Text Classification Competition: Twitter Sarcasm Detection

Oran Chan (wlchan2) and Edward Ma (kcma2)

ABSTRACT

Sarcasm detection is a specific case of sentiment analysis. Instead of classifying all kinds of motion, this task only focuses on sarcasm. Given a text as input, the detection model outputs whether it is sarcastic or not. The most challenging part is that judgement of sarcasm detection is not very clearly defined or it is subjective. In this classification competition, we suggest using less human and computer resources to achieve a better result. We demonstrate how synthetic data helps to boost up model performance with manual effort.

1. INTRODUCTION

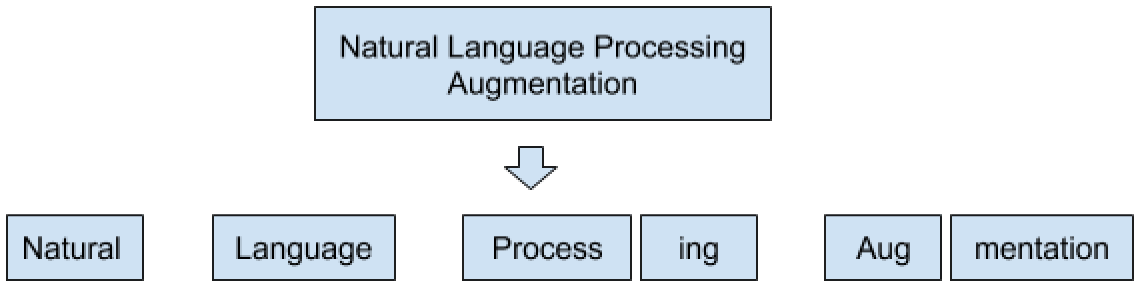
In this classification competition, the training data size is 5000 with equal distribution. This size is relatively small in natural language processing (NLP). Therefore, we proposed to use transfer learning and data augmentation strategy to tackle this problem. For the transfer learning part, we will train our model based on a pre-trained model which was trained on a very large corpus to solve some basic NLP tasks. It is a promised way to start with them instead of training from scratch. It does not only provide a converge word embeddings but also shortens training time. For the data augmentation part, we leverage contextual word embeddings training methods to generate synthetic data based on limited training data.

2. DATA PROCESSING

Given 5000 twitter text, we split it into training dataset and evaluation dataset with 9:1 ratio. In other words, the training dataset includes 4500 records while evaluation dataset includes 500 records.

2.1 Preprocessing

As mentioned before, we adopted a pre-trained model which can take care of lots of data processing. We do not need to do lots of feature engineering based on pure text but tokenizing text into subwords. Instead of using words as a feature, we decode to use subwords. For instance, “language” can be represented by “lang”, “uage” tokens. One of the major benefits is that it can handle out-of-vocabulary (OOV) problems. Giving that we use 26 (or 52 if case sensitive) characters, we can represent all English words. Another advantage is that it can converge rare word’s embeddings as we may break down a single rare word into multiple tokens. Also, subword algorithms leverage an affix behavior to further coverage subword embeddings. In English linguistics, an affix is a morpheme that is attached to a word stem to form a new word or word form. For example, a “dis” prefix means opposite while a “less” suffix means no. The following part covers Byte pair encoding (BPE) [1] and WordPiece [2] subword algorithm.



Example of Word Tokenization

2.2 BPE

BPE is proposed by Sennrich et al. (2016) and the general idea is counting the frequency of subwords up to a predefined maximum number of subwords. BPE is adopted by RoBERTa [5]. The algorithm is:

1. Prepare a large enough training data (i.e. corpus)
2. Define a desired subword vocabulary size
3. Split word to sequence of characters and append the suffix “</w>” to the end of word with word frequency. So the basic unit is character in this stage. For example, the frequency of “low” is 5, then we rephrase it to “l o w </w>”: 5
4. Generating a new subword according to the high frequency occurrence.
5. Repeating step 4 until reaching subword vocabulary size which is defined in step 2 or the next highest frequency pair is 1.

2.3 WordPiece

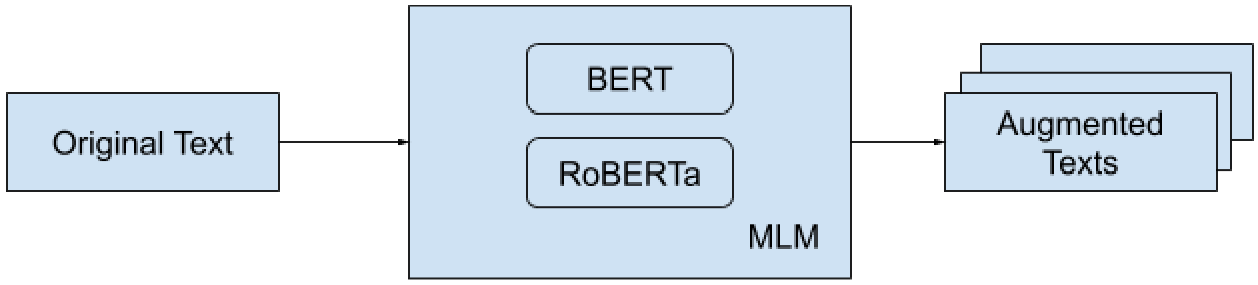
WordPiece is proposed by Schuster and Nakajima (2012). The idea is the same as BPE except the criteria of forming new subwords. New subwords will be formed based on likelihood but not the next highest frequency pair. WordPiece is adopted by BERT [6] The algorithm is:

1. Prepare a large enough training data (i.e. corpus)
2. Define a desired subword vocabulary size
3. Split word to sequence of characters
4. Build a languages model based on step 3 data
5. Choose the new word unit out of all the possible ones that increases the likelihood on the training data the most when added to the model.
6. Repeating step 5 until reaching subword vocabulary size which is defined in step 2 or the likelihood increases falls below a certain threshold.

2.4 Data Augmentation

There are lots of different ways to generate synthetic data. One of the typical ways is replacing words by synonyms [3] over NLTK [4] library. The limitation of using NLTK's synonyms is that it does not consider context. Considered the nature of sarcasm detection, we decided to levearge neural network models to find similar meaning words when considering context. The mechanism is picking a word randomly and using a neural work model to predict the possible word to replace it by providing whole content to the neural network models. We adopted BERT and RoBERTa neural network models to perform this data augmentation and the ration is 1:1. Here is the mechanism:

1. Pick a word from input
2. Replace the picked word by reserved token (e.g. [MASK])
3. Tokenize input with masked token
4. Feeding tokenize tokens to masked language model
5. Replace the picked word by language model prediction.



Flow of Augmentation

As nlpaug [7] implemented over 10 different data augmentation, we decided to leverage this library for data augmentation. In the evaluation, we tried different sizes of synthetic data to see how synthetic data affect the model performance. Also, we adopted two neural network models for comparison.

Type	**Content**
Original	@USER @USER Stephen Jones finally losing it on Twitter by claiming the Liberty is a great stadium . Never mind his thoughts on Sarries and the salary cap breach - this is a new low even for him . <URL>
Augmented Data #1	@USER @USER Stephen Jones finally losing it **at** Twitter **suddenly** claiming **Kings** Liberty is a great stadium . Never mind **after** thoughts **over** Sarries and the **transfer tax** breach - this is a **historic** low **from meeting** him . <URL>
Augmented Data #2	@user @user stephen jones finally losing it on twitter by claiming the **series** is its great **success** . never mind his thoughts on **soccer** but **high** salary cap **pro** - football is a **serious** low even for **david** . < url >

Example of augmented data

3. MODEL ARCHITECTURE

In this text classification, we evaluated both BERT and RoBERTa model and we finally picked RoBERTA as it outperforms BERT model based on our evaluation dataset.

3.1 BERT

BERT (Devlin et al., 2018) is a method of pre-training language representations, meaning that it was trained as a general-purpose “language understanding” model on a large text corpus (like Wikipedia), and adopting it for downstream NLP tasks that we care about. BERT outperforms previous methods because it is the first unsupervised, deeply bidirectional system for pre-training NLP.

BERT uses three embeddings to compute the input representations. They are token embeddings, segment embeddings and position embeddings. “CLS” is the reserved token to represent the start of a sequence while “SEP” is a separate segment (or sentence). Token embeddings are subword embeddings which represent the subword itself. Segment embeddings only include two embeddings which represent the first segment of input and second segment of input. Position embeddings refers to the position of the subword in the input. Segment embeddings help to distinguish two segments in some NLP downstream tasks such as question and answering tasks. For classification tasks, we only use a single segment. Position embeddings reflect the location as the same word may have different meanings in different positions of text.

For BERT’s training setup, it uses the Masked Language Model (MLM) and Next Sentence Prediction mechanism. By masking some tokens randomly, using other tokens to predict those masked tokens to learn the representations. For example, the original sentence is “I am learning NLP”. Assuming “NLP” is a selected token for masking. Then 80% of time, it will show as “I am learning [MASK]. For the Next Sentence Prediction approach, it targets to learn the relationship between sentences. The objective is classifying whether the second sentence is the next sentence or not.

3.2 RoBERTa

Liu et al. (2019) studied the impact of many key hyper-parameters and training data size of BERT. They found that BERT was significantly undertrained, and can match or exceed the performance of every model published after it. RoBERTa (Robustly optimized BERT approach) is introduced and performance is either matching or exceeding original BERT.

RoBERTa is developed based on BERT. By applying some modifications, it outperformed BERT model performance according to Liu et al (2019) experiments. First of all, it uses a larger training data. On the other hand, RoBERTA uses dynamic masking instead of static masking. Dynamic masking means that the masked token will be different every time.

4. EXPERIMENTS

In the experiments, we mainly compare two BERT and RoBERTa with different numbers of augmentation data. The size of original training record and evaluation record are 4500 and 500 respectively while size of augmentation data are various. Taking experiment #2 as an example, it used 9000 augmentation data and 4500 original data for training.

Instead of using a number of epochs for comparison, we use global steps. An epoch is one full pass through the training set, so that every sample gets the chance to be seen by model. Global steps refer to the number of batches seen by the model. For example, 1 epoch includes 4500 training records when it does not include any augmentation data while there are 13500 training records in 1 epoch when introducing 9000 augmentation data. Therefore, using global steps for comparison is a better approach.

4.1 BERT vs RoBERTa

This experiment aims at demonstrating the model performance between BERT and RoBERTa. RoBERTa shows it converges to minima faster than BERT. RoBERTa reaches its best F1 score at 6756 global steps while BERT reaches at 14000 global steps. Although BERT is sligher better than RoBERTa 0.004, we decided to adopt RoBERTa after balancing between computation and model performance.

Global Step	BERT			RoBERTa		
	Precision	Recall	F1	Precision	Recall	F1
1126	0.797	0.768	0.782	0.649	**0.960**	0.774
2815	0.741	**0.904**	0.814	0.776	0.764	0.770
6756	0.749	0.872	0.806	0.767	0.880	**0.819**
7319	0.796	0.748	0.771	**0.822**	0.756	0.787
11823	**0.816**	0.708	0.758	0.805	0.76	0.782
14000	0.777	0.876	**0.823**	0.771	0.860	0.813

Experiment Result #1

The following experiment is comparing model performances between BERT and RoBERTa when there are augmentation data. We noticed that RoBERTa outperforms BERT in 3 check global step checkpoints.

Global Step	BERT			RoBERTa		
	Precision	Recall	F1	Precision	Recall	F1

2000	0.711	0.876	0.785	0.736	0.880	0.801
4000	0.751	0.748	0.750	0.779	0.784	0.781
6000	**0.803**	0.768	0.785	0.762	**0.884**	**0.819**

Experiment Result #2

4.2 Data Augmentation

Besides model architecture, we want to see whether synthetic data helps to improve performance. Different sizes of augmentation data are adopted for comparisons. It includes 2250 (0.5 times of original dataset), 4500 (same size of original dataset), 9000 (2 times of original dataset) and 45000 (10 times of original dataset).

In this experiment, we want to evaluate whether augmentation data can boost up model performance. Due to the setup, experiment #1 does not include six thousandths global step so we use the nearest one which is 6756 global setup as reference. From the below figure, we can see that significant improvement when the size of augmentation data increased to 45000.

#	Augmentation Size	Global Step	Precision	Recall	F1
1	0	6756	0.767	0.880	0.819
2	2250	6000	0.805	0.840	0.822
3	4500	6000	0.782	0.804	0.793
4	9000	6000	0.762	**0.884**	0.819
5	45000	6000	**0.926**	0.848	**0.885**

Experiment Result #3

4.3 Final Submission

After conducting previous experiments, we decided to further train a RoBERTa with 45000 augmentation data for final submission. Between 0 and 10000 global steps, F1 score improved quickly. After that it converged slowly until 18000 global steps. After 18000 global steps, F1 score drops until 32000 global steps. Finally, the model only predicts either SARCASM or NOT_SARCASM for the whole validation dataset. Therefore, we picked the 18000 global step checkpoint as our final submission. Finally, our precision, recall and F1 are 0.687, 0.816 and 0.746 respectively.

#	Global Step	Precision	Recall	F1
1	2000	0.710	0.840	0.769
2	4000	0.801	0.852	0.826

3	6000	**0.926**	0.848	0.885
4	8000	0.924	0.976	0.949
5	10000	0.980	0.976	0.978
6	14000	0.992	0.968	0.980
7	16000	0.980	0.976	0.978
8	18000	0.980	**0.988**	**0.984**
9	20000	0.980	0.676	0.980
10	24000	0.984	0.976	0.980
11	30000	0.988	0.976	0.982
12	32000	0.98	0.980	0.980
13	34000	0	0	N/A
14	36000	0	0	N/A
15	136000	0	0	N/A

Experiment Result #4

4.4 Execution

We prepared a specific python file for each experiment. All of them will invoke the same python class with different parameters. The following table shows the mapping between python execution and experiment.

Python File	**Model**	**Augmentation Size**
run_bert_without_aug_epoch.py	BERT	0
run_bert_with_aug_9000.py	BERT	9000
run_roberta_without_aug_epoch.py	RoBERTa	0
run_roberta_with_aug_2250.py	RoBERTa	2250
run_roberta_with_aug_4500.py	RoBERTa	4500
run_roberta_with_aug_9000.py	RoBERTa	9000
run_roberta_with_aug_45000.py	RoBERTa	45000

To generate an answer for final submission, we use the following script.

Python File
prediction.py

4.5 Presentation

You may also check out this link (https://mediaspace.illinois.edu/media/t/1_fpgtkn68) for our video presentation.

5. FUTURE WORK

Besides data augmentation, we also brainstormed other ideas which include gathering more twitter data, evaluating other state-of-the-art models such as ELECTRA [8] and tuning hyperparameters.

6. REFERENCES

1. R. Sennrich, B. Haddow and A. Birch. Neural Machine Translation of Rare Words with Subword Units. 2015
2. M. Schuster and K. Nakajima. Japanese and Korea Voice Search. 2012
3. X. Zhang, J. Zhao and Y. LeCun. Character-level Convolutional Networks for Text Classification. 2015
4. E. Loper and S. Bird. NLTK: The Natural Language Toolkit. 2002
5. Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. 2019
6. J. Devlin, M. Chang, K. Lee and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018
7. E. Ma. NLP Augmentation. https://github.com/makcedward/nlpaug. 2019
8. C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M.Matena, Y. Zhou, W. Li and J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. 2019