AMAZON POLARITY CLASSIFICATION

González Freixa, Júlia
Mirabent Rubinat, Guillem
Pericot i Masdevall, Pere
Vélez Peñaherrera, Daniela Alexandra

In this project we test different approaches for binary classification using Amazon Polarity dataset: random classifier, logistic regression with TF-IDF, a classifier based on distilBert with different techniques and final model based on a lighter model tinyBert with knowledge distillation.

The Amazon Polarity dataset, available from Huggingface, consists of 4 million Amazon reviews,you could take a look here. The training set contains 3.6 million reviews, while the test set comprises 0.4 million reviews. Due to the computational expense of processing such a large dataset, we decided to create a sample comprising 10% of the dataset, resulting in 360,000 reviews for training and 40,000 for testing. To generate a validation dataset for metric evaluation, we split the training data into an 85:15 ratio. Consequently, we work with a dataset consisting of 306,000 rows for training, 54,000 rows for validation, and 40,000 for testing. The label 0 represents negative reviews, and label 1 represents positive reviews. The dataset is a balanced classification set, which helps to avoid overfitting. Negative reviews tend to be longer than positive ones, with an average of 94 words per negative review and 88 words for a positive review.

PART 1: SETTING UP THE PROBLEM

Our objective is to find out which techniques could yield a better performing classification model of amazon reviews. A potential business case could be settled by the request of an API of a different, potentially smaller than Amazon, e-commerce marketplace. The API would allow the marketplace to integrate its platforms to analyze the received reviews in real-time in order to gain insights of their customer experiences and preferences, tailoring the product offers of their marketplace, etc. This service could be monetized though subscription based plans to different businesses that would pay for volume of processes reviews.

As a state of the art, *Mahammad Khalid Shaik Vadla, Mahima Agumbe Suresh and Vimal K. Viswanathan. (2024)* are using BERT models incorporating T5 and achieving 92% accuracy in an emotion classification task involving three classes. On one hand, the models used are more complex than in our case -where we are using distilBERT- but on the other hand, our project consists of a binary classification task while in the cited paper the task is multiclass, including the category neutral. In another paper, Xie, Q., Dai, Z., Hovy, E., Luong, T., & Le, Q. (2020). Unsupervised data augmentation for consistency training. Advances in neural information processing systems, the performance reached a 97,37% accuracy using BERTLarge model on an Amazon dataset of 3.6 million reviews.

We start by delving into the data and do some exploratory analysis. When analyzing unigrams, the word "great" was most frequently associated with positive reviews, while "bad" was prevalent in negative reviews. However, when considering bigrams, the phrase "highly recommended" was most common for positive reviews, and "waste of money" was the leading phrase for negative reviews. In another paper, the performance

The first model we ran was a random classifier, which, as expected, yielded an accuracy of 50.33%. For our baseline, we used TF-IDF with logistic regression. Prior to modeling, we applied preprocessing, which included converting to lowercase, removing stopwords, and applying stemming. Due to the computational intensity of stemming, we created a sample specifically for this process. Despite its simplicity, this baseline model performed significantly better than random guessing, achieving an accuracy of 78.38%.

PART 2: DATA SCIENTIST CHALLENGE

We use a distilBERT model with a maximum number of tokens of 128 because of the average length of our reviews around 90 and the maximum length is 272 so this is a good size in between.The model processes input IDs and attention masks, yielding hidden states of shape (None, 128, 768), indicating 768-sized hidden layers. It comprises 66,362,880 trainable parameters. We extract the hidden state of each sequence's first token (CLS) for classification. Applying dropout to mitigate overfitting is very important because we will be using the model with trainable=True, this allows the weights to be updated. A dense output layer, equipped with a sigmoid function and glorot uniform initializer, consists of 769 parameters: 768 for inputs plus one bias term. We define our optimizer with Adam and a conservative learning rate. For the loss we choose binary crossentropy. The model achieved an 81.40% accuracy using 32 labels, the scores are good for our seed, but they could have a lot of variability depending on the seed given the small amount of the sample we are taking from the training dataset.

For data augmentation, we use the Synonym Augmenter from nlpaug, so we created an object that substitutes words with their synonyms from WordNet which is a pre-prepared dictionary , specifying that each text should be augmented 5 times. As a result, we ended up with 32* 5 = 160 rows for this model. During the model training, data is fed into the model and processed in batches of 2 instead of 1 as we are already dealing with a bigger dataset. While the architecture remained the same, the addition of data augmentation improved the performance, yielding an accuracy of 85.24%.

Regardless of the number of observations in the model, Zero-Shot Learning with Large Language Models (LLMs) and a token limit of 1024, this model achieves a higher accuracy of 93.10% compared to previous ones. However, it is highly improbable that it would perform this well if, in reality, it wasn’t supposed to achieve 93% accuracy with a larger dataset. This is extremely weird given that at least those were being fine-tuned a little bit by the 32 labeled examples but this model has seen, theoretically 0 labeled examples from our dataset.Upon closer examination of the paper that introduced the model that we're using for the zero-shot classification, we think that the reason behind these unexpectedly good scores might be the fact that the zero-shot model's training has prepared it extensively for the task at hand and, as such, it performs this task much more like a trained model than like a pure zero-shot classification model. It probably saw some (or quite a lot of) "positive" "negative" classification training data or at least some other data that implied similar contexts. As a result, we have to take these extremely good scores with a pinch of salt.

For the model Data Generation with LLM we decided to create a total of 128 additional reviews. We used a positive and negative prompt for GPT2 as LLM. Here we increased the batch size from 2 to 4. In this case we shuffle like in previous cases, but we find it especially important because the added reviews might follow a positional pattern otherwise due to how we create them by concatenating strictly differentiated dataframes with the synthetic reviews from GPT2. The results are not as good as Synonym Augmentation but the scores are still an improvement from the basic 32 label model so we can consider the LLM synthetic reviews to at least add some predictive power to the model.

In spite of the outstanding results from the zero-shot model that we used before, we have decided to use a model that combines the data augmentation part with and without the use of an LLM.Adding these 32

additional reviews to the dataset then we will perform a larger data augmentation by 20x. The model´s performance declined, underperforming even the basic model trained on 32. The issue was its inability to correctly classify negative reviews, attributed to low quality examples generated by LLM, introducing noise rather than clear signals. In conclusion, we can say that the best performing model in this section was, without a doubt, the zero-shot-learning model, although it might be "cheating" in some way.

PART 3: STATE OF THE ART COMPARISON

In this part, the model is trained in 1%, 10%, 25%, 50%, 75% and 100% of the training data, only on one epoch each fitting, for running time reasons. We train the models at trainable=True to get the models fine-tuned to our dataset. The results show that for all metrics, the scores are found around 91% of accuracy in both models ran with 1% and 10%, around 93%of accuracy for models ran with 25% and 50% of the data and around 94% of accuracy for both models trained with 75% and 100% of the data. In the latter, the last run showed slightly lower metrics for the model trained with 100% than those of the model trained with 75% but the difference is minimal and it could change if fitting was done on more epochs. Overall, we observe that all these models -being trained with fine-tuning on- get better as more labeled examples are seen. In all cases, these are outperforming the models with any of the implemented techniques in part 2.

Given the results obtained in part 2, data augmentation using Synonym Augmenter on the 25% of the training set -for running time reasons- and it will be augmented twice its size, resulting in a dataset as large as 50% of our training set, equivalently to approximately 150.000 labeled reviews. The results show that this model performs as well as the model trained on 100% of the training data -around 94% of accuracy-.We could conclude that using Synonym Augmenter when having quality labeled examples proves to be a solid approach.

In conclusion, the models trained with 75% and 100% of the training set are within a 3% difference in accuracy of the state of the art. We have also observed that data augmentation can yield outstanding results, even if more investigation should be done into how sustainable is data augmentation, that is, depending on how much base data and the times that is being duplicated, the model might not keep the performance. We have also realized how important it has been to run the models with fine-tuning, changing significantly its performance when being trained with large datasets. Further work to become close to the state of the art should come with the full data set training and on more epochs, improving the model architecture to be able to capture more complex patterns, using the base model BERT instead of distilBERT.

PART 4: MODEL DISTILLATION

To perform model distillation, we have trained a student model with a lighter architecture (TinyBert) than the teacher model (based on pre-trained distilBERT). Specifically, it has 16,5 times less parameters and it is 15 times less weighty in terms of MB. As a teacher model, we take our best performing model, obtained by the base model trained on 100% of the data, done in exercise 3a. The student model has been trained with a loss function that penalizes differences between the student model predictions and the true labels (a usual binary cross entropy loss) as well as the differences between final predictions of the student model versus those of the teacher model to force the student model to mimic the teacher model behavior.

After fitting the student model, we observe that based on epoch time preprocessing, the student model runs approximately 3 times faster than the teacher model. In terms of metrics, one must consider that the

student model has been trained on 5 epochs while the best performing model has been trained only on 1 epoch. In any case, we see that overall, while the teacher model trained on a 100% of the data (306.000 data points) reaches around 94% of all metrics accuracy, precision, recall and f1, the student model reaches a 90% in all these metrics. However, we observed differences in tasks at which each model performs best. This gives room for further improvement and investigation. For instance, tuning parameters of the student model differently to be able to learn better or more from the teacher model. Moreover, other knowledge distillation techniques could be used such as multi-teacher models where the student learns from the decision-making process of more than one model, increasing the ability of capturing a wider range of patterns and potentially increasing performance.

REFERENCES

Mahammad Khalid Shaik Vadla, Mahima Agumbe Suresh and Vimal K. Viswanathan. (2024). Enhancing Product Design through AI-Driven Sentiment. MDPI. Read the paper

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. Read the paper

Xie, Q., Dai, Z., Hovy, E., Luong, T., & Le, Q. (2020). Unsupervised data augmentation for consistency training. Advances in neural information processing systems, 33, 6256-6268. Read the paper

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pre training for language understanding. Advances in neural information processing systems, 32. Read the paper

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. Read the paper

You, S., Xu, C., Xu, C., & Tao, D. (2017, August). Learning from multiple teacher networks. In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1285-1294). .Read the paper