NLP project1 report

**Natural Language Inference for Fake News Detection**

SPRING 2019 資工四 B04902051 林承豫

Project 2 資工四 B04902099 黃嵩仁

Due Date: 2019.6.23 網媒博二 D06944009 蘇弘庭

Division of Work

資工四 B04902051 林承豫

Logistic Model training and data preprocessing

資工四 B04902099 黃嵩仁

SVM Model training and testing

Preprocessing

網媒博二 D06944009 蘇弘庭

NN Model training and testing

**Methods**

1.Logistic regression

We use pre-trained toolkit “Spacy” to get the vector representations of each word. Then, the sentence/tweet embedding can be obtained by averaging all word embeddings in that sentence. The word is only tokenized by space without any preprocessing. Then we trained a logistic regression model for each subtask.

2. SVM model

Embedding：We use pretrained online toolkit “fastText” (<https://fasttext.cc/>) to get the vector representations of each word. Then, the sentence/tweet embedding can be obtained by averaging all word embeddings in that sentence.

SVM：We use LinearSVC in the sklearn package. (arguments: random\_state=0, tol=1e-5, and for subtask c: multi\_class='crammer\_singer' )

Tokenization：In “EMB+SVM” version, we only use “space” as separator, so it’s not robust enough. For example, a simple meaning “thank” with get lots of version, “Thank”, “Thanks”, “Thanks!!!”, “Thanks,”, “Thanks.”. So, we take the paper “NULI at SemEval-2019 Task 6: Transfer Learning for Offensive Language Detection using Bidirectional Transformers” as reference. We remove all "@USER" first, then we deal with the segmentation part. We use the segmentation tool on github(<https://github.com/grantjenks/python-wordsegment>), it can handle the “hashtag” properly, for instance, “#conservativesattackondemocracy” will be converted to “['conservatives', 'attack', 'on', 'democracy']”. With this method, the performance improve in subtask\_b and subtask\_c.

4. Neural Networks: BiDAFake

Intuitively, to determine whether a sentence S2 agrees another sentence S1 or not, we pay attention to some specific words according to each other. More precisely, instead of simply represent a sentence by widely used bag of words or recurrent neural networks, we apply the attention mechanism to each ot the sentence based on another one (i.e. attend S1 with S2, and attend S2 with S1).

**Preprocessing**: from [5]

Emoji substitution: We use one online emoji project on github which could map the emoji uni-code to substituted phrase. We treat such phrases into regular English phrase thus it could main-tain their semantic meanings, especially when the dataset size is limited.

HashTag segmentation: The HashTag becomes popular culture cross multi social networks, in-cluding Twitter, Instagram, Facebook etc. In order to detect whether the HashTag contains profan-ity words, we apply word segmentation using oneopen source on the github.

Misc: We also convert all the text into lowercase. Consecutive‘@USER’s are limited to three times to reduce theredundancy.

<https://github.com/carpedm20/emoji>

<https://github.com/grantjenks/python-wordsegment>

BERT

BERT [2] achieves state-of-the-art performance in many tasks, such as reading comprehension. We implement a BERT-based classifier leveraging the pytorch implementation of BERT [4] to three tasks. As suggested by previous work [5], we set the max length to 64 and apply the default setting. Although the max length of the tweet is more than 512, we find setting max length to 512 and 64 results in a similar performance.

BERT+NER

Task C, which aims to detect the target of offensive language, we observe that most of the targets are named entities. However, according to prior work [3], BERT does not perform well comparing to the previous state-of-the-art (-7%) on the NER task while surpasses the previous state-of-the-art on most tasks. Observing this, when training task C, we multitask the model with another NER model training with CONLL-2003 dataset.

Experiment Results

|  |  |  |  |
| --- | --- | --- | --- |
|  | TaskA | TaskB | TaskC |
| BiLSTM [1] | .750 | .660 | .470 |
| CNN [1] | .800 | .690 | .470 |
| EMB+SVM | .688 | .470 | .420 |
| EMB+Logistic | .703 | .304 | .217 |
| EMB+SVM+WS | .632 | **.506** | .479 |
| BERT | **.744** | .470 | .481 |
| BERT+NER | - | - | **.511** |

BERT

BERT reaches the best result on both task A and C in our experiments, demonstrating the transferability of contextual representation. According to [1], the model can be further improved with more proper tuning, preprocessing, and an ensemble model.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Layer Shared | 0: Only Embs | 3 | 6 | 9 | 12 |
| Task C F1 | +0.1% | -1.8% | **+1.1%** | **+3.0%** | -24.5% |

BERT+NER

According to the table above, we show that multitasking with NER task strengthens our model on task C. We try different settings when sharing the models. Sharing the embedding and first 9 layers boost the F1 score by 3.0%. It is worth to mention that sharing the full BERT encoder drops the performance, indicating that two tasks share similar lower level and mid-level representations.

Reference

[1] Zampieri et'al, SemEval-2019 Task 6: Identifying and Categorizing Offensive Language

in Social Media (OffensEval), NAACL 2019 workshop

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, NAACL 2019

[3] Linguistic Knowledge and Transferability of Contextual Representations, NAACL 2019

[4] https://github.com/huggingface/pytorch-pretrained-BERT

[5] NULI at SemEval-2019 Task 6: Transfer Learning for Offensive Language Detection using Bidirectional Transformers, NAACL 2019 workshop

[6] https://github.com/grantjenks/python-wordsegment