Text classification using data augmentation and ensemble learning

Badard Alexis **Lemaire Baptiste** **Picard Matthias** **Senhadji Romain**

Abstract

Text classification with few train samples is not a trivial task. We developed a solution using LLMs to synthesize new data automatically. After studying the data we were given, we decided to use different approaches using Machine Learning models. Our final model reached an accuracy of 86.3%, placing our team in the third position on the leaderboard.

1 Introduction

Text classification is a fundamental task with profound implications across various domains, like sentiment analysis or spam filtering systems. These use cases can show really good performances if we use the appropriate techniques to solve them. Accordingly, numerous approaches for performing text classification have been invented and improved during the history of NLP. From frequency-based to LLM-based solutions, there is a huge range of models that one can create to make a good classifier, and the most recent Deep learning architecture are not always the best solutions. In this paper, we will try to solve a basic sentence classification task by testing different approaches. We will present our thought process and the methodology that led us to create our final solution. A section will also be dedicated to our results and to further potential improvements.

2 Developed solution

2.1 Given data

Our dataset comprises 12 distinct categories, each containing three example entries related to specific topics such as politics, health, finance, and travel among others. We also have a test dataset that contains 1140 unlabelled sentences. This data seems to result from synthetic generation. Indeed, we can observe some patterns and similarities across various entries, indicating the use of templated phrases or structured data creation methods to populate different categories with thematic content. All the lines have a similar length with an average of 12.6 words per sentence and a standard deviation of 2.4 words.

2.2 Data Augmentation

As described above, we didn't have a lot of data to train our models. Therefore, we had to find a way to get access to more data. We decided to synthesize our own using different methods, mainly generative models. If we ask ChatGPT or Llama to rewrite a phrase, they can generate a brand-new instance automatically.

We decided to use the model Llama2 [Tou+23] from Meta. The model we used is registered in the HuggingFace database as "Llama-2-13b-chat-hf". It corresponds to the Llama 2 models with 13 billion parameters and is fine-tuned for chatting with users. We used this model in two different ways.

First, we use it to paraphrase the train instances. We process each one of the 36 instances, one at a time. The prompt takes as input the instances and the label. We then apply a few-shot paradigm for the prompt:

<s>[INST]I want you to write a general statement on the theme {label}. Find a balance between exploration and specialization. Start the sentence with "The". The sentence cant ever exceed 21 words long.[/INST]

We first use the prompt above with an example of what the model should generate. In this case, the example is the first train instance. The model will then generate a new sentence related to the label. For the next pass, we give two examples: the train instance and the previous generation. Giving two examples leads the generation to be more diverse, but also to be different from the previous generation. Therefore, we don't have duplicates in

the dataset.
We iterate this algorithm on each one of the train instances 50 times. We then end up with a dataset of 1800 instances.
To generate more data, we then thought about a second pipeline that aims to generate data without conditions. Here, the model generates data freely according to the label. As the prompt is easier to process, the generation is much faster. We can then generate a dataset of 3600 instances in no time.
For the second pipeline, we observed that the generated data tended to distance itself from what we would expect. Hence, its quality can't be secured. Training on this dataset alone did not lead to good performances. We still decided to concatenate this dataset with the 1800 generated samples to get a more sizeable dataset.

We also tried two other data augmentation methods: Back translation and EDA [WZ19]. However, the performances of our models did not improve dramatically when these augmentations were used. Probably because of the duplicated data that were generated. We still use the Back translation augmentation, while removing duplicates.

2.3 Our solution

As said before, we realized the test instances were generated automatically and that the diversity of the test data was very low. We supposed that Machine Learning models would perform better on this classification task. We trained different simple Machine Learning models. We first use an embedding model from Hugging Face [Tra] to embed our train set into vectors of 384 dimensions. Then, we trained the following classifier models from scikit-learn: SGDClassifier, RidgeClassifier, LinearSVC, and ExtraTreesClassifier.

Whenever it was feasible, we added an important L2 regularization, to improve their generalizing performances. We also added class_weights to each class, based on the proportion of each class prediction on the test set. This is based on the assumption that the test set is balanced. We then retrain the model. Looking at their performances individually, we first decided what the best model was. We then aggregated all their predictions on the test set using a hard-voting system. In the case of equality in the vote (say, models 1 and 3 think the theme is "Finance", and the other two think it is "Politic"). The best model will be the one that decides the final prediction.

3 Results and Analysis

As we needed to test many different methods, We decided to hand-label a subset of the test set containing 200 sentences, to have a first indication of the performances of a model.

In the end, we trained our model on more than 10,000 sentences, generated using a mix of back-translation and LLM generation techniques. We needed about 5 minutes to train the four models on a classic computer CPU. We obtain an accuracy of 86,3% on our best kaggle submission. On the confusion matrix that we constructed using our annotated data (Figure 1 in Appendix), we can see that the model does not suffer from important biases. Predictably, some classes that are naturally "close" are getting confused more than others (like "Politics" and "Finance"). On our annotated data, we get an accuracy of 87% which is close to the accuracy displayed on Kaggle. This led us to think that our annotated data was relatively representative of the whole train set, even though there were certainly some ambiguous sentences that we miss-annotated. These ambiguities hindered our capacity to make efficient error analyses.

A deeper analysis of the errors made by our model would probably give us some insights into what data we should generate as a priority to improve our performances. Using different LLMs to generate our data would also probably give us more diverse sentences (Llama3 [AIM24], Mixtral 8x7B [Jia+24], etc.). In the same fashion, adding more models to our cluster of models would also probably enhance our performances, although it would be preferable to use very different models (LSTM, fine-tuned LLM, TF-IDF) to limit potential biases. If we continue to consider the current classification models, using different embedding models for each of our classifiers would also likely improve prediction diversity.

Appendix

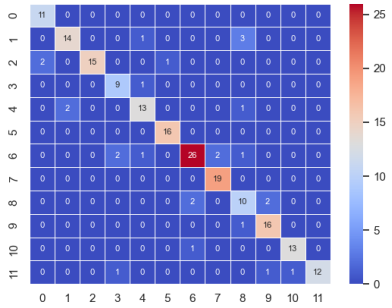

Figure 1: Confusion matrix computed on our annotated data. Note that the data is not balanced

References

[AIM24] AI@Meta. "Llama 3 Model Card". In: (2024). URL: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[Jia+24] Albert Q. Jiang et al. *Mixtral of Experts*. 2024. arXiv: 2401.04088 [cs.LG].

[Tou+23] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL].

[Tra] Sentence Transformers. *all-MiniLM-L6-v2*. URL: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

[WZ19] Jason Wei and Kai Zou. "EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6383–6389. URL: https://www.aclweb.org/anthology/D19-1670.