assign each data point to a particular cluster. When all the data points are assigned to some cluster, the second phase starts which recalculate the centroid of all the clusters by taking the average of the data points assigned to it. This two-process keeps on repeating until there is no changing in the cluster assignment in some iteration.

$$d(x, y) = \left[\sum_{i=1}^{k} (x_i - y_i)^2\right]^{\frac{1}{2}} \tag{7}$$

Euclidean Distance $d(x, y)$ is used for calculating the distance between a data point and centroids, where k denotes the length (300) of vector assigned to datapoints and centroids. After the convergence of the k-means algorithm, a list of n-grams and its corresponding cluster (positive, negative and neutral) assignment is generated. We make use of this n-gram and cluster assignment to define our auxiliary task as follows:

- For each text sequence $\{x_1, x_2, \ldots, x_l\}$, where l denotes the length of the sequence in the corpus. We assign a cluster class to each word x_i based on the majority class of all of its n-grams. In case of a tie, we break it by choosing the cluster class of bigger n-gram. For example, if we are using $n = 2$ and unigram of a word outputs cluster label as neutral and its bigram outputs positive then will take positive as our final cluster class for that particular word x_i assuming that bigger n-gram can capture the context in a better way.
- This process generates a set of data points $< x, y >$, where $x = \{x_1, x_2, \ldots, x_l\}$ represents input text sequence and $y = \{y_1, y_2, \ldots, y_l\}$ represents its corresponding cluster label sequence. This novel sequence to sequence mapping problem is used as an auxiliary task in our proposed network.

Movie Review Auxiliary Task-2 Definition: Part-of-speech (POS) tagging of a word make use of both its definition and context-i.e., the relationship of the word with content before and after the word. Therefore, a single word can have different POS tag based on the context in which it is used, for example in the sentence "Tell me your answer," answer is a noun whereas in "Answer the question," it is being used as a verb. This indicates that POS tagging can help in understanding and extracting relationship present in the given input text instance. Following the same intuition, we decided to make use of POS tagging as an auxiliary task in parallel with our main tasks. We make

use of NLTK POS tagger to generate POS tags for each data instance present in the MR corpus. For input instance $x = \{x_1, x_2, \ldots, x_l\}$, where l denotes the length of the sequence, we will have a corresponding POS tag set represented by $p = \{p_1, p_2, \ldots, p_l\}$ of the same length. We make use of a set of 15 POS tags as our output class corresponding to each word.

Auxiliary Tasks Definition for AG News and SST-2 Corpus: AG News consists of text utterances that belong to one of the four news categories, making it a multi-class classification problem whereas SST-2 is a binary sentiment classification task like MR classification problem. For AG News corpus, auxiliary task-1 assigns each word to one of the five clusters representing world news, business news, sci-tech news, sports news and neutral nature using a similar strategy as defined in auxiliary task 1 definition for MR corpus. Following the same definition of auxiliary task-1 and task-2 of MR task, we define sub-tasks for AG News and SST-2 corpus.

MTL based learning framework, refer Fig. 2(a), is trained by optimizing joint loss L_{MTL}, refer Eq. 8, which consists of $L_{main-task}$, loss function of main task and $L_{auxiliary-task}$, loss function of auxiliary task. The hyperparameter λ is used to control the effect of auxiliary task loss function on the MTL loss function, and it is optimized as part of the training process.

$$L_{MTL} = L_{main-task} + \lambda * L_{auxiliary-task} \tag{8}$$

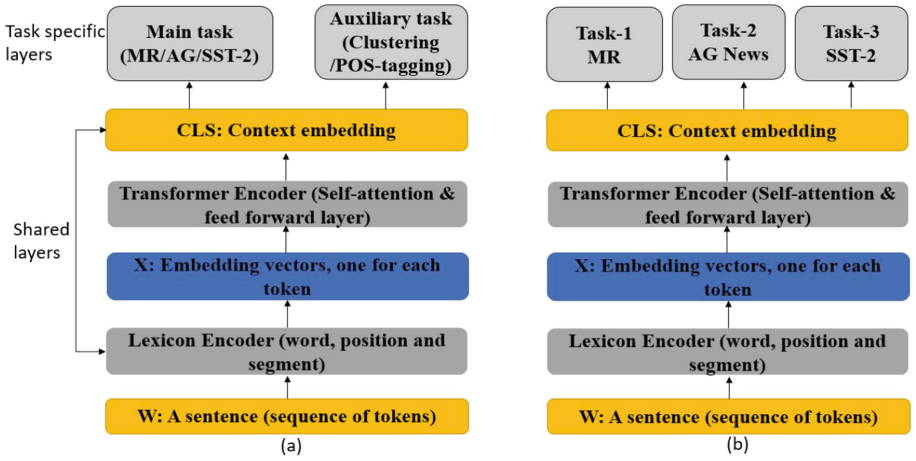

Fig. 2. (a) Auxiliary task-based MTL framework. (b) Learning multiple tasks in parallel

4.2 Single-Task Model

We also analyzed the performance of the standalone Transformer encoder stack for each task separately. The final output of the last layer of Transformer encoder is pooled (summed), which is effectively CLS token from the final layer (as described in Sect. 3.2), and is further input to a fully connected and then to a softmax layer. Softmax

output is a conditional class distribution over the vocabulary of output labels. We make use of Cross-Entropy loss function, which is nothing but Negative log likelihood of model output distribution (P) under the empirical distribution of the data (X).

$$Loss = -E_{p_{data(y|x)}} logp(y|x) \tag{9}$$

Where $p_{data(y|x)}$ is empirical distribution of the data(X).

4.3 Multi-task Model

Training related tasks in parallel leads to learning robust shared features [19]. It is a form of inductive bias like other regularization techniques, which causes the model not to learn peculiarities of a single task but rather prefer models, which can explain multiple tasks. This leads the learning to converge to models with better generalization errors. We used hard parameter sharing method for multi-task learning approach, which is generally applied by sharing the hidden layers among all tasks while keeping the output and softmax layers different and task specific.

We are inspired by [19], which showed that chances of overfitting are inversely proportional to the number of tasks jointly being learned by hard parameter sharing. Multi-task learning leads to implicit Data Augmentation and Transfer Learning. Different task dataset has different kind of inherent noise which if learned in a stand-alone manner leads to overfitting to corresponding data. Whereas, training two or more tasks jointly reduces the risk of overfitting by enabling the model to average the noise rather than sticking to particular data distribution. Following this, refer Fig. 2(b), we trained AG, MR and SST-2 joint model:

Training Schedule: In every epoch, we are shuffling the individual datasets for each sub-tasks. After that, refer Eqs. (10)–(12), we prepare separate batches for each dataset represented by B_k, which leads to a total number of batches N_{Batch}. We merge all the batches in a single file which is then randomly shuffled and used to train on the joint loss L_{total}.

Each mini-batch is task-specific and updates model weights by minimizing task-specific part of the joint loss function L_{total}.

$$D_i = Shuffle(D_i), for\ i = 1, 2, 3 \tag{10}$$

$$N_{Batch} = \frac{1}{Batch\_Size} \sum_{i=1}^{3} size(D_i) \tag{11}$$

$$\cup_{k=1}^{N_{Batch}} B_k \tag{12}$$

Joint loss function, which is a summation of individual loss, functions for each of the subtasks is defined as

$$L_{total} = L_{AG} + L_{MR} + L_{SST} \tag{13}$$

Where L_{AG}, L_{MR}, L_{SST} are cross-entropy loss functions for individual subtasks and L_{total} is the effective joint loss used for optimizing model parameters of our proposed network.

5 Datasets

Sentiment Analysis: Our model is trained separately on two datasets for sentiment analysis: Stanford Sentiment Treebank (SST) [20] and Movie Review (MR) [7]. Both dataset consists of movie reviews and their sentiment. We use each dataset's binary version. SST-2 contains 76961 phrases, 6919 sentences for training and 1820 sentences for testing. MR corpus consists of 10,662 reviews belonging to either positive or negative sentiment, with 5331 reviews of each class. We make use of 5-fold cross-validation to demonstrate the effectiveness of our proposed approaches.

Topic Classification: :The dataset is AG News corpus [21] in which the articles are divided into four categories. Four categories represent news of different domains, where domains are World, Business, Sports, and SciTech. This dataset comprises of 120,000 training and 7,600 test data instances.

6 Discussion

The presented accuracy Table 1 compares the performance of our proposed models with the current state-of-the-art architectures and indicates a significant reduction in the generalization error. We introduced four networks based on bidirectional transformer based encoder, MTL technique and joint training of various NLU tasks and all of them achieved start-of-the-art accuracies on the tasks of MR, AG News, SST-2 classification. The inclusion of shared learning through MTL and joint training of multiple tasks showed consistent improvement over the performance of standalone architecture.

Table 1. Datasets classification accuracies

Model architecture	MR	AG news	SST-2
BiLSTM-CRF [9]	82.3	-	-
WCCNN [14]	83.8	-	-
KPCN [14]	-	88.4	-
DPCNN [15]	-	93.1	-
BCN + Char + CoVe [10]	-	-	90.3
bmLSTM [11]	-	-	91.8
Transformer Network (TN)	87.7	93.9	92.3
TN + POS auxiliary task	88.8	94.1	93.7
TN + Clustering auxiliary task	89.1	**94.5**	94.6
TN + All tasks trained in parallel (MR + AG News + SST-2)	**89.8**	94.2	**95.1**

For MR corpus, our best model achieved more than 6% improvement in overall accuracy over the state-of-the-art accuracies mentioned in [9, 14] whereas for AG News and SST-2 corpus our best model achieved 1.4% and 3.3% overall accuracy improvement respectively compared to the current state-of-the-art results.

7 Conclusion

In this paper, we analyzed the effectiveness of MTL based bidirectional transformer architecture for sentiment classification (MR and SST-2) and topic classification (AG News) and showcased consistent improvement over current state-of-the-art architectures. We introduced MTL based learning by proposing clustering based sequence-to-sequence auxiliary task as well as by using POS-tagging as one of the auxiliary task to further enhance the performance of our transformer network. We observed significant improvement with the addition of MTL framework when compared with the standalone network. We also analyzed the performance of our transformer based network by training it with a combined corpus of MR, AG News, and SST-2, which led to the improvement in generalization ability of our network across all the tasks. Our best model improved state-of-the-art overall accuracy of MR, AG News and SST-2 corpus by 6%, 1.4%, and 3.3% respectively. For future work, we will work on proposing other related auxiliary tasks that can be jointly trained with the main tasks to improve model's generalization ability.

References

1. Gao, J., Galley, M., Li, L.: Neural approaches to conversational AI. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 1371–1374. ACM (2018)
2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018)
3. Zhang, Y., Yang, Q.: A survey on multitask learning, arXiv:1707.08114 [cs], July 2017. http://arxiv.org/abs/1707.08114
4. Liu, X., Gao, J., He, X., Deng, L., Duh, K., Wang, Y.: Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In: Proceedings of NAACL (2015)
5. Luong, M., Le, Q., Sutskever, I., Vinyals, O., Kaiser, L.: Multitask sequence to sequence learning. In: Proceedings of ICLR, pp. 1–10 (2016)
6. Mullen, T., Collier, N.: Sentiment analysis using support vector machines with diverse information sources. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (2004)
7. Pang, B., Lee, L.: Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 115–124. Association for Computational Linguistics, June 2005
8. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, October 2014, pp. 1746–1751. Association for Computational Linguistics (2014)

9. Chen, T., Xu, R., He, Y., Wang, X.: Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN. Expert. Syst. Appl. **72**, 221–230 (2017)
10. McCann, B., Bradbury, J., Xiong, C., Socher, R.: Learned in translation: contextualized word vectors. In: NIPS (2017)
11. Radford, A., Jozefowicz, R., Sutskever, I.: Learning to Generate Reviews and Discovering Sentiment, arXiv:1704.01444 [cs], April 2017. http://arxiv.org/abs/1704.01444
12. Shin, B., Lee, T., Choi, J.D.: Lexicon integrated CNN models with attention for sentiment analysis. In: Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pp. 149–158 (2017)
13. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. In: Proceedings of ACL, pp. 328–339 (2018)
14. Wang, J., Wang, Z., Zhang, D., Yan, J.: Combining knowledge with deep convolutional neural networks for short text classification. In: Proceedings of IJCAI (2017)
15. Johnson, R., Zhang, T.: Deep pyramid convolutional neural networks for text categorization. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1, pp. 562–570 (2017)
16. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
17. Peters, M.E., et al.: Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT), New Orleans, Louisiana (2018)
18. Wu, Y., et al.: Google's neural machine translation system: bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016)
19. Baxter, J.: A Bayesian/information theoretic model of learning to learn via multiple task sampling. Mach. Learn. **28**(1), 7–39 (1997)
20. Socher, R., et al.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of EMNLP, pp. 1631–1642 (2013)
21. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances in Neural Information Processing Systems, pp. 649–657 (2015)