ChatEGP Project: Newspaper Sentiment Analysis

Team 4: Philopateer Amgad, Youssef Amin, and George Sherif

May 26, 2023

Abstract

Sentiment analysis is one of the most prominent topics in Natural Language Processing (NLP) that aims to extract and identify determine emotions expressed by the author, whether positive, neutral, or negative. In this project, we aim to develop a sentiment analysis model that determines the sentiment inferred from Arabic financial news.

We divided the implementation into two parts. The first part includes training and fine-tuning the BERT model with an English dataset that consists of over 4500 headlines for financial news. After that, we evaluated the model using well-known metrics such as accuracy and f1-score. The second stage is translating a considerably smaller manually labeled Arabic dataset into English. The translated headlines are used to evaluate the model's performance in determining the sentiment of financial news headlines from another region.

Prior to each part discussed earlier, the texts used are processed to remove stop words, punctuation, and other non-relevant information. These texts are provided to BERT, which works by using a deep neural network to process and understand the meaning of the text. We decided to use BERT as our architecture as it has shown promising results in various NLP tasks, including sentiment analysis.

The final model allows us to understand the public sentiment toward finance news topics, which can be valuable for identifying some economic or financial trends at a given time. Additionally, this project highlights whether the news sentiment from the Arabic region can be identified using a model fine-tuned with news from a different region.

1 Introduction

Natural Language Processing (NLP) is a field of artificial intelligence that focuses on the interaction between humans and computers through natural language, not only by reading or processing them but also by understanding the implied meanings, and therefore be able to perform various tasks such as topic modeling, language modeling, sentiment analysis, and others. Nowadays, NLP is a hot topic of research, where scientists try to improve the complexities of the language models and enhance their training by using bigger and better datasets representing real life. An example of this could be the more powerful version of GPT-4 than its predecessor GPT-3, which was trained using much more parameters and larger datasets, making it able to perform tasks that are currently beyond the capabilities of GPT-3.

Our project focuses on sentiment analysis, one of several NLP applications. Sentiment analysis is the process of categorizing the sentiment of the person authoring the text by analyzing his feelings through words and categorizing the sentence as positive, neutral, or negative. We attempt to determine the sentiment of financial Arabic newspaper articles for this project. This issue is critical because it may serve as an indicator of the economy, financial status, or behavior during a certain time period. The model performing the sentiment analysis could be used to detect any difficulties facing that specific sector that yielded negative sentiments in its news.

To be able to perform the sentiment analysis, the project was divided into several phases. The first step was to analyze the dataset, in order to have an insight into the data we are working on. Data analysis allowed us to figure out the important features of the data to keep and the non-important ones to remove. This was feasible through the availability of a wide variety of libraries, such as *matplotlib*, *seaborn*, and the *tf-idf* [SJ04].

The step following the data analysis was to actually preprocess the data. During this phase, the *nltk* [BK09] library was used, which offered us several functions that perform the cleaning process. The preprocessing phase included tokenizing the texts, removing stop words, stemming, and standardizing the case of the texts to be all lowercase. The tokens are then appended again into sentence form to be in the correct format for the BERT tokenizer in the following step.

The third phase was to choose the architecture of our project and to train the model. We chose the BERT architecture [DCLT18], since it is a powerful tool that can perform several NLP tasks with high accuracy, including sentiment analysis. At first, we fine-tuned the BERT model with an English dataset, that is very similar to what we wanted to do with our Arabic dataset. The following step was to translate the Arabic text files using MarianMTModel [JDGD$^+$18], which is a Neural Machine Translation framework written in C++ language. These translated text files are then used to evaluate the effectiveness of the model to determine their sentiment, and whether the translation will affect the accuracy or not.

2 Data Analysis

Our chosen dataset is called SANAD (Single-Label Arabic News Articles Dataset). The articles were collected using Python scripts written specifically for three popular news websites: AlKhaleej, AlArabiya and Akhbarona. Articles are categorized into 7 topics as shown in Figure 1.

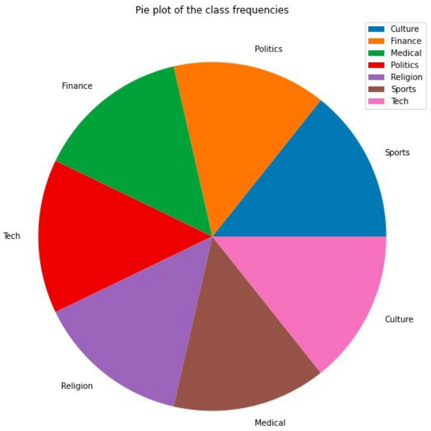


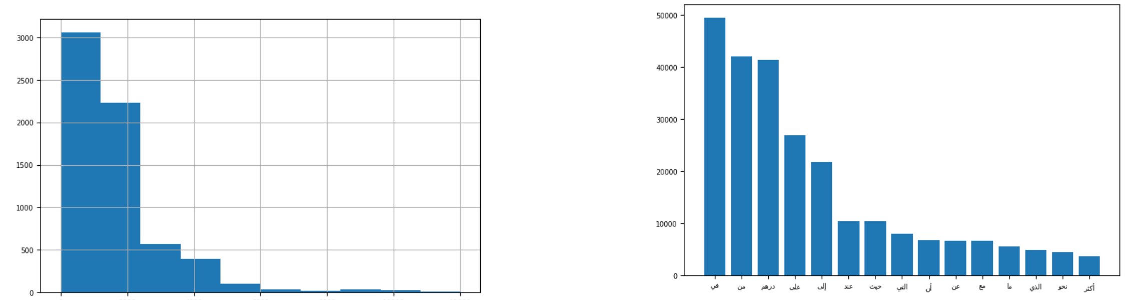
Figure 1: Complete Arabic dataset distribution

Our focus is solely on the Finance category. In the following lines, we aim to demonstrate our knowledge of the dataset through analysis and visualizations. Our corpus consists of 6500 individual text files. The distribution of the length of each file in characters is displayed in Figure 4(a), showing that almost half of the dataset has a length of 1000 characters. The median of word lengths is 5. A very important application of data analysis is determining stop words. The library nltk [BK09] has some universal Arabic stop words. These are mainly propositions (حروف الجر و اسماء الوصل) and punctuation. These of which appearing in our dataset are displayed in 2(b). This is consistent with the data obtained from calculating Term Frequency(tf), as many of the highest scoring words in the tf table 1 are those which are in the nltk list.

Table 1: TFs

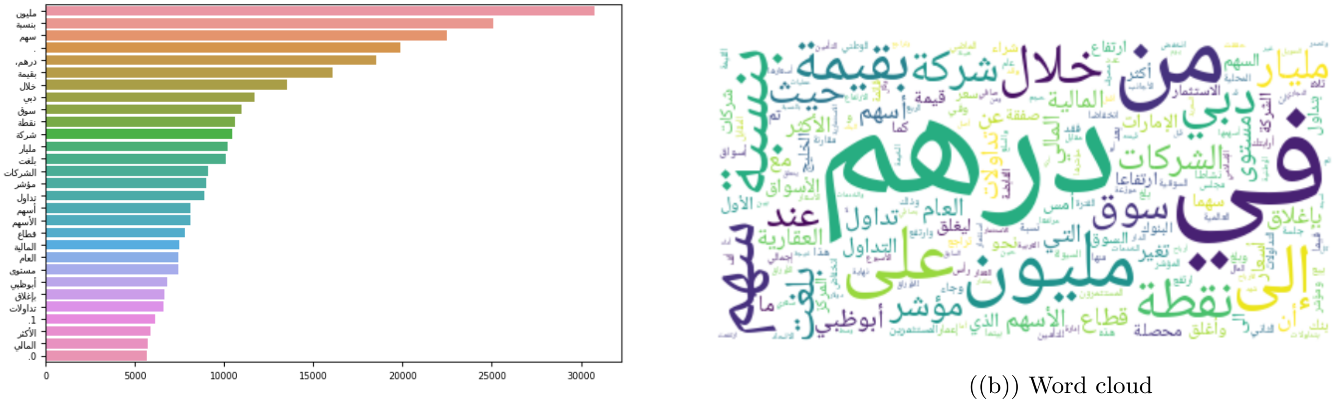
Word	في	من	على	إلى
Sum of TFs	90.61	68.17	41.14	33.79

Another representation of the most common words is the word cloud as in figure 4(b). Here we can see a more domain-oriented possibility for stop words removal. Words of currencies (دينار,درهم), stocks سهم or orders of magnitude (مليار,مليون) could be regarded as stop words in the domain of finance. These were obtained after removing the universal stop words and evaluating the tf once more. Using bigrams could also be inferential [WM12]. Using the library sci-kit learn, the method



((a)) Length of files in Characters

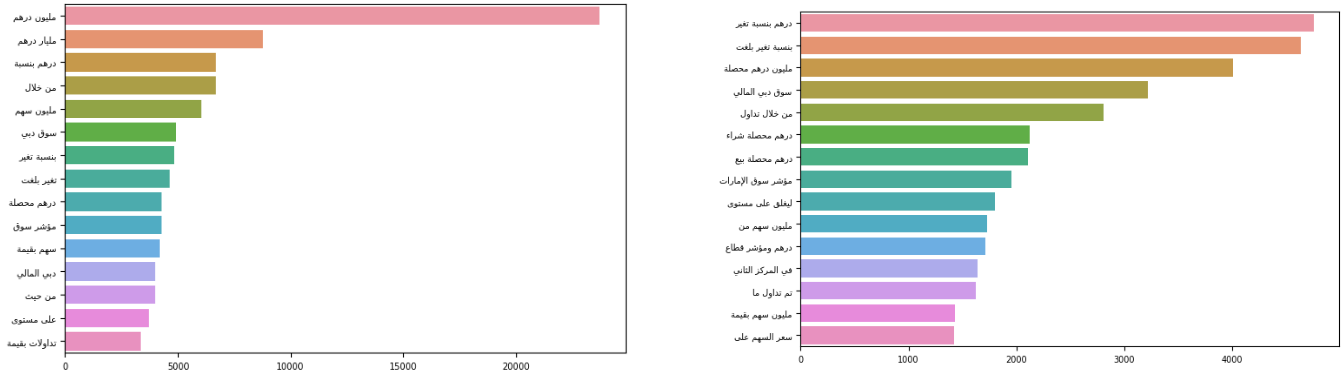
((b)) Frequency of Universal Stop Words in our dataset



((a)) Domain Specific Stop words

((b)) Word cloud

CounVectorizer [PVG+11] is used to calculate bigrams and trigrams. Using this data, we are able to make more informed decisions in preprocessing and later in hyper-parameter tuning. As a simple example, another word cloud is shown 5 highlighting the most significant words after removing some of the stop words. Which is also consistent with the results obtained from tf-idf. The tf-idf enhances data obtained from the tf, as many of the words with a high tf score are insignificant (stop words).



((a)) Bigram frequencies

((b)) Trigram frequencies

Figure 4: N-grams

3 Data Preprocessing

Preprocessing of the data is the part concerned with cleaning the data. By cleaning the data we mean removing any unnecessary information that may result in confusing our model. To do this part we

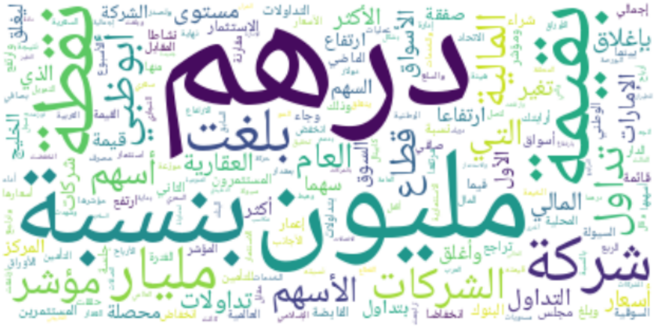


Figure 5: Word Cloud after removing some stop words

Table 2: TF-IDFs

Word	بلغت	تداول	بقيمة	بنسبة
Sum of TF-IDFs	201.05	213.18	338.13	481.95

used the help of the NLTK library [BK09], which is a very well-known library that has lots of utilities that we can use. The first step we did is to tokenize the sentence coming from the dataset using the *nltk.word_tokenize()* function, in order to be easily able to work on the word level. The second step was doing the stemming part which is concerned with returning words to their origin, and also applying lowercase to all words in a sentence. For this part, we used the *PorterStemmer()* function. Finally, we removed stop words and punctuation using the set of stop words and punctuation known in the English dictionary, the reason we applied this step is that those stop words are not of great impact on the sentence, so we removed them to consider only words that matter.

4 System Architecture

Due to its countless applications and impact on the Natural Language Processing field, we chose BERT as our model for the sentiment analysis task.

BERT is a machine learning framework for NLP, which stands for Bidirectional Encoder Representations from Transformers. BERT employs a bidirectional transformer design, which allows it to gather contextual information from both previous and following words in a given text. The process of training BERT allows it to deeply understand the language, including sentiment-related patterns and semantic relationships.

Although BERT was trained on publicly available text from the internet and data from BooksCorpus and English Wikipedia, it still requires fine-tuning in order to make it topic-specific. Fine-tuning means doing adjustments to the model in order to give better results. The process of fine-tuning includes gathering a labeled dataset, which is then split into training, validation, and testing sets. These sets are then converted to match BERT's input form. After running the model with the new dataset, evaluation takes place to check if the model is performing well or not on this specific topic.

This architecture, BERT, has shown awesome results in sentiment analysis in previous tasks. For example, on the Movie Reviews (SST-2) dataset, accuracy of the sentiment analysis was 93.7% [BRT]. So we decided to use this architecture as well for our project.

5 Methodology

As mentioned before, our aim is to analyze the sentiment of the Arabic news headline dataset. Unfortunately, the Arabic dataset was not labeled [ARD], thus we followed another approach 6. Rather than labeling the whole dataset in order to fine-tune an Arabic version of BERT, we decided to fine-tune a normal English BERT model with a ready-labeled English dataset. Afterward, we manually label a portion of the Arabic dataset and translate it. Using the MarianNMT [MNM] framework, we could

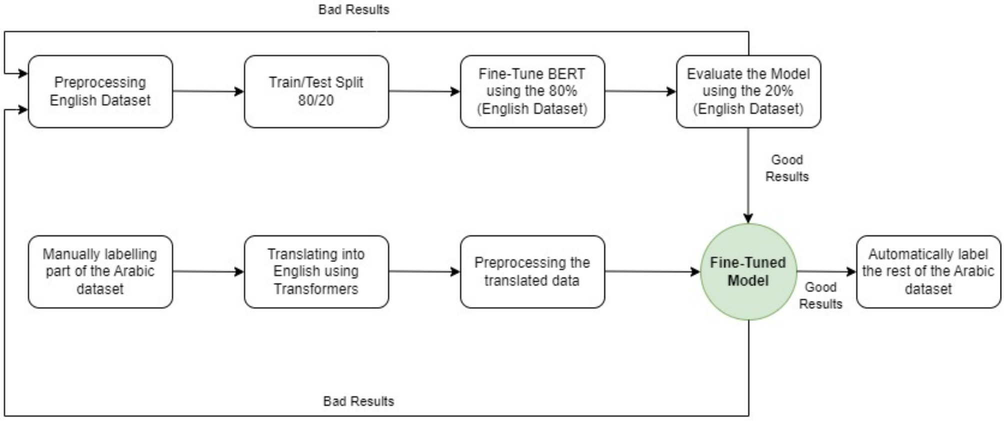

Bad Results
Preprocessing English Dataset
Train/Test Split 80/20
Fine-Tune BERT using the 80% (English Dataset)
Evaluate the Model using the 20% (English Dataset)
Good Results
Manually labelling part of the Arabic dataset
Translating into English using Transformers
Preprocessing the translated data
Fine-Tuned Model
Good Results
Automatically label the rest of the Arabic dataset
Bad Results

Figure 6: Methodology

translate the Arabic dataset to an English one, using OPUS-MT [OPU] Neural machine translation model. The library used was the Helsinki-NLP [TT20]. This translated portion of the Arabic dataset was then used to validate the fine-tuned BERT model, to ensure that the model's accuracy is acceptable. In order to ensure high accuracy in the model, the English dataset that we used was covering precisely the same topic as the Arabic dataset, however for a different region.

6 Procedure

To start our sentiment analysis tasks, we followed three different approaches to determine the best model to proceed with. These three approaches are as follows:

1. Using the English dataset as is.

2. Preprocessing the English and the translated Arabic datasets.

3. Doing data augmentation to increase the number of entries.

In all three previous approaches, the dataset was balanced. In other words, the number of negative, positive, and neutral entries where exactly the same in the training phase. The total number of training entries was 1,812 entries for the first two approaches and 3,624 entries for the data augmentation approach.

6.1 The 3 Approaches

6.1.1 Approach 1: Without Preprocessing

In this approach, we decided to pass the dataset as is to BERT. As mentioned in several QA forums [NOPb, NOPa], BERT model works better with raw text, or at least preprocessing will not give better results. One reason for that is because of the Byte-Pair Encoding (BPE) technique that BERT uses. BPE is a tokenization technique that breaks the word into smaller words. For example, the word *running* to *run + ##ing*. This is similar to what stemming/lemmatization does. Another reason not to preprocess the dataset is that BERT's attention mechanism focuses only on words that are related to the topic, not on the words that are frequently repeated.

6.1.2 Approach 2: With Preprocessing

For another reference [TRI], it was mentioned that in order to figure out which is better for our task, we have to try all options out. In this approach, we tested several types of preprocessing, such as stemming/lemmatization, removing punctuation, lowering the case for all words, and removing stop words. The new form is now fed to the model for fine-tuning

6.1.3 Approach 3: With Data Augmentation

Fine-tuning gives better results with larger datasets. That is why we tried another approach which is increasing our dataset size using data augmentation. The technique we used in the data augmentation was adding paraphrased sentences into our dataset. For this specific task, we used the Pegasus language model, which is designed for text generation tasks. After implementing this idea, the dataset size was doubled from 1,812 entries to 3,624 entries.

6.2 Training the Model

For each of the 3 approaches, we chose to use a batch size of 8 and train the model for 16 epochs. The data was first encoded using BertTokenizer, where the *max_length* of the sentence was *150* tokens and the *pad_to_max_length* parameter was set to *True*. The following step was to prepare the data to match the BERT format using the *TensorDataset* library. AdamW optimizer was used to make sure that weight decay is applied correctly, which reduces the chances of overfitting and gives better final results. In addition to the optimizer, the *get_linear_schedule_with_warmup()* function was called to adjust the learning rate during training. This allows the model to stabilize and gradually adapt to the training data. The model then trains using the final format of the dataset for 16 epochs, where the accuracy and the training loss for each epoch are calculated to choose the best epoch for each approach. The best epoch acts as the fine-tuned model, which will then be validated using the Arabic data.

6.3 AR-EN Translation

As mentioned before, the main of this project was to be able to create a pipeline for doing sentiment analysis for Arabic News. After training the model and checking that it achieves satisfactory accuracy on the English dataset, it was the turn to validate using the Arabic data. The Arabic dataset was translated using the Opus-MT [OPU], which did a good job translating the sentences after visual inspection. For this part, we only used 85 entries to be translated. Although 85 entries are a very small amount to validate, this matches precisely the project's aim, which is creating a model to define the sentiment analysis for Arabic news without having labeled data. Again, everything before was repeated again, including preparing the data to match BERT's format. For the preprocessing approach, the translated data were preprocessed before using it to validate the chosen model.

7 Results

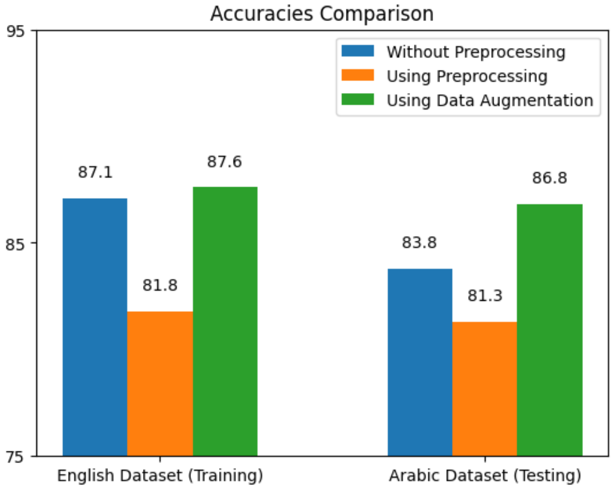


Figure 7: The English dataset for training and the Arabic dataset for validation

Figure 7 shows each approach and its accuracy, in both the training phase and the testing phase. For the training phase, the figure shows that using data augmentation had the best accuracy of 87.6%. The second-ranked approach was using the data as is, without any preprocessing. It scored an accuracy of 87.1%. The worst approach scored 81.8%, which was the preprocessing approach. As discussed before, BERT prefers raw data. BERT uses some similar techniques to perform preprocessing thus we do not need to do anything before.

On the right-hand side of figure 7, we can see that using the data-augmented model scored the highest accuracy as well, scoring an 86.8%, which is slightly lower than the training accuracy for the same model. Just after it, the model with no preprocessing scored an 83.8%, and in last place comes the model with preprocessing at an accuracy of 81.3%

7.1 Approach 1: Without Preprocessing

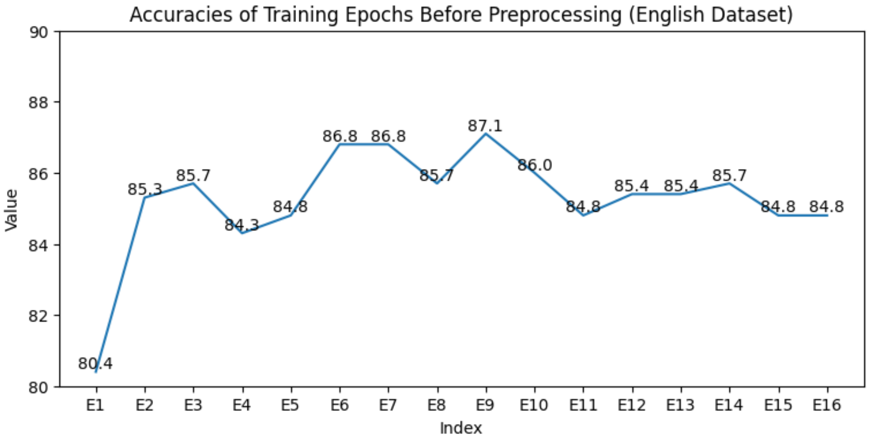


Figure 8: Accuracy of the 16 Epochs without doing any preprocessing

Out of the 16 epochs of training the model, the 9th epoch scored the highest accuracy at 87.1%. The model started at 80.4% during the first epoch and reached the peak in the 9th epoch, and then kept falling again till reaching 84.8% in the final epoch. We believe that training the model more than 16 epochs will not do a great difference, or even no difference at all to the peak accuracy.

7.2 Approach 2: With Preprocessing

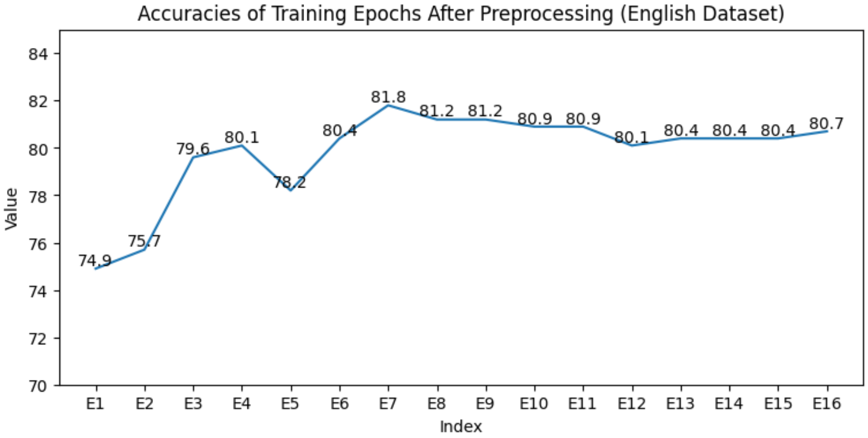


Figure 9: Accuracy of the 16 Epochs after doing preprocessing

Using preprocessing in figure 9, the average accuracy is lower than the first approach. The lowest accuracy was at the very beginning with 74.9%, reaching its maximum at the 7th epoch with an accuracy of 81.8%. Following the 7th epoch, the model performed approximately the same, as it ended the whole training phase with an accuracy of 80.7%. Training the model for more epochs might have yielded different peak accuracy, however, for the standardization of the experiment, we fixed the number of epochs to be only 16.

7.3 Approach 3: With Data Augmentation

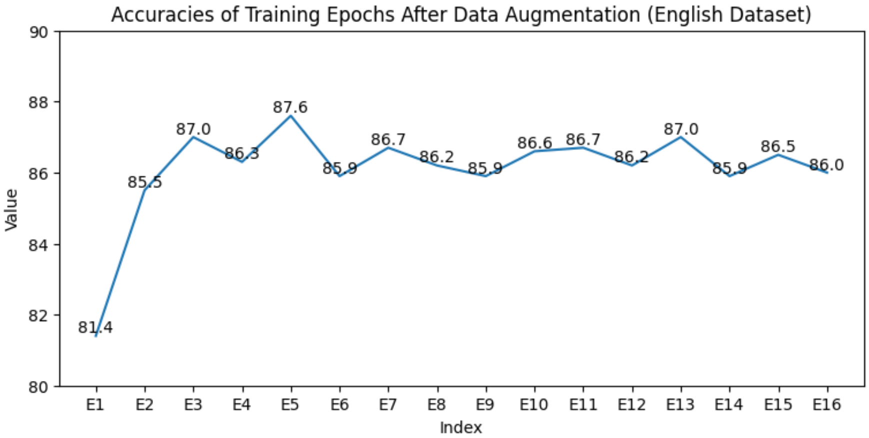


Figure 10: Accuracy of the 16 Epochs after doing data augmentation

For the final approach, figure 10 shows that epoch 5 scored the highest accuracy during training, scoring an 87.6%. The model started at 81.4% on its first epoch and ended up at 86% on the final one. We believe that there is a chance of a slight improvement if we increased the number of epochs.

8 Conclusion, Limitations, and Future Work

8.1 Conclusion

To conclude this report, our project aimed at defining a tool to analyze the sentiment of Arabic finance news headlines. We faced some problems such as the unlabeled Arabic dataset, which we solved using the available pre-trained BERT English model and a helper language translating model. Using these tools, we fine-tuned BERT on a very similar English dataset that was labeled and then validated the model using some of the Arabic data after translating them.

We divided our work into 3 approaches, one using raw data, one using preprocessed data, and the last one using data augmentation. The data augmentation showed the best results during training and also during validation with the translated0 Arabic data. It scored an accuracy of 87.6% during training and 86.8% during validation. Raw data scored an 87.1% in training and 83.8% in validation. Preprocessed data scored the least accuracy at 81.8% during training and 81.3% during validation.

8.2 Limitations

Sentiment analysis is usually a subjective topic, where some sentences may have more than one sentiment, depending on who you are. For example, in the Arabic dataset, there was a sentence that might be understood by the news as neutral, however from the employee's point of view it might be positive. The sentence was basically told that the company decided to distribute the yearly dividends. For us, we labeled these kinds of sentences as positive (employee perspective), however, the model decided its neutral.

Another limitation is that the validation dataset was too small. Validating a model with only 85 entries might not generalize or give the most accurate results.

8.3 Future Work

For future work, the limitations can be easily overcome by increasing the number of validation data and providing a higher-quality English training dataset. If accuracy in the future increases by 95%, we believe this pipeline could be used to manually label datasets without human intervention, or with a minimum one. Since the world is evolving towards computerizing everything, we believe our project is of high importance.

References

[ARD] Arabic news articles dataset. *https://www.kaggle.com/datasets/haithemhermessi/sanad-dataset.*

[BK09] Edward Loper Bird, Steven and Ewan Klein. *Natural Language Processing with Python. O'Reilly Media Inc.* 2009.

[BRT] Nkhata, g. (2022). movie reviews sentiment analysis using bert. *https://scholarworks.uark.edu/etd/4768.*

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[JDGD+18] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. Marian: Fast neural machine translation in C++, July 2018.

[MNM] Neural machine translation framework using c++. *https://marian-nmt.github.io/.*

[NOPa] Do you need to preprocess text for bert? *researchgate.net/post/Do_you_need_to_preprocess_text_for_BERT.*

[NOPb] Using trained bert model and data preprocessing. *stackoverflow.com/questions/63979544/using-trained-bert-model-and-data-preprocessing.*

[OPU] Neural machine translation model for translating from arabic (ar) to english (en). *https://huggingface.co/Helsinki-NLP/opus-mt-tc-big-ar-en.*

[PVG+11] 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. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[SJ04] Karen Spärck Jones. Idf term weighting and ir research lessons. *Journal of documentation*, 60(5):521–523, 2004.

[TRI] Why there is no preprocessing step for training bert? *datascience.stackexchange.com/questions/113359/why-there-is-no-preprocessing-step-for-training-bert.*

[TT20] Jörg Tiedemann and Santhosh Thottingal. OPUS-MT – building open translation services for the world. pages 479–480, November 2020.

[WM12] Sida I Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 90–94, 2012.