Capstone Project

Haitao Cai

August 7, 2019

1 Definition

1.1 Project Overview

Prescription medication abuse (for short, PMA) has become a severe nationwide issue of the United States, an extreme consequence of which is death. For instance, the number of drug abuse deaths in 2017 was 70,237, according to a report published by the Centers for Disease Control and Prevention [6]. However, most of the established methods for monitoring only apply to licensed practitioners and distributors. As a consequence, it is difficult to obtain crucial information regarding the abusers.

To complement the existing methods, more sources of data have been explored, such as social media. Particularly, it is reported that abundant positive samples of prescription medication abuse were systematically detected from Twitter [14].

In order to further facilitate the detection of self-reported PMA events on Twitter and other social media sites, I developed a more accurate and more robust classifier for social media text data.

1.2 Problem Statement

Since numerous articles and comments are posted on social media every minute, the scope of detection was narrowed down to those texts which mention a medication name. Nonetheless, the positive samples among the medication-mentioning social media texts are also rather sparse. For example, the data mined from Twitter using medication names (used in this project) only contain $\approx 16\%$ of positive samples. It has been long noticed that the imbalance often poses extra challenges for learning [5]. As an illustration, the F1 score reported in [14] is 0.46 for the positive PMA class, though an overall accuracy of 82% is obtained.

To develop a better performing PMA classifier (as is measured by the metrics defined in Section 1.3), I make use of the increasingly popular language representation model BERT proposed by Google Research [3]. Mainly, the classifier makes predictions by taking as input the language representation generated by pre-trained BERT models. In addition, I will experiment with various neural network layers and techniques such as Dropout for better generalizability of the classifier [15].

1.3 Metrics

Apart from accuracy (i.e., the percentage of correctly predicted samples), the classifier built in this project was evaluated using precision, recall and F1-score. All of the metrics are standard in the evaluation of text classifiers.

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

$$\text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

$$\text{F1} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Intuitively, precision measures the relative frequency of false positive predictions, whereas recall measures that of false negative predictions. Furthermore, F1-score provides a collective representation of precision and recall. Using the three metrics can avoid improper evaluation of a classifier which blindly predicts every sample to belong to the majority class for an imbalanced dataset. Given the inherent imbalance of social media data of PMA, the three metrics are appropriate for evaluating the classifier developed in this project.

2 Analysis

2.1 Data Exploration

The data used in this project consists of 16379 tweets fetched from the publicly accessible Twitter API by specifying a list of keywords. Specifically, each tweet mentions at least one from a list of target medication names (e.g., *adderall*, *ativan*).

Two annotators manually labeled the tweets with four labels [11]. The labels and respective samples are as follows. As can be seen in the sample for class ‘a’, the indication of PMA events can be far from straightforward.

– Potential Abuse/Misuse (‘a’): *you don’t know what melting feels like until you’re incredibly stoned on weed and hydrocodone in 100° weather*;
– Non-abuse Consumption (‘c’): *Replacing Xanax with Benadryl 2018*;
– Drug Mention Only (‘m’): *What is Xanax. Like do we call it something else here or just not have it*;
– Unrelated (‘u’): *i listen to feenin by lyrica like everyday.*

The tweets were randomly split into a training set, a validation set and a testing set. The count of each label for the datasets is listed in Table 1. The benchmark model used in this project has been trained and evaluated on the same datasets. Particularly, the positive PMA class ‘a’ only makes up about 16% of the entire dataset.

In addition, the text lengths (i.e., number of tokens) of the training data are summarized in Table 2.

	‘a’	‘c’	‘m’	‘u’	Total
Train	1869	3214	5914	475	11472
Validation	260	449	874	53	1636
Test	503	919	1722	127	3271

Table 1: Counts of samples by Class

maximum	median	mean
109	16	18.73

Table 2: Summary of token counts

2.2 Exploratory Visualization

Echoing the counts presented in Section 2.1, a plot of the sample proportions by group and class is as Figure 1. Although the positive PMA class ‘a’ is not the smallest class among the four, it contains much fewer samples than ‘c’ and ‘m’.

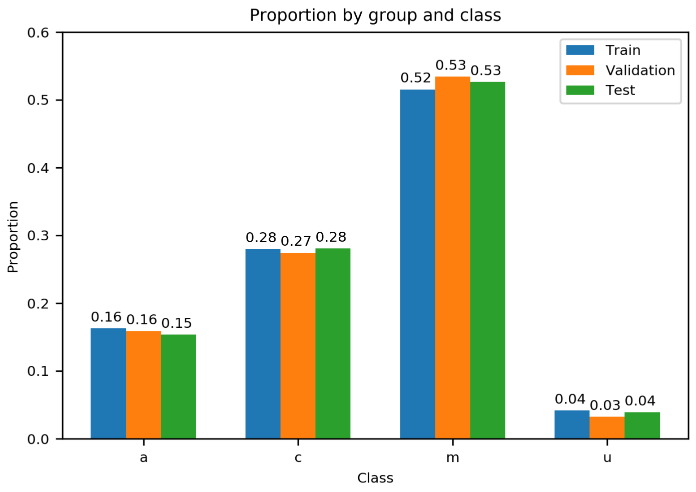

Figure 1: Counts of samples by group and class

2.3 Algorithms and Techniques

The PMA classifier is developed based on one of the most successful language representation models, namely, **B**idirectional **E**ncoder **R**epresentations from **T**ransformers (for short, **BERT**). BERT is chosen for this project for at least the following reasons.

- ▷ The detection of such low-proportion events relies on a better machine-understanding of language meaning. BERT is essentially a language representation model[3], a core constituent of which is semantic meaning.
- ▷ BERT-based models have achieved new highest scores in many natural language processing tasks (e.g., GLUE, SWAG) [3].

BERT provides a language model by pre-training on two tasks. Namely, masked language model and next sentence prediction [3]. Particularly, it exploits the contextual information in both the left-to-right and the right-to-left directions. This strategy is in line with the intuition that the interpretation of words or phrases not only depends on the preceding context, but also the following context.

Specifically, BERT takes into account the vectorized representation (i.e., embedding) of both words and the positions of words. The output of the embedding layers is further encoded by the Transformer initially proposed by [17]. The BERT-representation of text data can be fed to simple neural network layers such as feed-forward Dense layers for tasks such as text classification.

Moreover, the BERT-based PMA classifier involves the following parameters.

- ▷ BERT parameters.
  - ⋆ Maximal sequence length (maximal number of tokens to encode).
- ▷ Classifier parameters.
  - ⋆ Types of layers (e.g., feed-forward Dense, Convolutional, Pooling).
  - ⋆ Number of layers.
  - ⋆ Size of each layer.
  - ⋆ Rate of dropping out.
- ▷ Training parameters.
  - ⋆ Learning rate.
  - ⋆ Batch size of training data.

2.4 Benchmark

Currently, there are very few existing studies of this task. After experimenting with a large variety of algorithms, a Support Vector Machine based model obtained the highest scores[1]. The model makes use of the RBF kernel and following features.

- Counts of words.
- Counts of word clusters [12].

After being fitted on the training dataset, it achieves the following scores on the testing data (Table 3).

Precision				Recall				F1-Score				Accuracy
'a'	'c'	'm'	'u'	'a'	'c'	'm'	'u'	'a'	'c'	'm'	'u'	
0.466	0.766	0.806	0.817	0.602	0.601	0.836	0.740	0.526	0.673	0.821	0.777	0.730

Table 3: Scores of the benchmark model

Notably, the benchmark SVM classifier outperforms the models built on top of word2vec and Convolutional Neural Network, which made many breakthroughs in the domain of NLP and especially text classification in the past years. In addition, the scores are also significantly better than those obtained by SVM with linear kernel[2].

[1]The experiments were conducted by Abeed Sarker and Yucheng Ruan for the NIH project 1R01DA046619-01

[2]Following the suggestion from the reviewer of my Capstone proposal, an additional experiment was conducted to train and test SVM with linear kernel.

3 Methodology

3.1 Data Preprocessing

For each sentence or sequence of words, BERT generates a two-dimensional matrix representation (part of) which can be passed to successive neural network layers for specific tasks. Specifically, the input is a sequence of tokens which are represented as corresponding integer ids. Therefore, the text data of tweets undergo the following steps of preprocessing.

- ⋆ Case normalization. All characters are uniformly converted to the lower case, since there is no motivation to treat words in different case forms as different tokens.
- ⋆ Masking mentions of Twitter user names, which may contain noisy terms. For instance, some user names may happen to contain words indicative of PMA events (e.g., *faint*).
- ⋆ Removal of punctuation, which is not included as part of the BERT input.
- ⋆ Tokenization of text, producing the units of input.
- ⋆ Mapping tokens to integer ids, which are the direct input of BERT.

Notably, no feature selection is needed, since the input of BERT consists of nothing else than the sequence of tokens.

3.2 Implementation

The implementation of the PMA classifier essentially makes use of Keras and keras-bert. The experiments are run on Google Colab using the free instance of Nvidia Tesla T4 GPU. Due to the limit of GPU memory (16 GB), one of the small pre-trained BERT models is employed in the experiments for the PMA classifier. Essentially, it contains the following components. In total, the number of parameters for the pre-trained BERT model is around 110M.

- 12 Transformer blocks;
- 768 units for each hidden layer;
- 12 self-attention heads.

In addition, based on the summary in Table 2, the (maximal) sequence length is set to 100.

In the step of tokenization, a special token `[CLS]` is prepended to the beginning of the output sequence. The final encoding generated for this token by the Transformers serves as the aggregate representation of the entire sequence in classification tasks [3]. Specifically, the encoding corresponding to `[CLS]` is passed to the layers appended to BERT for classification.

The initial implementation of the classifier follows the description of a minimal architecture presented by [3]. Namely, after feeding the sequence of token ids to BERT, the representation encoded in `[CLS]` is passed to a softmax layer to predict the class for each sample, which is only intermediated by a Dropout layer. In other words, the initial implementation of the classifier only involves the following two layers beyond the pre-trained BERT.

- A Dropout layer with dropping rate 0.1;
- A Dense layer with 4 units and softmax activation.

Moreover, the training employs the Adam Optimization with rate decay and warm-up [9]. Specifically, the parameters for training are as follows.

– Batch size of training data is 32.
– The training runs for 3 epochs[3].
– The maximal learning rate (lr) is 1e-4.
– In the first 10% of training steps, the learning rate grows from 0 to lr with a constant step size.
– In the later 90% of training steps, the learning rate decays to 0.

At the end of each epoch, the accuracy of prediction on the validation dataset is calculated.

Overall, the initial implementation is straightforward. The only complication occurring during the implementation is to avoid excessive usage of GPU RAM (if GPU is used for training). For instance, for the free instance of Google Colab Tesla GPU, when the training data batch size is set to 64, there will be warning that the GPU RAM usage is reaching the limit.

3.3 Refinement

The training and validation loss for each epoch was plotted in Figure 2.

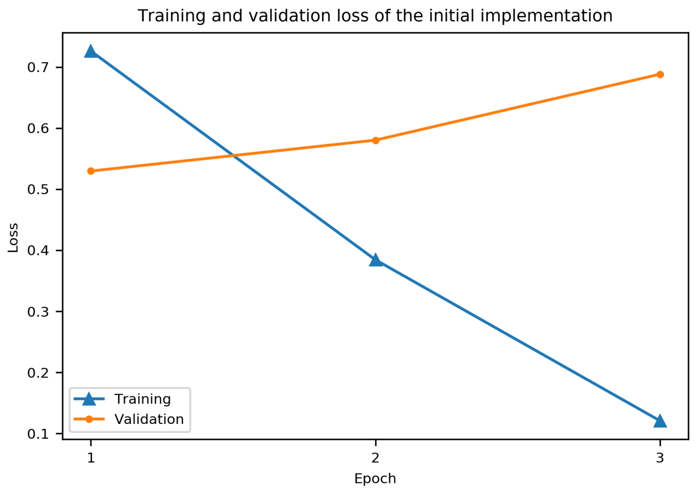

Figure 2: Training and validation loss of the initial implementation by epoch

In order to improve the performance of the BERT-based PMA classifier, the following strategies for refinement were implemented.

▷ Larger values for Dropout were tested: 0.2, 0.3, 0.4, 0.5, due to the apparent over-fitting (Figure 2).
▷ Also for the purpose of reducing over-fitting, Batch Normalization[4] was employed in conjunction with Dropout [7, 1]. Specifically, following the proposal justified by

[3]Again, this setting follows the description documented by [3]
[4]Thanks to the reviewer of my Capstone proposal for drawing my attention to this technique.

theoretical and experimental analyses presented by [2, 10], Batch Normalization was employed before the Dropout layer, with the default values for momentum (0.99) and epsilon (0.001) in Keras[5].

▷ A feed-forward Dense layer of size 32 and with Relu activation is inserted immediately after the pre-trained BERT model. L1 was applied to the output of the layer (i.e., `activity_regularizer`), whereas L2 was applied to the weight matrix (i.e., `kernel_regularizer`). The tested values for the regularizers include: 0.001, 0.002, 0.003, 0.004, 0.005.

▷ Convolutional neural network (CNN) has produced remarkable results in text classification when implemented on top of word embeddings [8]. Therefore, a 1D-convolutional[6] layer was added as the first processor of the output of the BERT-model, which is followed by a 1D Max Pooling layer. The values of parameters are as follows.

  ⋆ Number of filters: 16.
  ⋆ Kernel size: 5.
  ⋆ Pool size: 4.

▷ Figure 2 indicates that generalization error started to increase in epoch 3. Therefore, another strategy for refinement was to stop the training after 2 epochs.

4 Results

4.1 Model Evaluation and Validation

The final model is produced by the initial implementation. The architecture and the training parameters are described in Section 3.2. Although the final model is simple, the architecture and the training parameters mostly follow the description for BERT-based text classifiers made by the authors of BERT [3], which have performed remarkably well on many other text classification tasks. In addition, the scores of the PMA classifier on the test dataset are listed in Table 4.

Precision				Recall				F1-Score				Accuracy
‘a’	‘c’	‘m’	‘u’	‘a’	‘c’	‘m’	‘u’	‘a’	‘c’	‘m’	‘u’	
0.588	0.776	0.857	0.922	0.618	0.775	0.850	0.843	0.603	0.775	0.854	0.881	0.793

Table 4: Test scores of the BERT-based final model

The comparison of test scores between the BERT-based model and the benchmark model is plotted in Figure 3. Obviously, the former outperformed the latter on each of the three metrics and each of the four classes. Particularly, for the class of the most interest ‘a’, the improvement in precision and therefore f1-score is salient. Therefore, it is reasonable to state that the model meets the expectation of significantly boosting the quality of PMA classification.

Moreover, the results echo the BERT authors’ claim that BERT performs well in a large variety of tasks without much task-specific architecture. Still, it is surprising that none of the strategies for refinement described in Section 3.3 can significantly improve the performance of the classifier. It may to a large extent be due to the limited size of the training data and the inherent difficulty of this classification task (e.g., the samples listed in Section 2.1).

[5] `https://keras.io/layers/normalization/`

[6] The BERT-representation of the `[CLS]` token for each sequence of words is a 1D vector of length 768.

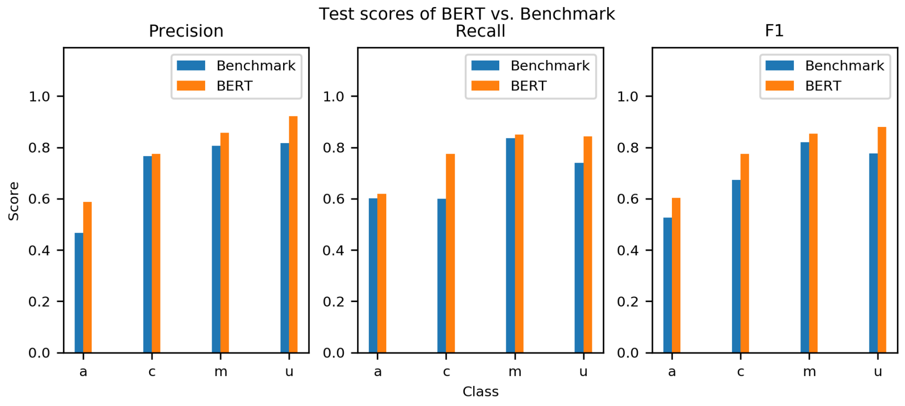

Figure 3: Comparison of test scores

In order to verify the robustness of the model, it was also trained on the training data with 10% of data removed. The results are listed in Table 5. Comparing with the scores obtained after training on the full training dataset, only minor drop of performance is spotted (e.g., F1 for 'a', 0.584 vs. 0.603)

Precision				Recall				F1-Score				Accuracy
'a'	'c'	'm'	'u'	'a'	'c'	'm'	'u'	'a'	'c'	'm'	'u'	
0.612	0.764	0.839	0.889	0.559	0.767	0.863	0.819	0.584	0.765	0.851	0.852	0.788

Table 5: Test scores after training on only 90% of the data

To sum up, it can be concluded that the BERT-based PMA classifier is both well-performing and robust.

4.2 Justification

Apart from the apparent improvement of testing scores obtained by the BERT-based model, it can also be statistically demonstrated that the improvement over the benchmark model is significant. Mainly, the results of McNemar's Test ($\chi^2 = 252.0$, $p < 0.001$) indicate that the two models not only make errors on different samples, but also make a different proportion of errors in testing [4]. Hence, after training on the same set of data, the two models have different performance on the testing dataset.

5 Conclusion

5.1 Free-Form Visualization

For medication names mentioned in at least 70 tweets among the testing samples, the prediction accuracy is plotted in Figure 4. It can be seen that accuracy displays non-trivial variation across the medications. For instance, adderall-mentioning tweets is only predicted with an accuracy of 0.72, whereas lyrica and methadone reach 0.90. This observation may indicate that training samples mentioning different drugs need to have different weights in training.

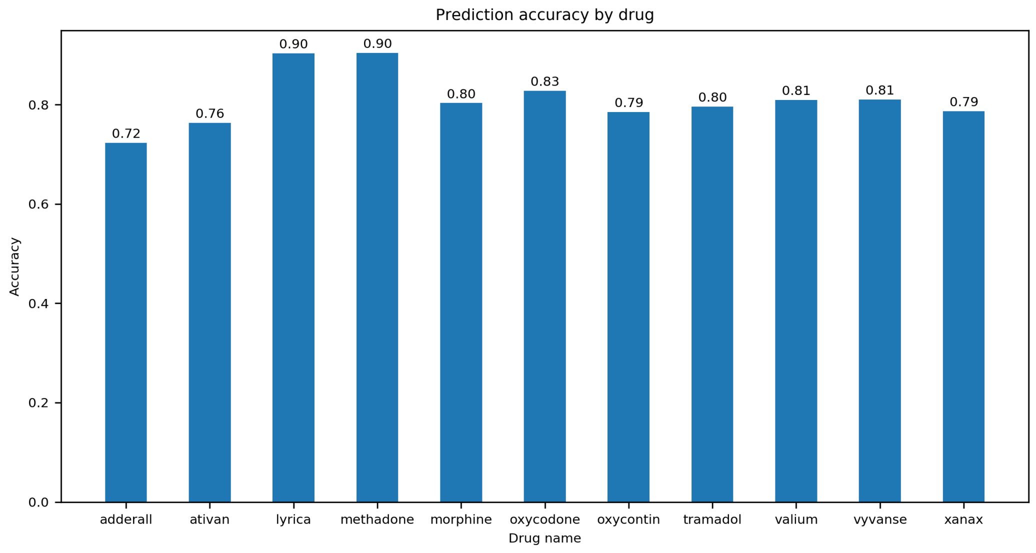

Figure 4: Testing accuracy of drugs with count of mention >= 70

5.2 Reflection

The process for this project can be summarized as the following steps (after selecting this task).

1. The annotated data and the predictions of the benchmark model were downloaded / assembled.
2. I searched for existing models for language representation and text classification. BERT was found and I worked through the source code.
3. The strategy for preprocessing was decided and implemented.
4. The initial implementation of the BERT-based PMA classifier was completed and tested against the testing dataset.
5. A variety of techniques (e.g., Batch Normalization, CNN) were implemented to further improve the performance of the classifier.
6. The robustness of the model was verified by training on 90% of the data.
7. The BERT-based classifier was compared with the benchmark model. The difference in performance was statistically demonstrated.

It is interesting to see how well the initially implemented classifier performs on the testing dataset. Only a Dropout layer and a softmax prediction layer were appended to the pre-trained BERT model.

As for the most difficult aspect of this project, it is rather time-consuming to search for and test the various strategies for further refinement, which also involved extensive literature search. At the same time, I enjoyed reading the articles and learned a lot in this process.

Overall, it is pleasant that a well-performing and robust PMA classifier was developed. I can confidently state that the finally chosen model and solution fit my expectation for this project. Moreover, the same workflow should be applicable to similar types of problems in a general setting, since the reliability of the outcome is verified by close scrutiny of the results.

5.3 Improvement

The solution can potentially be improved in the following ways, which may obtain better performance.

- ▷ The pre-trained BERT model can be further fine-tuned in multiple NLP tasks [16].
- ▷ XLNet was proposed as an improved language representation model by taking into account the pros and cons of BERT [18], which may take the place of BERT in the solution.
- ▷ If more RAM of GPU is available, larger pre-trained BERT models can be employed for the classifier.
- ▷ Sampling strategies can be used to solve the difficulty caused by the imbalance of the datasets [13].

References

[1] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In *Advances in Neural Information Processing Systems*, pages 7694–7705, 2018.

[2] Guangyong Chen, Pengfei Chen, Yujun Shi, Chang-Yu Hsieh, Benben Liao, and Shengyu Zhang. Rethinking the usage of batch normalization and dropout in the training of deep neural networks. *arXiv preprint arXiv:1905.05928*, 2019.

[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] Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[5] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9):1263–1284, 2008.

[6] Holly Hedegaard, Arialdi M Miniño, and Margaret Warner. Drug overdose deaths in the united states, 1999–2017. 2018.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[8] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] Xiang Li, Shuo Chen, Xiaolin Hu, and Jian Yang. Understanding the disharmony between dropout and batch normalization by variance shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2682–2690, 2019.

[11] Karen O'Connor, Annika DeRoos, Alexis Upshur, Graciela Gonzalez-Hernandez, Jeanmarie Perrone, and Abeed Sarker. Toxicovigilance from social media: Annotation guidelines. Technical report, University of Pennsylvania, 2018.

[12] Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, and Nathan Schneider. Part-of-speech tagging for twitter: Word clusters and other advances. *School of Computer Science*, 2012.

[13] Foster Provost. Machine learning from imbalanced data sets 101. In *Proceedings of the AAAI'2000 workshop on imbalanced data sets*, volume 68, pages 1–3. AAAI Press, 2000.

[14] Abeed Sarker, Karen O'Connor, Rachel Ginn, Matthew Scotch, Karen Smith, Dan Malone, and Graciela Gonzalez. Social media mining for toxicovigilance: automatic monitoring of prescription medication abuse from twitter. *Drug safety*, 39(3):231–240, 2016.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[16] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? *arXiv preprint arXiv:1905.05583*, 2019.

[17] 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.

[18] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.