Reading by Translating

Anonymous Author(s)
Affiliation
Address
email

Abstract

A machine reader is considered to be strong if it can create meaningful translations. Based on this idea, we propose a method called learning to read by translating.

1 Introduction

The major contributions of this paper include:

-

2 Related Works

2.1 Text Representation Learning

The GPT model [1] is a language model (LM) based on Transformer [2]. Different from Transformer which defines a conditional probability on an output sequence given an input sequence, GPT defines a marginal probability on a single sequence. In GPT, conditional probability of the next token given a historical sequence is defined using a Transformer decoder. Weight parameters are learned by maximizing likelihood on token sequences. BERT [3] aims to learn a Transformer encoder for representing texts. BERT's model architecture is a multi-layer bidirectional Transformer encoder. In BERT, Transformer uses bidirectional self-attention. To train the encoder, BERT masks some percentage of input tokens at random, and then predicts those masked tokens by feeding hidden vectors (produced by the encoder) corresponding to masked tokens into an output softmax over word vocabulary. BERT-GPT [4] is a model used for sequence-to-sequence modeling where a pretrained BERT is used to encode input text and GPT is used to generate output texts. In BERT-GPT, pretraining of BERT encoder and GPT decoder is conducted separately, which may lead to inferior performance. Auto-Regressive Transformers (BART) [5] has a similar architecture as BERT-GPT, but trains BERT encoder and GPT decoder jointly. To pretrain BART weights, input texts are corrupted randomly, such as token masking, token deletion, text infilling, etc., then a network is learned to reconstruct original texts. ALBERT [6] uses parameter-reduction methods to reduce memory consumption and increase training speed of BERT. It also introduces a self-supervised loss which models inter-sentence coherence. RoBERTa [7] is a replication study of BERT pretraining. It shows that BERT's performance can be greatly improved by carefully tuning training processes, such as (1) training models longer, with larger batches, over more data; (2) removing the next sentence prediction objective; (3) training on longer sequences, etc. XLNet [8] learns bidirectional contexts by maximizing expected likelihood over all permutations of factorization order and uses a generalized autoregressive pretraining mechanism to overcome the pretrain-finetune discrepancy of BERT. T5 [9] compared pretraining objectives, architectures, unlabeled datasets, transfer approaches on a wide range of language understanding tasks and proposed a unified framework that casts these tasks as a text-to-text task. ERNIE 2.0 [10] proposed a continual pretraining framework which builds and learns incrementally pretraining tasks through constant multi-task learning, to capture lexical, syntactic and

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

Notation	Meaning
E	Encoder weights
F	Decoder weights
W	Network weights of MT model
$D^{(tr)}$	MT training dataset
$D^{(val)}$	MT validation dataset
U	Unlabeled text dataset

Table 1: Notations

semantic information from training corpora. [11] proposed task adaptive pretraining (TAPT) and domain adaptive pretraining (DAPT). Given a RoBERTa model pretrained on large-scale corpora, TAPT continues to pretrain RoBERTa on training dataset of target task. DAPT continues to pretrain RoBERTa on datasets that have small domain differences with data in target tasks.

2.2 Text Classification

Text classification [12] is one of the key tasks in natural language processing and has a wide range of applications, such as sentiment analysis, spam detection, tag suggestion, etc. A number of approaches have been proposed for text classification. Many of them are based on RNNs. [13] use multi-task learning to train RNNs, utilizing the correlation between tasks to improve text classification performance. [14] generalize sequential LSTM to tree-structured LSTM to capture the syntax of sentences for achieving better classification performance. Compared with RNN-based models, CNN-based models are good at capturing local and position-invariant patterns. [15] proposed dynamic CNN (DCNN), which uses dynamic k-max-pooling to explicitly capture short-term and long-range relations of words and phrases. [16] proposed a character-level CNN model for text classification, which can deal with out-of-vocabulary words. Hybrid methods combine RNN and CNN to explore the advantages of both of them. [17] proposed a convolutional LSTM (C-LSTM) Network, which uses a CNN to extract phrase-level representations, then feeds them to an LSTM network to represent the whole sentence.

3 Method

In our framework, there are three learning stages. In the first stage, a model learns to create translations. The model has an encoder-decoder architecture. The encoder takes a source text as input and produces an embedding. The embedding is fed into a decoder which decodes a translation into a target text. In the first stage, we use a machine translation dataset to train the encoder and decoder. Each example in the machine translation (MT) dataset is associated with a weight. This amounts to solving the following problem:

$$E^*(A), F^*(A) = \min_{E,F} \sum_{i=1}^{N} a_i L(E, F, d_i^{(tr)}). \tag{1}$$

where F denotes the network weights of the coder and E denotes the network weights of the encoder. $A = \{a_i\}_{i=1}^N$ denote weights of MT training examples. N is the number of MT training examples.

In the second stage, we use the trained encoder and decoder to generate a pseudo MT dataset. Then use this dataset to train an MT model, which amounts to solving the following optimization problem:

$$W^*(E(A), F(A)) = \min_W \; L(W, U, E^*(A), F^*(A)). \tag{2}$$

Given some unlabeled texts U, we use $E^*(A)$ and $F^*(A)$ to generate an MT dataset. Then we use this dataest to train an MT model whose network weights are W.

Finally, we evaluate $W^*(E(A), F(A))$ on the validation set of the MT dataset and learn A by minimizing the validation loss.

Putting these pieces together, we have the following optimization problem.

$$\begin{aligned} &\min_A \; L(W^*(E^*(A), F^*(A)), D^{(val)}) \\ &s.t. \;\; W^*(E^*(A), F^*(A)) = \min_W \; L(W, U, E^*(A), F^*(A)) \\ &\quad E^*(A), F^*(A) = \min_{E,F} \; \textstyle\sum_{i=1}^{N} a_i L(E, F, d_i^{(tr)}) \end{aligned} \tag{3}$$

Domain	Dataset	Label Type	Train	Dev	Test	Classes
BIOMED	CHEMPROT	relation classification	4169	2427	3469	13
	RCT	abstract sent. roles	180040	30212	30135	5
CS	ACL-ARC	citation intent	1688	114	139	6
	SCIERC	relation classification	3219	455	974	7
NEWS	HYPERPARTISAN	partisanship	515	65	65	2
	AGNEWS	topic	115000	5000	7600	4
REVIEWS	HELPFULNESS	review helpfulness	115251	5000	25000	2
	IMDB	review sentiment	20000	5000	25000	2

Table 2: Statistics of datasets used in [11]. Sources: CHEMPROT [18], RCT [19], ACL-ARC [20], SCIERC [21], HYPERPARTISAN [22], AGNEWS [16], HELPFULNESS [23], IMDB [24]. This table is taken from [11].

where $A = \{a_i\}_{i=1}^N$. U is a set of unlabeled texts.

3.1 Optimization Algorithm

In this section, we develop an optimization algorithm to solve the problem defined in Eq.(). We approximate $E^*(A)$ and $F^*(A)$ using one-step gradient descent w.r.t $\sum_{i=1}^N a_i L(E, F, d_i^{(tr)})$:

$$E^*(A) \approx E' = E - \eta_e \nabla_E \sum_{i=1}^{N} a_i L(E, F, d_i^{(tr)}), \tag{4}$$

$$F^*(A) \approx F' = F - \eta_f \nabla_F \sum_{i=1}^{N} a_i L(E, F, d_i^{(tr)}). \tag{5}$$

We plug E' and F' into $L(W, U, E^*(A), F^*(A))$ and get an approximated objective. We approximate $W^*(E^*(A), F^*(A))$ using one-step gradient descent w.r.t the approximated objective:

$$W^*(E^*(A), F^*(A)) \approx W' = W - \eta_w \nabla_W L(W, U, E', F'). \tag{6}$$

We plug W' into $L(W^*(E^*(A), F^*(A)), D^{(val)})$ and get an approximated objective. We update A using gradient descent:

$$A \leftarrow A - \eta_a \nabla_A L(W', D^{(val)}), \tag{7}$$

where

$$\nabla_A L(W', D^{(val)}) = (\frac{\partial E'}{\partial A}\frac{\partial W'}{\partial E'} + \frac{\partial F'}{\partial A}\frac{\partial W'}{\partial F'})\nabla_{W'} L(W', D^{(val)}) \tag{8}$$

$$(\eta_e \eta_w \nabla^2_{A,E}(\sum_{i=1}^{N} a_i L(E, F, d_i^{(tr)}))\nabla^2_{E',W} L(W, U, E', F') + \eta_d \eta_w \nabla^2_{A,F} \tag{9}$$

$$(\sum_{i=1}^{N} a_i L(E, F, d_i^{(tr)}))\nabla^2_{F',W} L(W, U, E', F'))\nabla_{W'} L(W', D^{(val)}) \tag{10}$$

Algorithm 1 Optimization algorithm for reading by translating

1: **while** not converged **do**
2: Update encoder weights E using Eq.(4)
3: Update decoder weights F using Eq.(5)
4: Update RC model weights W using Eq.(6)
5: Update importance weights A using Eq.(7)
6: **end while**

	CoLA	RTE	QNLI	STS-B	MRPC	WNLI	SST-2	MNLI (m/mm)	QQP	AX
Train	8551	2490	104743	5749	3668	635	67349	392702	363871	-
Dev	1043	277	5463	1500	408	71	872	9815/9832	40432	-
Test	1063	3000	5463	1379	1725	146	1821	9796/9847	390965	1104

Table 3: GLUE dataset statistics.

Dataset	**RoBERTa**	**DAPT**	**TAPT**	**SSL-Reg**	**Baseline X**	**RCRCT (Ours)**
CHEMPROT	$81.9_{1.0}$	$84.2_{0.2}$	$82.6_{0.4}$	$83.1_{0.5}$		
RCT	$87.2_{0.1}$	$87.6_{0.1}$	$\mathbf{87.7}_{0.1}$	$87.4_{0.1}$		
ACL-ARC	$63.0_{5.8}$	$75.4_{2.5}$	$67.4_{1.8}$	$69.3_{4.9}$		
SCIERC	$77.3_{1.9}$	$80.8_{1.5}$	$79.3_{1.5}$	$81.4_{0.8}$		
HYPERPARTISAN	$86.6_{0.9}$	$88.2_{5.9}$	$90.4_{5.2}$	$92.3_{1.4}$		
AGNEWS	$93.9_{0.2}$	$93.9_{0.2}$	$\mathbf{94.5}_{0.1}$	$94.2_{0.1}$		
HELPFULNESS	$65.1_{3.4}$	$66.5_{1.4}$	$68.5_{1.9}$	$69.4_{0.2}$		
IMDB	$95.0_{0.2}$	$95.4_{0.1}$	$95.5_{0.1}$	$95.7_{0.1}$		

Table 4: Results on datasets used in [11]. For vanilla (unregularized) RoBERTa, DAPT, and TAPT, results are taken from [11]. For each method on each dataset, we run it for four times with different random seeds. Results are in m_s format, where m denotes mean and s denotes standard derivation. Following [11], for CHEMPROT and RCT, we report micro-F1; for other datasets, we report macro-F1.

4 Experiments

4.1 Dataset

We evaluated our method on the datasets used in [11], which are from various domains. For each dataset, we follow the train/development/test split specified in [11]. Dataset statistics are summarized in Table 2. In addition, we performed experiments on the datasets in the GLUE benchmark [26]. The General Language Understanding Evaluation (GLUE) benchmark has 10 tasks, including 2 single-sentence tasks, 3 similarity and paraphrase tasks, and 5 inference tasks. For each GLUE task, labels in development sets are publicly available while those in test sets are not released. We obtain performance on test sets by submitting inference results to GLUE evaluation server[1]. Table 3 shows the statistics of data split in each task.

4.2 Experimental Settings

4.2.1 Baselines

For experiments on datasets used in [11], we use RoBERTa [7] as the base model for text classification. For experiments on GLUE datasets, we experimented BERT [7] and RoBERTa as base models for text classification.

We compare our proposed RCRCT method with the following baselines.

- **Unregularized RoBERTa** [25]. In this approach, the Transformer encoder is initialized with pretrained RoBERTa. Then the pretrained encoder and a classification head form a text classification model, which is then finetuned on a target classification task. Architecture of the classification model is the same as that in [25]. Specifically, representation of the `[CLS]` special token is passed to a feedforward layer for class prediction. Nonlinear activation function in the feedforward layer is tanh. This approach is evaluated on all datasets used in [11] and all datasets in GLUE.
- **Unregularized BERT.** This approach is the same as unregularized RoBERTa, except that the Transformer encoder is initialized by pretrained BERT [3] instead of RoBERTa. This approach is evaluated on all GLUE datasets.

[1] https://gluebenchmark.com/leaderboard

	CoLA (Matthew Corr.)	SST-2 (Accuracy)	RTE (Accuracy)	QNLI (Accuracy)	MRPC (Accuracy/F1)
The median result					
BERT, Lan et al. 2019	60.6	93.2	70.4	92.3	**88.0/-**
BERT, our run	62.1	93.1	74.0	92.1	86.8/90.8
TAPT	61.2	93.1	74.0	92.0	85.3/89.8
SSL-Reg (SATP)	63.7	**93.9**	**74.7**	92.3	86.5/90.3
SSL-Reg (MTP)	**63.8**	93.8	**74.7**	**92.6**	87.3/90.9
RCRCT (Ours)					
The best result					
BERT, our run	63.9	93.3	75.8	92.5	**89.5/92.6**
TAPT	62.0	93.9	76.2	92.4	86.5/90.7
SSL-Reg (SATP)	65.3	94.6	**78.0**	92.8	88.5/91.9
SSL-Reg (MTP)	**66.3**	**94.7**	**78.0**	**93.1**	**89.5**/92.4
RCRCT (Ours)					

Table 5: Results of BERT-based experiments on GLUE development sets, where results on MNLI and QQP are the median of five runs and results on other datasets are the median of nine runs. The size of MNLI and QQP is very large, taking a long time to train on. Therefore, we reduced the number of runs. Because we used a different optimization method to re-implement BERT, our median performance is not the same as that reported in [6].

	MNLI-m/mm (Accuracy)	QQP (Accuracy/F1)	STS-B (Pearson Corr./Spearman Corr.)	WNLI (Accuracy)
The median result				
BERT, Lan et al. 2019	86.6/-	91.3/-	90.0/-	-
BERT, our run	86.2/86.0	91.3/88.3	90.4/90.0	56.3
TAPT	85.6/85.5	91.5/88.7	90.6/90.2	53.5
SSL-Reg (SATP)	86.2/86.2	91.6/88.8	**90.7/90.4**	56.3
SSL-Reg (MTP)	**86.6/86.6**	**91.8/89.0**	90.7/90.3	56.3
RCRCT (Ours)				
The best result				
BERT, our run	86.4/86.3	91.4/88.4	90.9/90.5	56.3
TAPT	85.7/85.7	91.7/89.0	90.8/90.4	56.3
SSL-Reg (SATP)	86.4/86.5	91.8/88.9	**91.1/90.8**	**59.2**
SSL-Reg (MTP)	**86.9/86.9**	**91.9/89.1**	**91.1/90.8**	57.7
RCRCT (Ours)				

Table 6: Continuation of Table 5.

- **Task adaptive pretraining (TAPT)** [11]. In this approach, given the Transformer encoder pretrained using RoBERTa or BERT on large-scale external corpora, it is further pretrained by RoBERTa or BERT on input texts in a target classification dataset (without using class labels). Then this further pretrained encoder is used to initialize the encoder in the text classification model and is finetuned to perform classification tasks which use both input texts and their class labels. TAPT is studied for all datasets in this paper.

- **Domain adaptive pretraining** (DAPT) [11]. In this approach, given a pretrained encoder on large-scale external corpora, the encoder is further pretrained on a small-scale corpora whose domain is similar to that of texts in a target classification dataset. Then this further pretrained encoder is finetuned in a classification task. DAPT is similar to TAPT, except that TAPT performs the second stage pretraining on texts T in the classification dataset while DAPT performs the second stage pretraining on external texts whose domain is similar to that of T rather than directly on T. The external dataset is usually much larger than T.

4.2.2 Hyperparameter Settings

Hyperparameters were tuned on development datasets.

	BERT	TAPT	SSL-Reg (SATP)	SSL-Reg (MTP)	RCRCT (Ours)
CoLA (Matthew Corr.)	60.5	61.3	**63.0**	61.2	
SST-2 (Accuracy)	94.9	94.4	95.1	**95.2**	
RTE (Accuracy)	70.1	70.3	71.2	**72.7**	
QNLI (Accuracy)	92.7	92.4	92.5	**93.2**	
MRPC (Accuracy/F1)	85.4/89.3	85.9/89.5	85.3/89.3	**86.1/89.8**	
MNLI-m/mm (Accuracy)	**86.7**/85.9	85.7/84.4	86.2/85.4	86.6/**86.1**	
QQP (Accuracy/F1)	89.3/72.1	89.3/71.6	89.6/72.2	**89.7/72.5**	
STS-B (Pearson Corr./Spearman Corr.)	87.6/86.5	**88.4**/87.3	88.3/**87.5**	88.1/87.2	
WNLI (Accuracy)	65.1	65.8	65.8	**66.4**	
AX(Matthew Corr.)	39.6	39.3	40.2	**40.3**	
Average	80.5	80.6	81.0	**81.3**	

Table 7: Results of BERT-based experiments on GLUE test sets, which are scored by the GLUE evaluation server (`https://gluebenchmark.com/leaderboard`). Models evaluated on AX are trained on the training dataset of MNLI.

	CoLA (Matthew Corr.)	SST-2 (Accuracy)	RTE (Accuracy)	QNLI (Accuracy)	MRPC (Accuracy/F1)
The median result					
RoBERTa, Liu et al. 2019	68.0	**96.4**	**86.6**	94.7	**90.9/-**
RoBERTa, our run	68.7	96.1	84.8	94.6	89.5/92.3
SSL-Reg (MTP)	**69.2**	96.3	85.2	**94.9**	90.0/**92.7**
RCRCT (Ours)					
The best result					
RoBERTa, our run	69.2	96.7	86.6	94.7	90.4/93.1
SSL-Reg (MTP)	**70.2**	96.7	86.6	**95.2**	**91.4/93.8**
RCRCT (Ours)					

Table 8: Results of RoBERTa-based experiments on GLUE development sets, where the median results are the median of five runs. Because we used a different optimization method to re-implement RoBERTa, our median performance is not the same as that reported in [25].

Hyperparameter settings for RoBERTa on datasets used in [11]. For a fair comparison, most of our hyperparameters are the same as those in [11]. The maximum text length was set to 512. Text encoders in all methods are initialized using pretrained RoBERTa [7] on a large-scale external dataset. For TAPT and DAPT, the second-stage pretraining on texts T in a target classification dataset or on external texts whose domain is similar to that of T is based on the pretraining approach in RoBERTa. For all datasets, we used a batch size of 16 with gradient accumulation. We used the AdamW optimizer [27] with a warm-up proportion of 0.06, a weight decay of 0.1, and an epsilon of 1e-6. In AdamW, β_1 and β_2 are set to 0.9 and 0.98, respectively. The maximum learning rate was 2e-5.

Hyperparameter settings for BERT on GLUE datasets. The maximum text length was set to 128. Since external texts whose domains are similar to those of the GLUE texts are not available, we did not compare with DAPT. For each method applied, text encoder is initialized using pretrained BERT [3] (with 24 layers) on a large-scale external dataset. In TAPT, the second-stage pretraining is performed using BERT. Batch size was set to 32 with gradient accumulation. We use the AdamW optimizer [27] with a warm-up proportion of 0.1, a weight decay of 0.01, and an epsilon of 1e-8. In AdamW, β_1 and β_2 are set to 0.9 and 0.999, respectively.

Hyperparameter settings for RoBERTa on GLUE datasets. Most hyperparameter settings follow those in RoBERTa experiments performed on datasets used in [11]. We set different learning rates and different epoch numbers for different datasets as guided by [25].

	MNLI-m/mm (Accuracy)	QQP (Accuracy)	STS-B (Pearson Corr./Spearman Corr.)	WNLI (Accuracy)
The median result				
RoBERTa, Liu et al. 2019	90.2/90.2	92.2	92.4/-	-
RoBERTa, our run	90.5/90.5	91.6	92.0/92.0	56.3
SSL-Reg (MTP)	**90.7/90.7**	91.6	92.0/92.0	**62.0**
RCRCT (Ours)				
The best result				
RoBERTa, our run	90.7/90.5	91.7	92.3/92.2	60.6
SSL-Reg (MTP)	90.7/90.5	**91.8**	92.3/92.2	**66.2**
RCRCT (Ours)				

Table 9: Continuation of Table 8.

4.3 Results

4.3.1 Results on the datasets used in [11].

Performance of text classification on datasets used in [11] is reported in Table 4. Following [11], for CHEMPROT and RCT, we report micro-F1; for other datasets, we report macro-F1. From this table, we make the following observations. **First**,

4.3.2 Results on the GLUE benchmark

Table 5 and Table 6 show results of BERT-based experiments on development sets of GLUE. As mentioned in [28], for the 24-layer version of BERT, finetuning is sometimes unstable on small datasets, so we run each method several times and report the median and best performance. Table 7 shows the best performance on test sets. Following [26], we report Matthew correlation on CoLA, Pearson correlation and Spearman correlation on STS-B, accuracy and F1 on MRPC and QQP. For the rest datasets, we report accuracy. From these tables, we make the following observations. **First**,

4.4 Ablation Studies

In this section, we perform ablation studies to investigate the importance of individual components in our framework. In each ablation study, we compare the ablation setting with the full framework. Specifically, we study the following ablation settings.

References

[1] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Qingyang Wu, Lei Li, Hao Zhou, Ying Zeng, and Zhou Yu. Importance-aware learning for neural headline editing. *arXiv preprint arXiv:1912.01114*, 2019.

[5] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[6] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[8] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764, 2019.

[9] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[10] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. Ernie 2.0: A continual pre-training framework for language understanding. *arXiv preprint arXiv:1907.12412*, 2019.

[11] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of ACL*, 2020.

[12] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep learning based text classification: A comprehensive review. *arXiv preprint arXiv:2004.03705*, 2020.

[13] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016.

[14] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July 2015. Association for Computational Linguistics.

[15] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[16] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NeurIPS*, 2015.

[17] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and F. C. M. Lau. A c-lstm neural network for text classification. *ArXiv*, abs/1511.08630, 2015.

[18] Jens Kringelum, Sonny Kim Kjærulff, Søren Brunak, Ole Lund, Tudor I. Oprea, and Olivier Taboureau. ChemProt-3.0: a global chemical biology diseases mapping. In *Database*, 2016.

[19] Franck Dernoncourt and Ji Young Lee. Pubmed 200k RCT: a dataset for sequential sentence classification in medical abstracts. In *IJCNLP*, 2017.

[20] David Jurgens, Srijan Kumar, Raine Hoover, Daniel A. McFarland, and Dan Jurafsky. Measuring the evolution of a scientific field through citation frames. *TACL*, 2018.

[21] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *EMNLP*, 2018.

[22] Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. SemEval-2019 Task 4: Hyperpartisan news detection. In *SemEval*, 2019.

[23] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *ACM SIGIR*, 2015.

[24] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *ACL*, 2011.

[25] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach, 2019. arXiv:1907.11692.

[26] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[27] I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101, 2017.

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.