University of Bonn

CAISA lab

Introduction to Natural Language Processing

(Winter Semester 2023/2024)

Default Project

Final Report of Team

Rude Comment Scoring: A Comparative Evaluation of Classical Regression Models and LLM Fine-Tuning Techniques

Group Members:

Mohammad Mehdi Balouchi 3369523

Lecturer: Prof. Dr. Lucie Flek
Course coordinator: Vahid Sadiri Javadi

March 16, 2024

1. Introduction

On social media platforms, hateful and offensive language negatively impact the mental well-being of users and the participation of people from diverse backgrounds [1]. Therefore, automatic techniques for identifying toxic language are required. One useful dataset in this area is Ruddit; extracted Reddit comments in English and is continuously labelled with the rudeness of the comment. Given that the dataset labels represent a continuous score between -1 and 1, regression models emerge as the most suitable candidates for predicting the rudeness of new comments.

In our exploration of classical approaches, we employed various regression models, including Linear Regression [2], Support Vector Regression (SVR) [3], Multi-Layer Perceptron (MLP) [4] regressor, and Random Forest Regressor [5]. Our curiosity led us to test and evaluate different text representations such as BOW (Bag Of Words) , TFIDF [6], Word2Vec [7], Glove [8], Fasttext [9] and BERT [10] as input features for these methods. Furthermore, given the recent advancements in transformer-based methods, which have demonstrated state-of-the-art performance in numerous Natural Language Processing (NLP) tasks, we opted to compare the results of classical approaches with the fine-tuning of Large Language Models (LLMs) such as BERT on the Ruddit dataset.

In our implementation, we started with text cleaning as a part of our data preprocessing. This involved a series of steps including punctuation removal, lowercasing, stop word elimination, and lemmatization. Subsequently, upon evaluating the results, we conducted additional tests by training the models without cleaning the corpus, utilizing the raw text dataset. The disparity in the evaluation outcomes prompted us to include text cleaning as a crucial factor in our comparative analysis.

1..1 Objectives:

The primary goal of this study is to compare classic ML regression methods with the LLM fine tuning method. Additionally, we looked at the impact of various pre-training settings, such as data cleansing. Since the selected dataset is a traditional regression NLP problem, our goals are to train various models, compare evaluations and timings, and investigate how data cleaning affects various approaches.

1..2 Research Questions:

Our main research questions in this projects are:

- What are the conventional ML regression models' performances?
- What is the influence of various dataset representations on each model's result?
- Which one have a better performance on the regression task: LLM fine-tunning vs traditional regression models?
- What is the impact of data cleaning on the performance of various models? Is it consistently associated with performance improvement?

2. Data Exploration and Preprocessing:

The dataset we choose is the Ruddit dataset [1]. It contains Reddit comments in English and is continuously labelled with the rudeness of the comment. Where -1 is a most supportive comment and 1 is a most offensive comment.

2..1 Dataset Extraction:

The dataset is extracted from the Pushshift repository [1]. It contains 808 posts. For each post 50 comments are collected. With further processing and labelling this results in a total of 6000 labelled comments.

They provided comment IDs and not the comment body in accordance to the GDPR regulations. We extracted the comment body using the Reddit API. As some of the comments were deleted or removed, the final number of comments extracted is 5634.

2..2 Statistics:

Table 1: Datasets comment's text statistics

Dataset	#Comments	Average #tokens	Max #tokens	Min #tokens
train	4225	17.59	83	0
test	1408	17.28	80	1

Table 2: Datasets score's text statistics

Dataset	Mean	STD	Min	Max
train	-0.031	0.32	-0.88	0.93
test	-0.02	0.34	-0.87	0.97

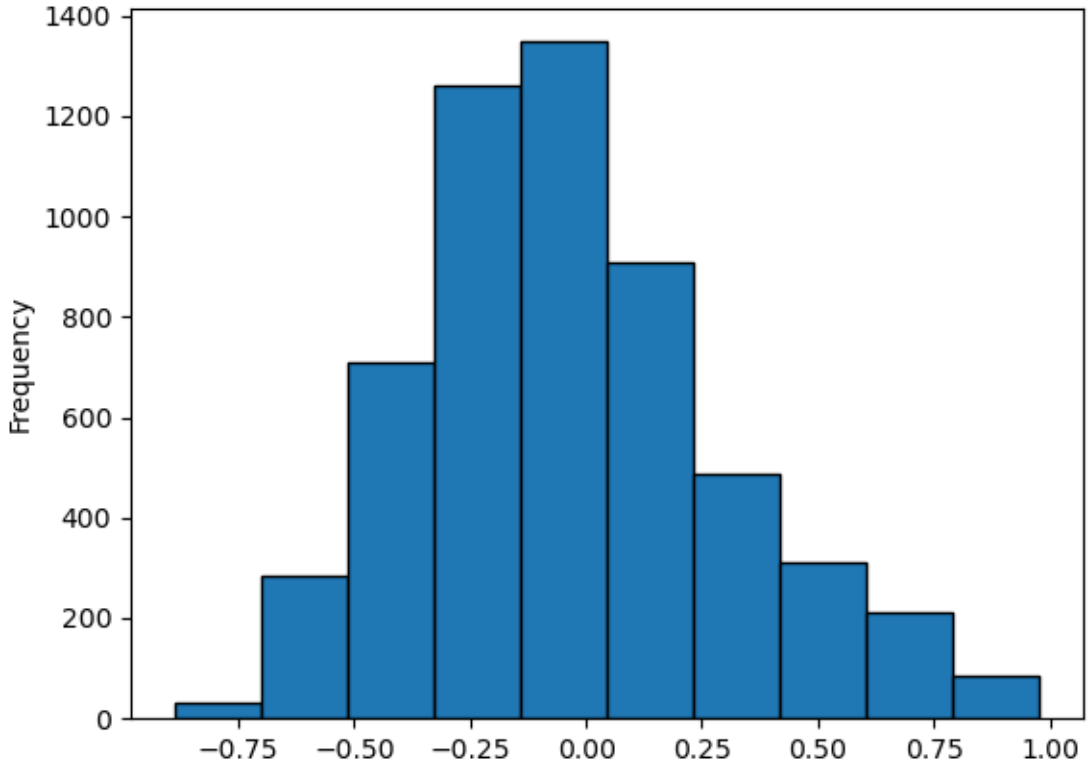

Figure 1: Distribution of offensiveness score.

3. Methodology:

The first step of our project was dataset extraction. Then the final results was saved into an .csv file. The first step is the dataset parsing which is converting the dataset to a dataframe using the comment body and the related score. The second implemented method was dataset cleaning including punctuation removal, lowercasing, stop word elimination, and lemmatization. After downsampling the dataset by splitting it to the train-test subsets (train-ratio=0.75) we saved the clean and raw datasets for each train and test subsets. The next step is generating different

representations such as BOW (Bag Of Words), TFIDF, Word2Vec, Glove, Fasttext and BERT embeddings and feed it to the desired traditional models. We trained each model and evaluate it on the test dataset representations and log the results. Moreover, we fine-tuned BERT with our dataset and evaluate it with the same metric (MSE).

3..1 Models:

- **Linear Regression:** Linear regression is a linear approach to modeling the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the dependent variable and the predictors and aims to minimize the sum of the squared differences between the observed and predicted values.
- **Support Vector Regression (SVR):** SVR is a type of support vector machine algorithm that is used for regression tasks. It works by finding the hyperplane that best fits the data points while minimizing the error. SVR uses a subset of training data points, known as support vectors, to define the hyperplane.
- **Multi-Layer Perceptron (MLP) Regressor:** An MLP regressor is a type of artificial neural network with multiple layers of nodes. It can be used for regression tasks by learning a mapping from input variables to output variables. MLP regressors consist of an input layer, one or more hidden layers, and an output layer. Each node in the network performs a weighted sum of its inputs and passes the result through an activation function.
- **Random Forest Regression:** Random forest regression is an ensemble learning method that constructs a multitude of decision trees during training and outputs the average prediction of the individual trees for regression tasks. Each decision tree in the forest is built on a randomly sampled subset of the training data and a random subset of features. The final prediction is the average of predictions from all trees in the forest, which tends to reduce overfitting and improve generalization.
- **BERT:** Bidirectional Encoder Representations from Transformers (BERT) is a language model based on the transformer architecture, notable for its dramatic improvement over previous state of the art models. It was introduced in October 2018 by researchers at Google.A 2020 literature survey concluded that "in a little over a year, BERT has become a ubiquitous baseline in Natural Language Processing (NLP) experiments counting over 150 research publications analyzing and improving the model."BERT was originally implemented in the English language at two model sizes: (1) BERTBASE: 12 encoders with 12 bidirectional self-attention heads totaling 110 million parameters, and (2) BERTLARGE: 24 encoders with 16 bidirectional self-attention heads totaling 340 million parameters. Both models were pre-trained on the Toronto BookCorpus(800M words) and English Wikipedia (2,500M words).

3..2 Code Architecture & Libraries:

- Python Version: 3.10
- Libraries:
  - `pandas` - A data manipulation and analysis library.
  - `sklearn` - A machine learning library providing various algorithms and tools for classification, regression, clustering, etc.
  - `re` - A library for regular expressions in Python.
  - `nltk` - The Natural Language Toolkit, a library for natural language processing tasks such as tokenization, stemming, lemmatization, and more.
  - `transformers` - A library by Hugging Face for state-of-the-art natural language processing tasks, including pre-trained models and tokenizers.
  - `torch` - The PyTorch library for tensor computations and deep learning.
  - `gensim` - A library for topic modeling, document indexing, and similarity retrieval with large corpora.
- Models and Tools:
  - `AutoTokenizer, AutoModelForSequenceClassification` - Classes from the `transformers` library for tokenization and sequence classification tasks.

- `mean_squared_error` - A function from `sklearn.metrics` for calculating the mean squared error.
- `TrainingArguments, Trainer` - Classes from the `transformers` library for defining training arguments and training models.
- `CountVectorizer` - A class from `sklearn.feature_extraction.text` for converting text documents to matrix of token counts.
- `TfidfVectorizer` - A class from `sklearn.feature_extraction.text` for converting text documents to matrix of TF-IDF features.
- `Word2Vec` - A class from `gensim.models` for word embedding using Word2Vec.
- `api` - An interface for accessing the Gensim model repository.
- `BertModel, BertTokenizer` - Classes from `transformers` library for BERT-based model and tokenizer.
- `LogisticRegression, LinearRegression, SVR, MLPRegressor, RandomForestRegressor` - Classes from `sklearn.linear_model`, `sklearn.svm`, and `sklearn.ensemble` for regression tasks using different algorithms.

4. Experimental Setup

The selected base model is `"bert-base-cased"`, which is a pre-trained BERT model with a cased vocabulary.

Fine-tuning involves taking the pre-trained BERT model and training it on a specific downstream regression task using the 'AutoModelForSequenceClassification' class of huggingface.

Hyperparameters:

- Learning Rate: The learning rate determines the step size at which the model weights are updated during training. In this case, the learning rate is set to 2×10^{-5}.
- Max Length: The maximum length of input sequences, which is set to 256 tokens.
- Batch Size: The number of training examples processed in one iteration. Here, the batch size is set to 8.
- Epochs: The number of complete passes through the training dataset during training. The model is trained for 7 epochs.
- Weight Decay: A regularization technique that penalizes large weights in the model. Here, the weight decay is set to 0.01.

5. Evaluation and Results

We used Mean Squared Error (MSE) which is a commonly used evaluation metric for regression models. It measures the average squared difference between the actual values and the predicted values produced by the model. MSE is particularly useful because it gives more weight to larger errors due to squaring each error term.

The MSE formula is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Table 3: MSE results comparison on cleaned dataset

	bow	tfidf	Glove	Word2Vec	bert_embeddings	***fasttext**
***SVR**	0.1238	0.1201	0.0503	0.0482	0.0526	****0.0469**
Linear	0.1734	0.1411	0.0783	0.0707	**0.0582**	0.0727
MLP	0.2564	0.2237	0.0733	0.0655	0.0736	**0.0564**
RF	0.1131	0.1063	0.0729	0.0701	0.0675	**0.0645**

Table 4: MSE results comparison on raw dataset

	tfidf	bow	Glove	Word2Vec	***bert_embeddings**	fasttext
***SVR**	0.1303	0.1282	0.0734	0.0596	****0.0461**	0.0588
Linear	0.1721	0.1646	0.0947	0.0895	**0.0526**	0.0801
MLP	0.2325	0.2105	0.1038	0.0970	**0.0602**	0.0810
RF	0.1159	0.1133	0.0853	0.0804	**0.0667**	0.0773

The bold values in the tables represent the best MSE score within each row. It's evident that fasttext embeddings and bert_embeddings consistently produce the lowest MSE values in their respective datasets and models, making them strong candidates for feature representation in regression tasks.

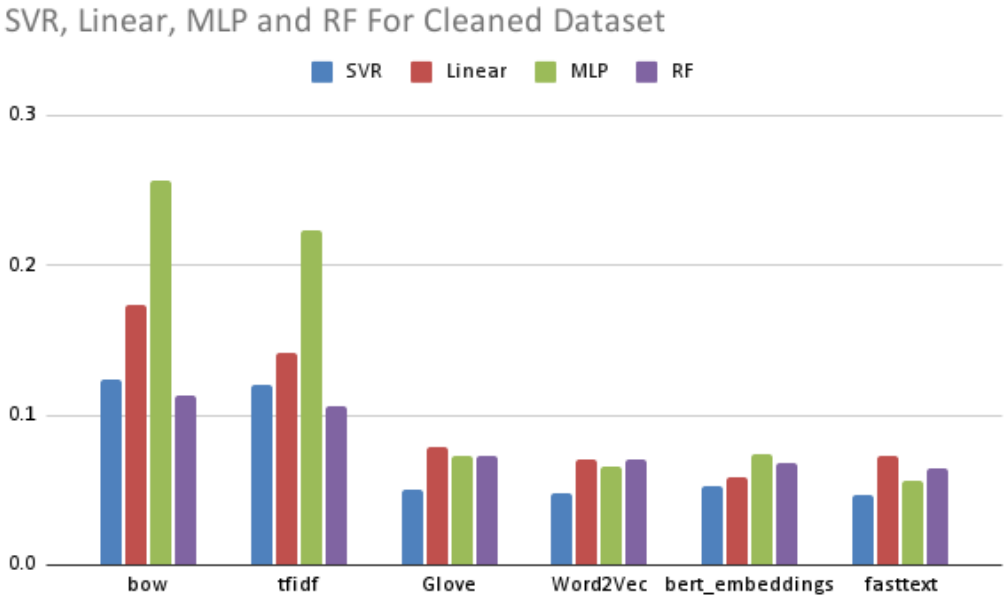

Figure 2: Comparison of the models with different representations on pre-processed cleaned dataset

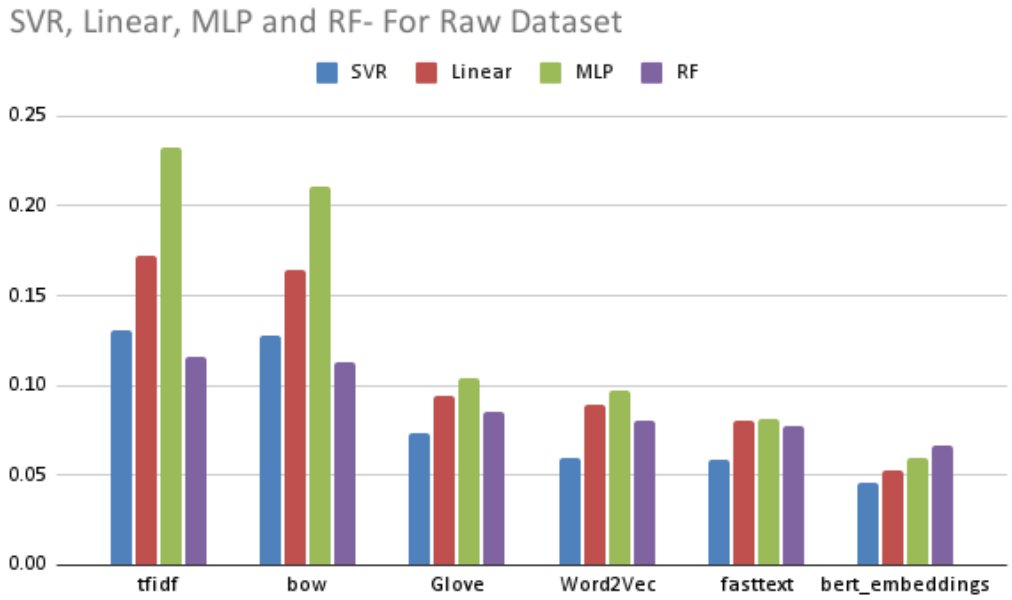

Figure 3: Comparison of the models with different representations on raw dataset

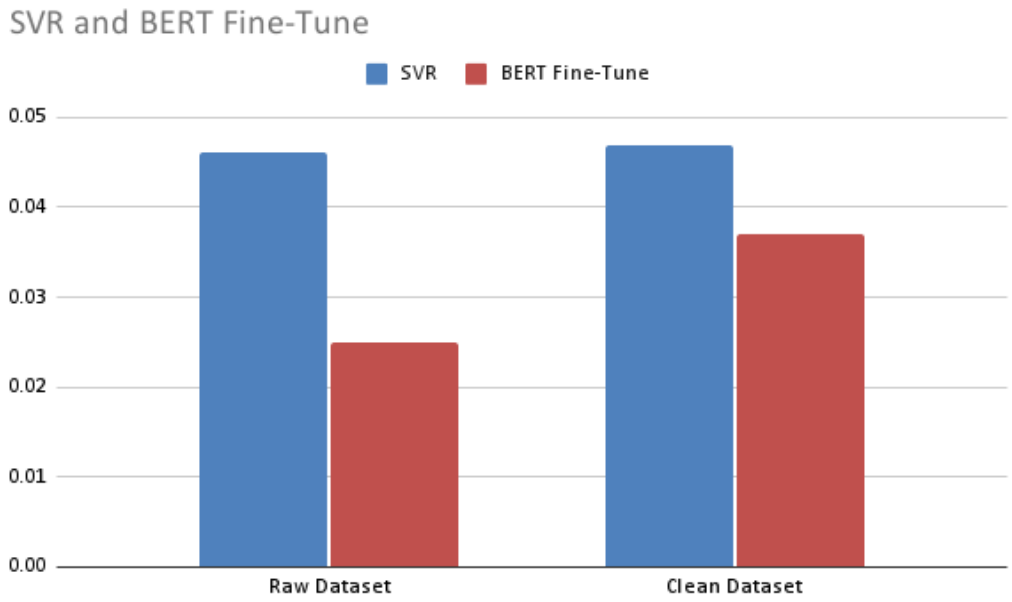

Figure 4: Comparison of the best performed traditional ML regression method (SVR) vs Fine-Tune on Raw and Clean Dataset

6. Analysis and Discussion

6..1 Comparison Across Models:

Across both cleaned and raw datasets, SVR with fasttext embeddings consistently performs the best in terms of MSE compared to other models. This indicates that fasttext embeddings are effective in capturing the underlying patterns in the data for regression tasks. Linear regression also performs reasonably well, but SVR generally

outperforms it across different feature representations. MLP (Multi-layer Perceptron) and RF (Random Forest) models perform variably across different feature representations and datasets. They don't consistently outperform SVR or linear regression.

6..2 Comparison Across Feature Representations:

For the cleaned dataset, Glove and Word2Vec embeddings generally result in lower MSE values across most models compared to bag-of-words (bow) and tfidf representations. For the raw dataset, the results are more mixed. While bert_embeddings consistently perform well across different models, tfidf and bow representations also show competitive performance. Overall, there isn't a single best-performing feature representation across all models and datasets, indicating the importance of experimenting with different representations based on the specific task and dataset characteristics.

6..3 Best Performed Traditional ML regression model (SVR) vs BERT fine-tuned:

On both the Raw Dataset and the Clean Dataset, SVR exhibits higher MSE values compared to BERT Fine-Tune. This suggests that SVR's predictions have larger errors on average. This indicates that SVR's performance is consistent across the two datasets, as the MSE values are close to each other. BERT Fine-Tune consistently outperforms SVR on both datasets in terms of MSE. Although BERT Fine-Tune's performance is superior to SVR, it seems to perform relatively better on the Raw Dataset compared to the Clean Dataset. The MSE on the Clean Dataset is slightly higher, indicating a slightly lower predictive accuracy on this dataset.

7. Conclusion

- SVR with fasttext embeddings consistently performs the best compared to other traditional ML regression models.
- The BERT Fine-Tune model seems to be more effective overall.
- The Bert shows better generalization and performance compared to the SVR model.
- LLM fine-tuned models seem to perform relatively better on the Raw Dataset compared to the Clean Dataset.

8. Future Work

We can extend this project in several aspects:

- Fine-tuning different LLMs and comparing it to our models.
- Trying out different hyper-parameters and evaluate their effects.
- Using different dimensionality reductions methods such as PCA and LDA and apply them to have variant input feature sizes.
- Experiment different permutations of data cleaning and not only applying all the data cleaning steps.

References

[1] Rishav Hada, Sohi Sudhir, Pushkar Mishra, Helen Yannakoudakis, Saif M Mohammad, and Ekaterina Shutova. Ruddit: Norms of offensiveness for english reddit comments. *arXiv preprint arXiv:2106.05664*, 2021.

[2] J. Brian Gray. Introduction to linear regression analysis. 44(2):191–192.

[3] Alex Smola and Bernhard Scholkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

[4] Fionn Murtagh. Multilayer perceptrons for classification and regression. 2(5):183–197.

[5] R. Goot. Normalizing social media texts by combining word embeddings and edit distances in a random forest regressor.

[6] Stephen Akuma, Tyosar Lubem, and Isaac Terngu Adom. Comparing bag of words and TF-IDF with different models for hate speech detection from live tweets. 14(7):3629–3635.

[7] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method.

[8] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

[9] Igor Santos, Nadia Nedjah, and Luiza De Macedo Mourelle. Sentiment analysis using convolutional neural network with fastText embeddings. pages 1–5.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North*, pages 4171–4186. Association for Computational Linguistics.