Evaluation of BERT and XLNet Models on Irony Detection in English Tweets

Cheng Zhang† Masashi Kudo† Hayato Yamana‡

†Graduate School of Fundamental Science and Engineering, Waseda University 3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-8555 Japan

‡Faculty of Science and Engineering, Waseda University 3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-8555 Japan

E-mail: {zchelllo, kudoma34, yamana}@yama.info.waseda.ac.jp

Abstract Automatically detecting irony is helpful and important for mining fine-grained information from social web data. Therefore, the International Workshop on Semantic Evaluation (SemEval) presented the first shared task on irony detection called "Irony Detection in English Tweets" in 2018. For this task, the system should determine whether the tweet is ironic (Task A) and which type of irony is expressed (Task B). The top teams obtained F_1=0.71 for Task A and F_1=0.51 for Task B. These teams all used LSTM models exploiting some word embedding features like Glove embeddings. While in recent years more powerful models like BERT and XLNet appeared. Therefore, in our paper, we evaluate the performance of BERT and XLNet models on Irony Detection in English Tweets by two methods: word embedding method and fine-tuning method. Through the experiment, these two models could get relatively high scores showing that BERT and XLNet models are capable to understand the irony to some extent.

Keyword Irony Detection, Tweets, Natural Language Processing, BERT, XLNet

1. Introduction

The development of the social web has stimulated the use of figurative and creative language, including irony, in public [1]. According to literary scholars [2], irony was defined as a trope where the speaker intends to express a contradictory situation or the opposite meaning of what is literally said. It adopts a subtle technique where incongruity is used to suggest a distinction between reality and expectation in order to produce a humorous or emphatic effect on the listener. Because irony is widely used in social web, it has important implications for the natural language processing (NLP) tasks that aim to understand and generate human language.

Correctly recognizing the irony is helpful to understand the social web better, especially for some natural language processing applications such as sentiment analysis [3]. A typical sentiment analysis model will classify the tweet “Monday mornings are my fave #not” as positive, but the true result is that it expresses a negative feeling[4]. Moreover, the SemEval-2014 shared task “Sentiment Analysis in Twitter” [5] demonstrated the effect of irony on automatic sentiment classification. The results revealed that, while sentiment classification performance on regular tweets was F_1= 0.71, scores on the ironic tweets varied from F_1 = 0.29 to F_1= 0.57. Therefore, automatic irony detecting techniques are important to improve the performance of sentiment analysis.

Even for human beings, recognizing an irony is sometimes difficult, let alone the poor intelligence of machine. In other word, there is just subtle difference between ironic and non-ironic texts. By far researchers have tried many ways to identify the complicated irony in texts or tweets. Rule-based or machine learning-based methods are the two popular ways in this task [6]. Rules based methods usually depend on lexicons (positive or negative sense words) to identify irony [7][8]. But these rules-based methods cannot extract the contextual information from texts. Traditional machine learning-based methods such as SVM [9] are also effective for this task, but time-consuming manual feature engineering is necessary. Recently, deep learning techniques become hot in this task. For example, Ghosh et al. [3] proposed to use a CNN-LSTM model to deal with a binary irony classification task. Their method got a great performance even without heavy feature engineering.

Recently, the pretrained models such as BERT and XLNet break the state-of-the-art result in many datasets including GLUE[17] and SQuAD[18]. But these datasets are all used for testing the progress of general language understanding. Whether these pretrained models such as BERT and XLNet are appliable for the irony which is a special type in daily communication requires further research. Therefore, in this paper, we evaluate the performance of BERT and XLNet models on the shared task "Irony Detection in English Tweets" dataset based on two ways, word embedding method and fine-tuning method.

In the rest of this paper, the content of the irony detection task in SemEval is introduced in Section 2. The BERT and XLNet models are introduced in Section 3. The details of methods to adopt BERT and XLNet models are presented in Section 4. We show the experiment results in Section 5 and conclude in Section 6.

2. Irony Detection Task in SemEval

2.1. Task Description

To evaluate the automatic irony detection techniques, the SemEval 2018 presented a task called "Irony Detection in English Tweets". The subtask A is aimed to determine whether a tweet is ironic. The subtask B is aimed to identify three ironic types of tweets: *verbal irony by means of a polarity contrast*, *situational irony*, or *another type of verbal irony*. Definition and example are as follows [10]:

Verbal irony by means of a polarity contrast

For this type, the polarity (positive, negative) is inverted between the literal and the intended evaluation.

- *I love waking up with migraines #not* 😫

Situational irony

This type is about describing situational irony, or situations that fail to meet some expectations.

- *most of us didn't focus in the #ADHD lecture. #irony*

Another type of *verbal irony*

This category shows no polarity contrast between the literal and the intended evaluation but is nevertheless ironic.

- *Human brains disappear every day. Some of them have never even appeared. #brain #sarcasm*

2.2. Dataset

The detailed statistics of the dataset in this task[10] are shown in Table 1. The three labels (i.e., V, O, and S) represent the three types respectively: verbal irony by means of a polarity contrast, other types of verbal irony and situational irony. Note that Irony-related hashtags will not be present in the test set.

Table 1. The detailed statistics of dataset[10]

Task	A		B			
Label	Ironic	Non-ironic	V	O	S	Non-ironic
#train	1,901	1,916	1,383	202	316	1,916
#test	784		784			

2.3. Evaluation

For both subtasks, results were evaluated using standard evaluation metrics, including accuracy, precision, recall and F_1 score, calculated as follows:

$$accuracy = \frac{true\ positives + true\ negatives}{total\ number\ of\ instances} \tag{1}$$

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{2}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{3}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4}$$

In subtask A, the performance of systems is evaluated by F_1 for the positive class. In subtask B, the macro-averaged F_1 over all classes is used as the metric. Macro-averaging of the F_1 score implies that all class labels have equal weight in the final score.

2.4. Systems and results for Task A

The result of the top 5 teams in task A[10] is shown in Table 2. In this table, the teams are ranked by the official F_1 score.

Table 2. The top 5 teams for Task A[10]

Team	Acc	Precision	Recall	F_1
UCDCC	**0.797**	**0.788**	0.669	**0.724**
THU_NGN	0.735	0.630	0.801	0.705
NTUA-SLP	0.732	0.654	0.691	0.672
WLV	0.643	0.532	**0.836**	0.650
NLPRL-IITBHU	0.661	0.551	0.788	0.648
Unigram SVM BL	0.635	0.532	0.659	0.589

As shown in Table 2, the systems of the top five teams outperforms the unigram SVM baseline by a sizable margin. For these best 5 teams, they all used training data provided only, which means that they didn't use other similar datasets for training. The UCDCC team ($F_1 = 0.724$) developed a siamese architecture which consists of two subnetworks (containing an LSTM) making use of Glove word embeddings [11]. The THU_NGN team ($F_1 = 0.705$)

created a densely connected LSTMs architecture based on pretrained word embeddings, sentiment features and syntactic features [12]. The NTUA-SLP team ($F_1 = 0.672$) built an ensemble classifier of a word-based Bi-LSTM and a character-based Bi-LSTM making use of pretrained character and word embeddings on a corpus of 550 million tweets [13]. The WLV team ($F_1 = 0.650$) developed an ensemble voting classifier containing logistic regression (LR) and support vector machine (SVM) based on pretrained word and emoji embeddings [14]. The NLPRL-IITBHU team ($F_1 = 0.648$) used an XGBoost Classifier making use of ten types of handcrafted features based on DeepMoji [15].

2.5. Systems and results for Task B

The result of the top 5 teams in task B[10] is shown in Table 3 in which the teams are ranked by the official F_1 score. As observed in Table 3, the top 5 are UCDCC ($F_1 = 0.507$), NTUA-SLP ($F_1 = 0.496$), THU_NGN ($F_1 = 0.495$), NLPRL-IITBHU ($F_1 = 0.474$) and NIHRIO ($F_1 = 0.444$). The NIHRIO team was a multilayer perceptron-based architecture exploiting lexical, syntactic and semantic features (Glove word embeddings). Other teams' approaches have been introduced in section 2.4. All teams exploited different approaches but all of them clearly outperformed the baseline.

Table 3. The top 5 teams for Task B[10]

Team	Acc	Precision	Recall	F_1
UCDCC	**0.732**	**0.577**	0.504	**0.507**
NTUA-SLP	0.652	0.496	0.512	0.496
THU_NGN	0.605	0.486	**0.541**	0.495
NLPRL-IITBHU	0.603	0.466	0.506	0.474
NIHRIO	0.659	0.545	0.448	0.444
Unigram SVM BL	0.563	0.416	0.364	0.341

3. New pretraining methods for NLP

3.1. BERT

In 2018, Jacob et al proposed a new language representation model called BERT[16]. The whole name of BERT is Bidirectional Encoder Representations from Transformers. BERT is pretrained on unlabeled text (Wikipedia) by jointly conditioning on both left and right context in all layers and has obtained state-of-the-art results on eleven natural language processing tasks, including increasing the GLUE[17] to 80.5%, MultiNLI accuracy to 86.7%, SQuAD v1.1[18] question answering F_1 score to 93.2 and SQuAD v2.0 F_1 score to 83.1.

3.2. XLNet

In 2019, Yang et al. proposed a generalized autoregressive pretraining method called XLNet that uses a permutation language modeling objective to combine the advantages of autoregressive and autoencoder methods [19]. They claimed that XLNet outperformed BERT on 20 tasks by a sizable margin, including question answering, natural language inference, sentiment analysis and document ranking.

4. Methods to Adopt BERT/XLNet

Our goal is to evaluate the performance of BERT and XLNet models in this task. For these two models, there are two ways to exploit BERT and XLNet models for classification tasks. One is called word embedding method, i.e., using pre-trained model to transform text into embeddings and classifying them with traditional machine learning techniques such as SVM and LR. The other one is to fine-tune the pre-trained model directly on the dataset of irony detection task in SemEval-2018. We will introduce these two methods in detail below.

4.1. Word Embedding Method

The general procedure for the word embedding method is shown in Figure1. At first, we built a tokenizer to transform text to token ids and pad these id lists to the same length. In order to let the model know which position is the padding, we should also generate attention masks. Then we send token ids and attention masks into the model and get the output. Here we just use a part of output as word embedding. Finally, we develop a traditional machine learning model to fit these word embeddings. After the supervised learning, then this model can predict the label of the given tweet.

In this method, we choose several traditional machine learning classification techniques including Random Forest(RF), Naïve Bayesian(NB), Logistic Regression(LR) and Support Vector Machine(SVM).

4.2. Fine-tuning Method

The general procedure for the fine-tuning method is shown in Figure 2. At first, we send the training dataset into pre-trained model. The training dataset for this task is small compared with other datasets. After fine-tuning, we send token ids and attention masks of the test dataset into this fine-tuning model for prediction.

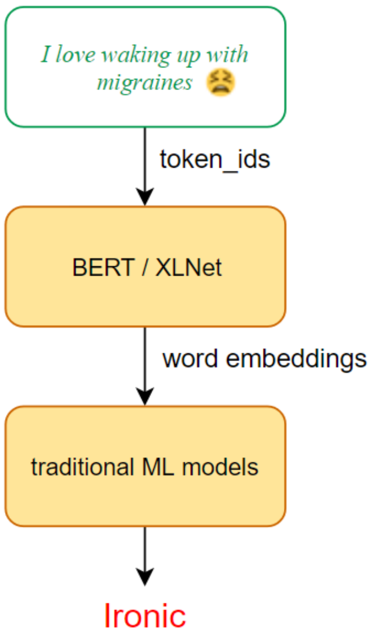

Figure 1: Word Embedding Method Structure

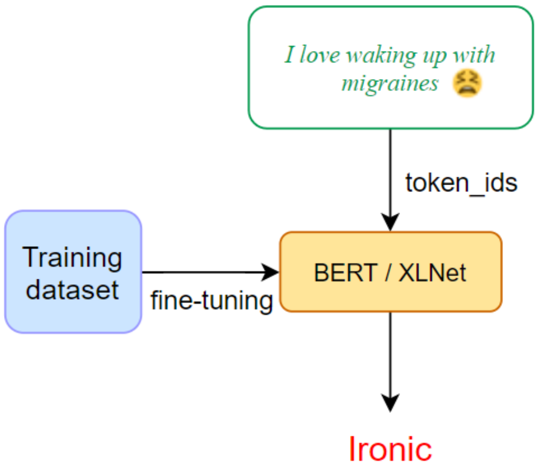

Figure 2: Fine-tuning Method Structure

5. Experiments and Results

In this section, we explain the experiments and show the results.

5.1. Data Preprocessing

Before the experiment, we should make some data preprocessing on the tweets since tweets are unstructured data and have many words that machine cannot recognize.

1. We discard all links in the tweets since the links are almost YouTube short urls. These urls do not contain any useful information.
2. Each emoji is replaced with their meaning by emoji lists[1] because emojis play a significant role in expressing the hidden feeling that cannot observed in tweets. For example, the meaning of 😀 is the grinning face.
3. Hashtags are split into the readable way because hashtags are also important for the text classification since users use hashtag to indicate their topics and events. For example, "#SadButTrue" should be split into "# Sad But True". Here, we calculate the probability based on the Brown corpus [20] to implement this idea.

[1] https://unicode.org/emoji/charts/full-emoji-list.html

5.2. Experiment Setting

BERT has many pre-trained models. Here we only use bert-base-uncased, bert-base-cased, bert-large-uncased and bert-large-cased. For XLNet, we use xlnet-base-cased and xlnet-large-cased. The base means the model contains 12 layers and 110M parameters while the large model contains 24 layers and 340M parameters. Uncased means that the pretrained was done on lower-cased English text while cased means that the pretrained was done on normal English text.

Also for the word embedding method, we tuned hyperparameters of traditional machine learning models for the better results. Here we use gridsearchcv function which will make the cross-validation automatically in scikit-learn to find best hyperparameters. The only thing we should do is to set several appropriate hyperparameters. For Random Forest, we tuned the number of estimators and max depth. For Logistic Regression, we tuned solver and C. For Support Vector Machine, we tuned C, gamma.

5.3. Experiment results for Task A

The result for task A based on word embedding method is shown in Table 4. Here we just show the F_1 score. RF, NB, LR, SVM are the traditional machine learning classification modes: Random Forest, Naïve Bayesian, Logistic Regression and Support Vector Machine. The first column is the abbreviation of model name. B, X, b, l, u and c correspond to BERT, XLNet, base, large, cased, uncased. The best score for each traditional machine learning method is bold in the table.

As observed in Table 4, SVM classification algorithm performs well with BERT. The best F_1 score for SVM is 0.693. This score can rank 3rd compared with the past competition results. It seems like that there is a problem in the combination of XLNet and SVM (with *). This combination got the 0.048 and 0.062 which are almost near to 0. The reason is that the SVM model will classify all samples into the negative so that the recalls are near to 0. Generally, we could say that in task A, SVM > LR > RF > NB.

Table 4. Word Embedding Results for Task A (F_1 score)

Model	RF	NB	LR	SVM
B-b-u	**0.586**	**0.455**	0.650	0.690
B-b-c	0.516	0.310	0.614	0.626
B-l-u	0.517	0.333	0.602	0.650
B-l-c	0.531	0.397	**0.655**	**0.693**
X-b-c	0.498	0.359	0.596	0.048*
X-l-c	0.566	0.282	0.594	0.062*

The F_1 score result for task A based on fine-tuning method is shown in Table 5. From the table, we can infer that the smaller batch size and the bigger epochs we set, the better performance we get. There are some scores that need attention. The symbol "-" in the table means that the large XLNet model is too large to train on a single 32GB GPU. Therefore, we do not record their score. The 0* here shows that large XLNet model sometimes will classify all samples into the negative after fine-tuning. The precision and recall score are all zero, so the F_1 score is zero. Moreover, the score of large XLNet model decrease significantly compared with the base XLNet model. The best F_1 score 0.704 is achieved by xlnet-base-cased model training with four epochs and 16 batch size. Compared with the top 5 teams' results, 0.704 can also rank 3rd and better than the best score 0.693 based on word embedding method.

Table 5. Fine-Tuning Results for Task A(F_1 *score*)

Model	E=3	E=4	E=4	E=4
	bs=64	bs=64	bs=32	bs=16
B-b-u	0.629	0.628	0.660	0.661
B-b-c	0.640	0.647	0.672	0.691
B-l-u	0.640	0.658	0.663	0.697
B-l-c	0.649	0.638	0.693	0.700
X-b-c	0.645	0.679	0.669	**0.704**
X-l-c	-	-	0*	0.565

E: epochs, bs: batch size

5.4. Experiment results for Task B

The result for Task B based on word embedding method is shown in Table 6. For task B, the LR with bert-large-uncased got the best F_1 score 0.423. But this score is not very good compared with the past competition results. Generally, we could say that in task B, LR $\approx$ SVM > NB > RF. Same with result of Task A, there is a problem in the combination of SVM and XLNet.

Table 6. Word Embedding Results for Task B(F_1 *score*)

Model	RF	NB	LR	SVM
B-b-u	**0.290**	**0.328**	0.402	0.406
B-b-c	0.266	0.307	0.369	0.355
B-l-u	0.280	0.254	**0.423**	0.395
B-l-c	0.264	0.306	0.392	**0.421**
X-b-c	0.273	0.315	0.382	0.188
X-l-c	0.234	0.157	0.345	0.188

The result for Task A based on fine-tuning method is shown in Table 7. The "-" mean parallel training too. From this table, we know that base XLNet also gets the best F_1 score 0.489 in task B. And the best F_1 score for BERT model is 0.445. The score of large XLNet model trained with four epochs and 32 batch size is still much lower than the base XLNet. But fortunately it is not zero again.

Table 7. Fine-Tuning Results for Task B(F_1 *score*)

Model	E=3	E=4	E=4	E=4
	bs=64	bs=64	bs=32	bs=16
B-b-u	0.360	0.385	0.419	0.426
B-b-c	0.335	0.339	0.338	0.441
B-l-u	0.338	0.349	**0.445**	0.431
B-l-c	0.392	0.419	0.431	0.410
X-b-c	0.413	0.461	**0.489**	0.436
X-l-c	-	-	0.188	0.380

We should also take a closer look at performance on each category of irony in task B. The top five teams' performance for each class is shown in Table 8[10]. V, S and O correspond to verbal irony, situational irony and other verbal irony. As can be inferred from Table 8, all teams got a higher score on *non-ironic* and *verbal ironic*, but performs not well on the *situational irony* and *other verbal irony*. The score of *other verbal irony* is the lowest. The reason might be that the other category contains diverse types of irony. More detailed insights should be provided about this category.

Table 8. F_1 score of each class for Task B[10]

Team	Non-ironic	V	S	O
UCDCC	**0.843**	**0.697**	0.376	0.114
NTUA-SLP	0.742	0.648	**0.460**	0.133
THU_NGN	0.704	0.608	0.433	**0.233**
NLPRL-IITBHU	0.689	0.636	0.387	0.185
NIHRIO	0.763	0.607	0.317	0.087

Then we look at the systems which get best F_1 score, including bert-large-uncased + Logistic Regression, best fine-tuning BERT (bert-large-uncased, 4 epochs, 32 batch size) and best fine-tuning XLNet (xlnet-base-cased, 4 epochs, 32 batch size). The result of them for each category is shown in Table 9. The best XLNet outperforms word embedding system and best BERT by a sizable margin. One more thing is that BERT and XLNet cannot recognize *other verbal irony* neither while the traditional machine learning models can recognize it even the score is very low. Compared with the scores of top five teams, the best XLNet model performs better on *the situational irony* while in other categories the performance is not satisfying.

Table 9. F_1 score of best three systems for each class for Task B

Model	Non-ironic	V	S	O
B-l-u+LR	0.727	0.478	0.406	0.081
BERT-best	0.665	0.639	0.382	0
XLNet-best	0.771	0.650	0.534	0

6. Conclusion

In this paper, we evaluate the performance of BERT and XLNet models based on word embedding method and fine-tuning method on shared task "Irony Detection in English Tweets". These two models could get relatively high scores showing that BERT and XLNet models are capable to understand the irony to some extent. But they still cannot defeat the 1st or 2nd team which used LSTMs. This can probably be explained by that irony is a high-level wisdom of humankinds and is infrequently used which seldom appears in the Wikipedia on which these two models were pretrained. Moreover, the top 1 team handled some strange words more carefully and used the data augmentation. Therefore, they got higher score for this specific task. The base XLNet model got the best score for both tasks showing that XLNet does outperform BERT in understanding human language, but the large XLNet models seem to instable during the training process. One more thing is that fine-tuning method outperforms the word embedding method in both tasks. For the future work, a detailed research should be done on the *other verbal irony* as well as the data augmentation. Moreover, some tweets need more preprocessing like translating the abbreviation or removing repeated words like "looooooong".

References

[1] A. Ghosh, G. Li, T. Veale, P. Rosso, E. Shutova, J. Barnden, and A. Reyes, "Semeval-2015 task 11: sentiment analysis of figurative language in twitter," Proc. of the 9th International Workshop on Semantic Evaluation, pp.470-478, June. 2015.

[2] G. Lakoff, The contemporary theory of metaphor, Cambridge University Press. 1993.

[3] A. Ghosh, and T. Veale, "Fracking sarcasm using neural network," Proc. of 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pp.161-169, June. 2016.

[4] C. Van Hee, E. Lefever, and V. Hoste, "Monday mornings are my fave:)# not exploring the automatic recognition of irony in english tweets," Proc. of the 26th Int'l Conf. on Computational Linguistic., pp.2730-2739, Dec. 2016.

[5] S. Rosenthal, N. Farra, and P. Nakov, "SemEval-2017 task 4: sentiment analysis in twitter," arXiv preprint arXiv:1912.00741. 2019.

[6] A. Joshi, P. Bhattacharyya, and M.J. Carman, "Automatic sarcasm detection: a survey," ACM Computing Surveys (CSUR), 50(5), 73, 2017.

[7] A. Khattri, A. Joshi, P. Bhattacharyya, and M. Carman, "Your sentiment precedes you: using an author's historical tweets to predict sarcasm." Proc. of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pp.25-30, Sept. 2015.

[8] D.G. Maynard, and M.A. Greenwood, "Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis," Proc. of the 9th Int'l Conf. on Language Resources and Evaluation (LREC'14), pp.4238-4243, Mar. 2014.

[9] N. Desai, and A.D. Dave, "Sarcasm detection in hindi sentences using support vector machine," International Journal, 4(7), 8-15, 2016

[10] C. Van Hee, E. Lefever, and V. Hoste, "Semeval-2018 task 3: irony detection in english tweets," Proc. 12th International Workshop on Semantic Evaluation, pp.39-50, June. 2018

[11] A. Ghosh, and T. Veale, "Ironymagnet at semeval-2018 task 3: a siamese network for irony detection in social media," Proc. of the 12th Int'l Workshop on Semantic Evaluation, pp. 570-575, June. 2018.

[12] C. Wu, F. Wu, S. Wu, J. Liu, Z. Yuan, and Y. Huang, "Thu_ngn at semeval-2018 task 3: tweet irony detection with densely connected lstm and multi-task learning," Proc. 12th Int'l Workshop on Semantic Evaluation, pp. 51-56, June, 2018.

[13] C. Baziotis, N. Athanasiou, P. Papalampidi, A.

Kolovou, G. Paraskevopoulos, N. Ellinas, and Potamianos, "Ntua-slp at semeval-2018 task 3: tracking ironic tweets using ensembles of word and character level attentive rnns," arXiv preprint arXiv:1804.06659, June. 2018.

[14] O. Rohanian, S. Taslimipoor, R. Evans, and R. Mitkov, "Wlv at semeval-2018 task 3: dissecting tweets in search of irony." Proc. of the 12th Int'l Workshop on Semantic Evaluation, pp. 553-559, June. 2018.

[15] H. Rangwani, D. Kulshreshtha, and A.K. Singh, "Nlprl-iitbhu at semeval-2018 task 3: combining linguistic features and emoji pre-trained cnn for irony detection in tweets," Proc. of the 12th Int'l Workshop on Semantic Evaluation, pp.638-642, June. 2018.

[16] J. Devlin, M.W. Chang, K. Lee, and K. Toutanova, "Bert: pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[17] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S.R. Bowman, "Glue: a multi-task benchmark and analysis platform for natural language understanding," arXiv preprint arXiv:1804.07461. 2018.

[18] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," arXiv preprint arXiv:1606.05250. 2016.

[19] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q.V. Le, "Xlnet: generalized autoregressive pretraining for language understanding," arXiv preprint arXiv:1906.08237. 2019.

[20] W.N. Francis, and H. Kucera, "Brown corpus manual: manual of information to accompany a standard corpus of present-day edited american english for use with digital computers." Brown University, Providence, Rhode Island, USA. 1979.