Test task: Sentiment prediction for Facebook posts

Konstantin Chernyshev
kdchernyshev@gmail.com

Abstract

Automatic text sentiment detection is a well-known problem in NLP and many approaches have been successfully applied. In this paper, we will probe some approaches and model architectures, such as classical machine learning (ML) models, deep learning (DL) models, and reformulation of the problem as an entailment problem (EFL). In our experiments, we show the prevalence of DL approaches and the viability of the EFL approach (but further research is required).

Source code is available on GitHub[1]. The experiments tracking data is available via neptune.ai[2].

Note: *This paper is NOT following a scientific paper layout, fullness, and standards, not all models are cited.* ***It is a report, rather than a publication.***

1 Introduction

Sentiment analysis, a rapidly growing problem in Natural Language Processing (NLP), has become a popular research topic as it can be applied to public option research via reviews analysis or social nets, and even stock training with news sentiment studies. In sentiment analysis, the task of identifying the polarity of a text, whether it is positive, negative, or neutral, is a fundamental problem.

[A placeholder for the research importance part]

Various approaches have been developed to address this problem, ranging from classical machine learning (ML) models to deep learning (DL) models. Classical ML models, such as Support Vector Machines (SVM) and Random Forests (RF), have been applied successfully in sentiment analysis tasks, but they require a handcrafted feature engineering process that can be time-consuming and labor-intensive. On the other hand, DL models, such as Transformers models, have gained popularity due to their ability to automatically learn features from raw data and achieve state-of-the-art (SOTA) performance on various NLP tasks.

In addition to these approaches, recent research has also explored the possibility of reformulating the sentiment analysis problem as an entailment problem (EFL) (Wang et al., 2021) In the EFL approach, the task of sentiment analysis is transformed into a binary classification problem, where the goal is to determine whether a premise sentence entails a hypothesis sentence with a positive or negative sentiment. This approach has shown promising results (current SOTA in the SNLI task), but its viability in sentiment analysis tasks is still an open question.

The rest of the paper is organized as follows: Section 2 provides a brief methodology discussion. Section 3 describes the experimental setup of this study. Finally, Section 4 presents the results and analysis of our experiments.

2 Design

In this section, we will briefly discuss some points on experiments design: dataset, evaluation, and models to experiment with.

2.1 Data

Class	#Samples	Avg. Len
negative	79	190.4
neutral	280	93.8
positive	641	115.5
total	*1000*	*115.3*

Table 1: Dataset statistics. Total number of Posts (#Samples), Number of Days with average text length in characters available (Avg. Len).

[1] github.com/k4black/JB-internship-2023-reaction-prediction

[2] new-ui.neptune.ai/k4black/jb-reaction-prediction

The datasets provided consists of a relatively small amount of data, 1000 FaceBook posts. The data is not divided to train/val/test split, which limits us even more. We see a class imbalance in the dataset and we will experiment to deal with it in later stages. Also noticeable is a significant difference in the average length of the post between the classes with the longest for negative reactions. However, we do not plan to include this as a feature, as it could lead to overfitting on our data and significantly affect it when generalizing to slightly different data (e.g. Twitter or Reddit data). The dataset statistics are available in Table 1.

2.2 Evaluation

F1 score with macro averaging was chosen as a target metric. At this point we do not have any preferences on detecting positive/negative posts, so we treat all classes equally.

Given a small amount of data – all experiments were performed using 5-fold cross-validation with out-of-fold scores reported. Ideally, we should consider separating the test set, since in the current setup the possibility of overfitting a model arises since we use in-fold test-split for both training validation and out-of-fold prediction. However, the amount of data does not allow us to work this out correctly, so it was decided to use cross-validation as is. This way we might get less reliable scores for a final model, but it works fine for model comparison.

2.3 Pretrained models usage

The task description included the line "you need to train the model yourself rather than using a pretrained one", which I interpreted as prohibiting end-to-end trained models to predict sentiment. Because at the current stage, it is impossible to imagine NLP without using pretrained Large Language Models (LM), as they deliver substantially better quality, as will be shown below. Therefore, pretrained LMs will be used for part of the study, but we will do all the finetuning. Similarly, the pretrained FastText model will be used as is.

2.4 Models

In this paper, we explored 3 approaches:

- We used different 'classical' Machine Learning (ML) models to select a baseline and got a general understanding of the scores expected;
- Then we chose a suitable Deep Learning (DL) model architecture and tried to find parameters to obtain the highest score by simple model finetuning;
- Lastly, we tried a slightly exotic *"Entailment as few-shot learner"* (EFL) (Wang et al., 2021) approach – re-formulation of the classification problem as an entailment problem. In our case, 3 prompts (one per class) will be constructed for each example. (e.g. classification task "I hate you!" as "I hate you! [SEP] It was hate speech.")

3 Experiments

In this section, we describe a number of experiments that are being conducted: with 'classical' Machine Learning (ML) models (more for comparison than for quality), finding a suitable architecture for Deep Learning (DL) models, and experiments with the chosen DL architecture. We used the `base`-sized models, we expect higher quality for the larger models, but due to the resource limitations, we used smaller versions for comparison. All experiments were conducted in Google Colab. Random seeds are fixed.

3.1 'Classical' Machine Learning

To select a baseline model to experiment with, we first performed a rough model search, training some widely-used models against some popular vectorization methods. The experiments were carried out using the `scikit-learn` (Pedregosa et al., 2011) library. Initial searches were conducted using default parameters. The following classifiers were employed: Logistic Regression, SVM, Random Forest, Gradient Boosting, Naive Bayes, and K-Nearest Neighbors, in conjunction with Count Vectorizer, Tf-Idf Vectorizer, Hashing Vectorizer, and FastText model (pretrained weights).

As indicated in Table 2, the best model and vectorizer combination was LinearSVM with Count vectorizer achieving **0.598** f1-macro, which was also one of the fastest models. As a result, this combination was chosen as the baseline model. All in all, you can see that this task is quite difficult, and achieving a high score will not be easy.

3.2 Deep Learning

Similarly to the 'classical' ML models, we performed some searches for the best DL architecture to experiment with. We used language models that

	Count	Tf-Idf	Hash	FastText
LogReg	0.587	0.446	0.472	0.375
SVM	0.505	0.440	0.469	0.474
LinearSVM	**0.598**	0.521	0.532	0.460
RandomForest	0.477	0.452	0.421	0.468
GradientBoosting	0.589	0.558	0.536	0.496
NaiveBayes	0.416	0.273	(not applicable)	(not applicable)
KNeighbors	0.445	0.436	0.433	0.441

Table 2: 7 ML models against 4 popular vectorizers trained using default parameters and basic preprocessing. Out-of-Fold f1-macro scores for 5-fold Cross Validation are reported. Bold font indicates the best result.

Model Name	f1-macro	Avg. Training Time
bert-base-uncased	**0.732**	3m 13s
roberta-base	0.694	3m 21s
albert-base-v2	0.691	2m 48s
vinai/bertweet-base	0.690	3m 13s
facebook/muppet-roberta-base	0.706	3m 22s
distilbert-base-uncased	**0.732**	**1m 50s**
distilroberta-base	0.703	2m 1s

Table 3: Finetuning of DL models. Names refer to the Huggingface Hub model name. Out-of-Fold f1-macro scores for 5-fold Cross Validation are reported. The average fold finetuning time on Tesla T4 GPU is provided for comparison and may differ on conditions. Bold font indicates the best result.

are trained based on the transformer architecture (Vaswani et al., 2017). The Transformer model is a State-of-the-Art DL architecture for natural language processing (suitable for a wide range of tasks). The `transformers` (Wolf et al., 2020) library was used, as well as pretrained models available on `Hugginface Hub`[3].

We fixed training parameters and went through training of some popular suitable models. We had a learning rate of 1e-5, batch size of 32, weight decay of 0.01, and max epoch of 20 with early stopping on 5 steps with increasing f1-macro score. The results are available in Table 3.

We can see that DL models are achieving better results overall, but the results are not yet 'good'. Model `bert-base-uncased` performed the best, but surprisingly model `distilbert-base-uncased` also got a very high result, both with **0.732** f1-macro scores. Since we are aiming at the enterprise use of the model, when the performance can be critical, we decided to continue experimenting with model `distilbert-base-uncased`, since it is several times more efficient in inference than the first one.

Afterward, a number of experiments were conducted varying the batch size and learning rate to find the optimal parameters. The best model `distilbert-base-uncased` achieved **0.786** f1-macro at a learning rate of 2e-5 and batch size of 8 (other parameters are fixed).

Also, a number of data augmentation experiments were conducted, but it did not lead to a considerable increase in quality.

3.3 Entailment as Few-Shot Learner

Due to the fact that the number of examples for training increases by 3 times (each is disclosed as 3 prompts) but the amount of data for the test does not increase – cross-validation is still required. So, training time increasing manifold, we can not carry out the fullness of experiments to get the higher score possible, but rather 'probe' this approach.

Following the original authors, we add some models pretrained on the MNLI task as it can increase the score. We fixed training parameters and went through training of some popular suitable models. We had a learning rate of 1e-5, batch size of 8, weight decay of 0.01, and max epoch of

[3] huggingface.co/models

Model Name	EFL Score	f1-macro
bert-base-uncased	0.889	0.738
textattack/bert-base-uncased-MNLI	0.887	**0.766**
roberta-base	0.893	0.760
textattack/roberta-base-MNLI	0.887	0.736
distilbert-base-uncased	0.881	0.726
typeform/distilbert-base-uncased-mnli	0.876	0.729

Table 4: EFL approach models. Names refer to the Huggingface Hub model name. The final Out-of-Fold f1-macro scores for 5-fold Cross Validation are reported. The EFL Score stands for the f1-macro score on the entailment problem on the fold test split. Bold font indicates the best result.

Model	Params Adj.	f1-macro
Count Vectorizer + LinearSVM	–	0.598
distilbert-base-uncased	–	0.732
distilbert-base-uncased	+	**0.786**
textattack/bert-base-uncased-MNLI + EFL	–	0.766

Table 5: Resulting table of the best model mentioned before. Out-of-Fold f1-macro scores for 5-fold Cross Validation are reported. Params Adj. stands for parameters adjustment – more careful hyperparameter searching. Bold font indicates the best result.

20 with early stopping on 5 steps with increasing f1-macro score of the entailment problem.

As we can see in Table 4 approach shows itself quite consistently obtaining comparable results. MNLI versions of the models perform... quite randomly, one can assume that because of 'catastrophic forgetting' and training these models requires more fine-tuning of the parameters. The best result of **0.766** was achieved by the textattack/bert-base-uncased-MNLI model – Bert model further pretrained with NLI task, with the assumption that it can be improved significantly by adjusting the parameters.

4 Results

In this study, we conducted experiments with both 'classical' Machine Learning (ML) models and Deep Learning (DL) models, as well as the EFL approach, to perform a classification task. We report the results of our experiments in this section.

A rough search of the ML models showed sufficient complexity of the task and a relatively low score was obtained. DL models performed better with the best model showing **0.786** f1-macro score in 5-fold cross-validation. While the ELF approach proved to work, but did not perform as well.

However, due to time and resource constraints, we have not performed all experiments possible in the current setup and further research may indeed be conducted. We expect higher quality when training large-sized models, and we also expect the EFL approach to perform better in fine-tuning.

References

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. 2021. Entailment as few-shot learner. *CoRR*, abs/2104.14690.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. pages 38–45. Association for Computational Linguistics.