Multilingual Twitter sentiment classification

Guangyue Qiu, Yanchong Peng, Zhenghang Yin, Yufan Lin, Juncheng Zhang, Guankai Sang
{gqiu,ycpeng,johnyin,eric1025,morganz,gs1084}@bu.edu

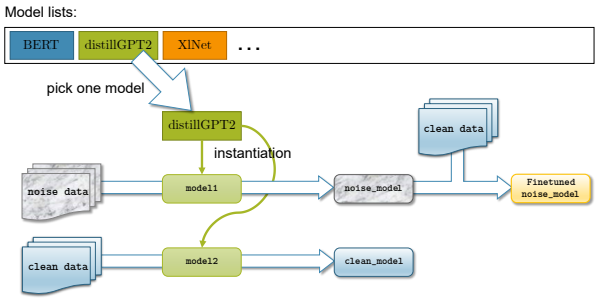

Figure 1 The training routine in the paper

1. Task

In recent years, natural language processing (NLP) is an increasingly hot research direction in the field of artificial intelligence. Twitter, as the world's largest social media platform, has attracted a wide range of interests from researchers. Hence, this work will focus on sentiment recognition in tweets. Sentiment recognition is a task of supervised learning, specifically, we train a function or a neural network to classify a series of sentences **X** into labels **y**, where **X** is tweets and y is sentiment label (might be binary where 0 as negative and 1 as positive, or three-category problems, depends on the problem definition). SemEval[1] competition defined a series of sentiment detection problems within tweets starts from 2013 till now, providing considerable sentiment training datasets of high-quality. Although many tasks have obtained achievable results, the existing models rely too much on human-annotated datasets, which are often exceedingly difficult to fetch and do not meet practical production needs. Therefore, this project focuses on investigating whether datasets automatically labeled by machines (referred as `noise_dataset` later) can be used in real production. This paper focuses on exploring two aspects:

1. We would evaluate the models that were currently widely used on Huggingface [2], including BERT, GPT and XLNet, train them using `noise_dataset` and human-annotated dataset respectively (referred as `clean_dataset` later), and evaluate the effects on the human-annotated test dataset to compare the models on both datasets.

2. Given that in production use, machine-generated datasets are more extensive and more readily available than manually labeled datasets. Therefore, we would like to explore whether we can fine-tune an original model, trained using `noise_dataset`, with a small amount of `clean_dataset` and if it will help to boost the performance.

2. Related Work

2.1. BERT

BERT [1] (Bidirectional Encoder Representations from Transformers) is a transformers model pretrained on a large corpus of English data in a self-supervised fashion. It uses a masked language modeling (MLM) objective and Next sentence prediction (NSP).

2.1.1. MLM

Before feeding sentences into BERT model, 15 percent of the words in the sentence will be replaced with a mask token. The model will be based on the other words in the sentence to predict the masked words. The process is shown in the Figure 3.

1. Adding a classification layer on top of the encoder output.

2. Multiplying the output vectors by the embedding matrix, transforming them into the

[1] https://semeval.github.io/

[2] https://huggingface.co/models

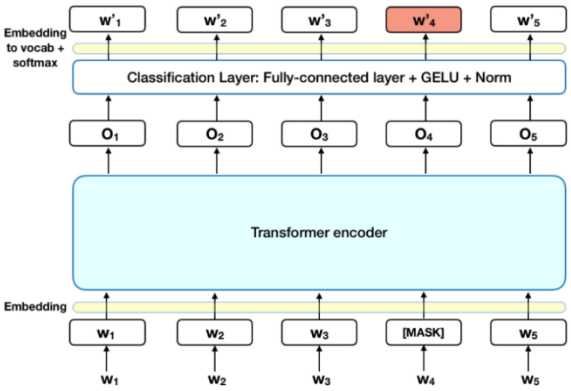

Figure 3 The process of BERT model's MLM

vocabulary dimension.

3. Calculating the probability of each word in the vocabulary with softmax.

2.1.2. NSP

BERT model receives the sentence as input and leans to predict if the sentence after the input sentence is the subsequence sentence in the original document. And the assumption is that the random sentence will be disconnected from the first sentence. The following steps are done before the input is fed into the BERT model as shown in Figure 2:

1. A `[CLS]` token is inserted at the beginning of the first sentence and a `[SEP]` token is inserted at the end of each sentence.

2. A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2.

3. A positional embedding is added to each token to indicate its position in the sequence. The concept and implementation of positional embedding are presented in the Transformer paper.

2.2. GPT2

GPT-2 is a large transformer-based language model, with generative pre-training of a language model on a diverse corpus of unlabeled text, followed by discriminative fine-tuning on each specific task. GPT has two major differences from ELMo:

1. The model architecture: ELMo uses the concatenation of forward and backward LSTMs, but GPT uses multi-layer transformers decoder.

2. Contextualized embedding: ELMo uses unsupervised Feature-based approach, while GPT fine-tunes the same base model for all end tasks.

2.3. XLNet

XLNet [2] is a generalized autoregressive pretraining method which uses the context word in two directions, both forward and backward. XLNet introduces a whole new technique called Permutation Language Modeling. It not only contains the advantage of AR models but also allows models to capture the advantage of bidirectional contexts.

Let Z_T be the set of all possible permutations of the length-T index sequence $[1, 2, \dots, T]$. We use Z_T and $z_{<t}$ to denote the t-th element and the first $t-1$ elements of a permutation $z \in Z_T$. Then, the permutation language modeling objective can be expressed as the following equation:

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}_T} \left[\sum_{t=1}^{T} \log p_\theta(x_{z_t} | \mathbf{x}_{\mathbf{z}<t}) \right]$$

An important thing is that the model only permutate factorization order and it keeps the original sequential order of the sentence the same and uses the positional encoding due to the original sequence and rely on a proper attention mask in Transformer to achieve permutation of

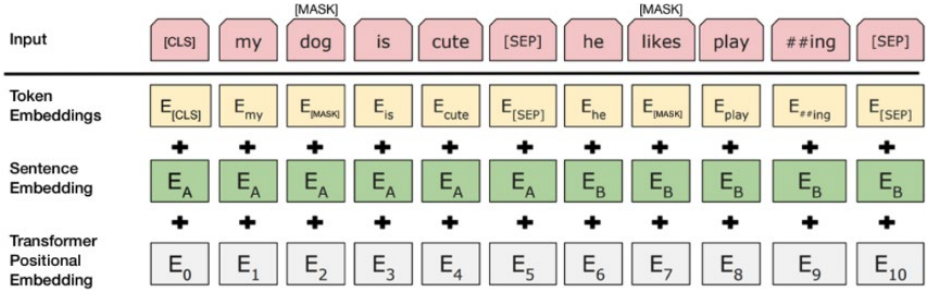

Figure 2 The process of BERT model's Next Sentence Prediction

	Model	Noise data	Clean data	Fine-tuned on clean data
Accuracy	**DistilGPT2**	74.9466%	**90.4762%**	**91.2439%**
Loss		67.7429%	23.4497%	22.5756%
Accuracy	**BERT_tiny**	**75.0729%**	86.7930%	87.0748%
Loss		50.6775%	32.1819%	30.3259%
Accuracy	**Xlnet_tiny**	71.0690%	79.1642%	78.3868%
Loss		58.5031%	50.0136%	49.9366%

Table 1 The accuracy and values of all models in this paper after finetuning and evaluated on `clean_testset`. The models in bold and highlighted red are the best performing models in the current dataset providing methods.

factorization order.

3. Approach

In this paper, we designed a uniform routine to explore the training performance of models on different datasets: noise and clean. The routine illustration is in Figure 1 above.

First the routine would be provided a list of models (BERT-tiny, distillGPT2, XLNet-tiny), and select one model at a time for training and evaluation. Then, for each model, it would first create two identical copies, i.e., two model instances with identical parameter values. One of them is trained using `noise_dataset` and the other one using `clean_dataset`. Then, we evaluate the two fine-tuned models using the same independent test dataset. Dataset split and segmentation will be discussed in chapter 4 below.

After completing this above step, the training routine would fine-tune the noise-tuned model further: using clean_dataset to fine-tune it again. Then this new model would have been fed both noise and clean datasets. In doing so, we could explore whether fine-tuning the model with noise data first will affect the final performance if we also want to give it a final `clean_dataset` fine-tuning.

4. Dataset and Metric

The following datasets were used in this paper.

- `clean_dataset`: For the training part, we leverage human-annotated tweets sentiment data[3] by Amazon Mechanical Turk, which is included in SemEval Task 2013, 2014, 2015 and 2016. In the following we will call it `clean_trainset`. For the testing part, we use the SemEval Task 2017's test data [4] (referred as `clean_testset`).

- `noise_dataset`: A tweets sentiment dataset automatically annotated based on emojis.

`clean_trainset` has a total of 29,500 tweets, of which 12,740 are positively classified (43%), 4,578 are negatively classified (16%), and 12,182 are neutral. `clean_testset` has a total of 20,632 tweets, of which 7,059 are positively classified (34%), 3,231 are negatively classified (15%), and 10,342 are neutral. `noise_dataset` has a total of 1.6 million tweets, of which The C dataset contains only positive and negative data sets, each with 50% (800,000 items).

Since `noise_dataset` has only 2 categories (positive and negative) and does not have neutral label. Therefore, to ensure consistency throughout the training, we removed all the data of neutral data during the Wtraining. After removing the data, the datasets statistics as follows:

- `clean_trainset`: In total 17318, positive: 12740 (74%), negative: 4578 (26%).

- `clean_testset`: In total 10290, positive: 7059 (69%), negative: 4578 (31%).

In this paper, we use Accuracy as the evaluation function. The function is defined as follows:

$$\frac{TP + TN}{TP + FP + TN + FN}$$

where TP is true-positive, TN is true-negative, FP

[3] 2017_English_final.zip (dropbox.com)

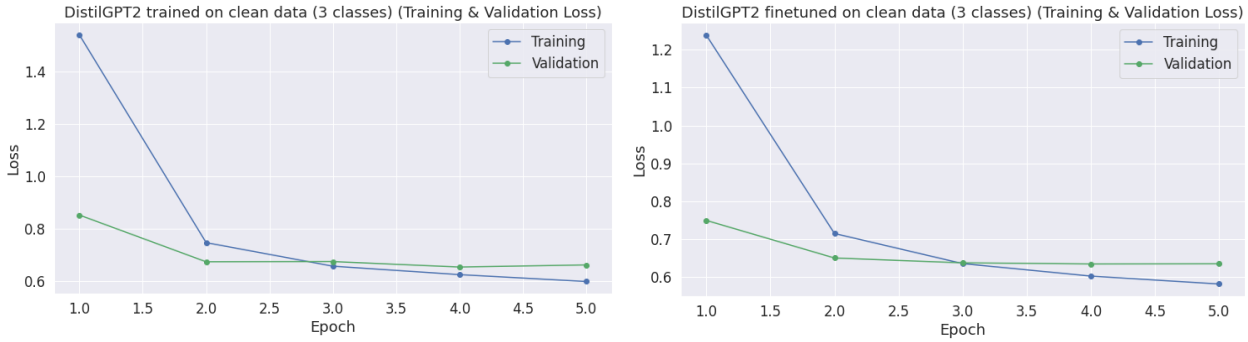

Figure 4 The accuracy of models trained using 3-labels training datasets and evaluated on 3-labels test datasets. The figure on the left shows the result of a model directly trained on `clean_trainset`, and the figure on the right shows the result of a DistillGPT2 model first trained using binary-classification noise data, and then finetuned using 3-labels training datasets.

is false-positive, and *FN* is false-negative.

5. Experiments and Results

We trained and tested each model used in this paper using the routine discussed in chapter 3 above. We first performed a training/validation dataset split on `clean_dataset` with neutral classification removed, with a split ratio of 9:1. The training results are in Table 1.

From the table, we can find that DistillGPT2 outperforms other models when trained using merely clean data or combing both noise data and clean data. Besides, BERT-tiny model performs well when training using only noise data. Xlnet-tiny model doesn't perform so well on each dataset-feed method. From the data we can see that in the case of clean and clean+noise, it is only 78% compared to other models (about 90%). And the accuracy for the noise only case is not very good either, with 71% compared to 74% for the other models. Our guess is that since the models are third-party implementations of XLNet code on Huggingface, it is possible that some specific code implementation problems is causing the lower accuracies.

Nevertheless, we found that, on average over all the three models, the iterative fine-tuning approach of first using noise data and second clean data always yielded better results. In particular, on both DistillGPT2 and BERT, the effect of noise + clean is slightly better than that of model only fine-tuned on clean dataset alone.

6. Preliminary Code

GITHUB: https://github.com/ycpeng8/BU_CS_505_NLP_Project.

7. Future Works

In our training, we found that DistillGPT2 can obtain the best accuracy results with the current routine. Therefore, we intend to continue to explore if we can get better results with the triple classification `clean_dataset` finetuning after the original model was finetuned binary-labelled `noise_dataset`.

After training we obtained the following results as shown in Figure 4. The accuracy of model trained using merely clean_trainset is **65.7861%**, while finetuned using both binary-classification noise data and 3-label clean data is **67.1335%**.

From this, it can be concluded that even if the noisy data are trained using 2 classification and then continue to be trained using 3-labelled data, the model can still be improved. We will continue to explore this aspect of performance comparison in the future on additional metrics such as the F-1 macro/micro score.

References

[1] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, Minnesota., 2019.

[2] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," arXiv, 2019.