HELWAN UNIVERSITY
Faculty of Computers and Artificial Intelligence
Computer Science Department

Sentiment Analysis For Arabic Language Using Text And Emoji

A graduation project dissertation by:

[Omar Hany Ahmed Samy (20170360)]

[Ahmed Ismail Mohamed (20170011)]

[Bassem Magdi Mahmoud (20170147)]

[Essam Adel Ali (20170326)]

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computers & Artificial Intelligence, at the Computer Science Department, the Faculty of Computers & Artificial Intelligence, Helwan University

Supervised by:

Dr. Soha Ahmed Ehssan

July 2021

Abstract

Sentiment analysis (SA) is a natural language processing (NLP) application that aims to analyze and identify sentiment within a piece of text. Arabic SA began to receive more attention in the last decade with many approaches showing some effectiveness for detecting sentiment on multiple datasets. Whereas there are some surveys summarizing some of the approaches for Arabic SA in literature, most of these approaches are reported on different datasets, which makes it difficult to identify the most effective approaches among those. In addition, those approaches do not cover the recent advances in NLP that use transformers. Transformer-based language models have been a revolutionary step for natural language processing (NLP) research. Those models, such as BERT, GPT and ELECTRA, led to state-of-the-art performance in many NLP tasks.

Most of these models were initially developed for English and other languages followed later. Recently, several Arabic-specific models started emerging. However, there are limited direct comparisons between these models.

The objective is to evaluate the performance of 3 of BERT based models on Arabic sentiment with emoji involved, since BERT based models have proven to be very efficient at language understanding, provided they are pre-trained on a very large corpus. Our results show that the models achieving the best performance are those that are trained on only Arabic data, including dialectal Arabic, and use a larger number of parameters, such as the recently released MARBERT. However, we noticed that MARBERT is one of the top performing models while being much more efficient in its computational cost

ACKNOWLEDGEMENT

We would like to thank our supervisor of this project **Dr. Soha Ahmed Ehssan** for the helpful assistance, support and guidance. Her willingness to motivate us, her insist, and her patience in times of despair kept us going and willing to perform better. We also would like to give our thanks for her huge effort in continuous contacting and providing us with all what we needed regarding information, recent updates for the graduation project needs and her beneficial ideas.

TABLE OF CONTENT

CHAPTER 1: INTRODUCTION

1.1 INTRO

Since natural language processing has been in rapid evolution the past decade, its application on language domains and how to define the human nature of communication using speech and text. It has become very handy for applications like Siri which is utilizing various artificial imitation forms of communication to understand human language and try to reply to the speaker or perform the task/s that the virtual assistant was commanded to do.

All of this thanks to natural language processing making use of neurolinguistic programming that employs various artificial intelligence algorithms in order to imitate/mimic human behavior, by using various interchangeable machine learning modules to take a huge input of data, process it and train another new module capable of understanding predefined patterns and semantics

Natural language processing also makes use of linguistics to draw semantic meaning from text or speech which in return derives the linguistics that processes the logical structure of sentences to identify the most relevant elements in text and understand the context that the sentence or speech came in and how to relate words or sets of words to each other to make it possible for machines to understand how the connection between words takes place to give any form of meaning that relates to how humans communicate and whether this knowledge is applicable or no.

As the daily usage of social media increases, and the rapid generation of text based content on these social media platforms continues to grow, natural language processing is being used to process the various forms of data and helping an excessive amount of applications to automate and sustain a substantial amount of information that could be helpful and beneficial.

Fragments of the daily generated data can be very tricky to deal with. Occasionally some chunks of this data are very sensitive for normal techniques to process. Heterogeneity of generated text based forms of data like tweets on twitter can contain some emojis. Emojis have seen heavy usage on social media websites over the last decade, all devices now support emojis in some variety or another. Yet, it is so challenging to draw any little to no semantic meaning from them since they are so sensitive and tricky to operate on when they are used in a strict or unusual context to give a semantic meaning to the sentence including its operating in different contexts and environments.take data from its raw, siloed and normalized source state and transform it into data that’s joined together, dimensionally modeled, de-normalized, and ready for analysis.

In order to understand what semantic value an emoji gives to a sentence in a given usual or unusual context, an immense amount of data has to be processed. First of all, data containing a variety of sentences that does not accommodate any emojis has to be collected along with data enriched by emojis. Then preprocessing on data has to take place since data-gathering methods are often loosely controlled, resulting in out-of-range values, some records of this data might not even be correct and some of them have no true meaning.

After preprocessing has taken place on data in order to clean it to the best possible form in preparation for processing on it to take place, some operation has to take place in order to draw semantics from a given emoji within a defined context. Data transformation is the process in which you take data from its raw, siloed and normalized source state and transform it into data that’s joined together, dimensionally modeled, denormalized, and ready for analysis.
Without the right technology stack in place, data transformation can be time-consuming, expensive, and tedious. Nevertheless, transforming data will ensure maximum data quality which is imperative to gaining accurate analysis, leading to valuable insights that will eventually empower each semantic decision. Since computers do not understand plain text or strings data is transformed to make it better-organized.

Transformed data may be easier for both humans and computers to use. Properly formatted and validated data improves data quality and protects applications from potential landmines such as null values, unexpected duplicates, incorrect indexing, and incompatible formats. then a combination of artificial neural networks classes starts to map this data

Transformer Neural Networks is the actual first stage processing on data in order for computers to understand the nature of the textual analysis a module has to achieve. After the transformation is successfully done on data, it's time for attention models to take place. Attention models, or attention mechanisms, are input processing techniques for neural networks that allows the network to focus on specific aspects of a complex input, one at a time until the entire data is well categorized to the best form possible.

Attention is a technique that mimics cognitive attention. The effect enhances the important parts of the input data and fades out the rest, the thought being that the network should devote more computing power on that small but important part of the data. Which part of the data is more important than others depends on the context and is learned through training data by gradient descent. "Attention is all you need."

Bidirectional Encoder Representations from Transformers is a pre-training mechanism; it makes use of Transformers, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its basic form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary.

BERT is used when a pre-trained neural network produces word embeddings which are then used as features in NLP models in this case emojis are the subject. It analyzes the entire sequence of words at once. This allows the model to learn the context of a word based on all of its surroundings text.

1.2 BACKGROUND

1.2.1 Recurrent Neural Network (RNN):

RNNs are a powerful and robust type of neural network, and belong to the most promising algorithms in use because it is the only one with an internal memory. because of their internal memory, RNN's can remember important things about the input they received, which allows them to be very precise in predicting what's coming next. This is why they're the preferred algorithm for sequential data, speech, text, financial data, audio, video, weather and much more. Recurrent neural networks can form a much deeper understanding of a sequence and its context compared to other algorithms. Recurrent neural networks (RNN) are a class of neural networks that are helpful in modeling sequence data. Derived from feedforward networks, RNNs exhibit similar behavior to how human brains function. RNN has two inputs: the present and the recent past.

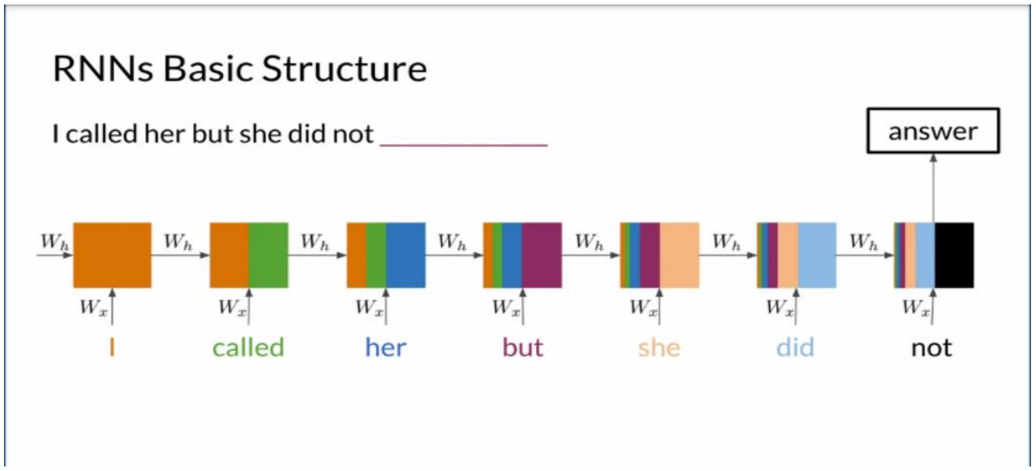


Figure 2.1: RNNs Basic Structure

To predict what the next word RNN stores all the previous words so RNN is not useful in all cases like what will happen if I have a very large data? here I will need more memory to store. so it is not effective in our case.

1.2.2 Long Short-Term Memory (LSTM):

Long Short Term Memory networks are a special kind of RNN, capable of learning long-term dependencies. They were designed to work well on many types of problems. Their goal is to avoid the long-term dependency problem; done by remembering information for a long time and thus they can learn easily.

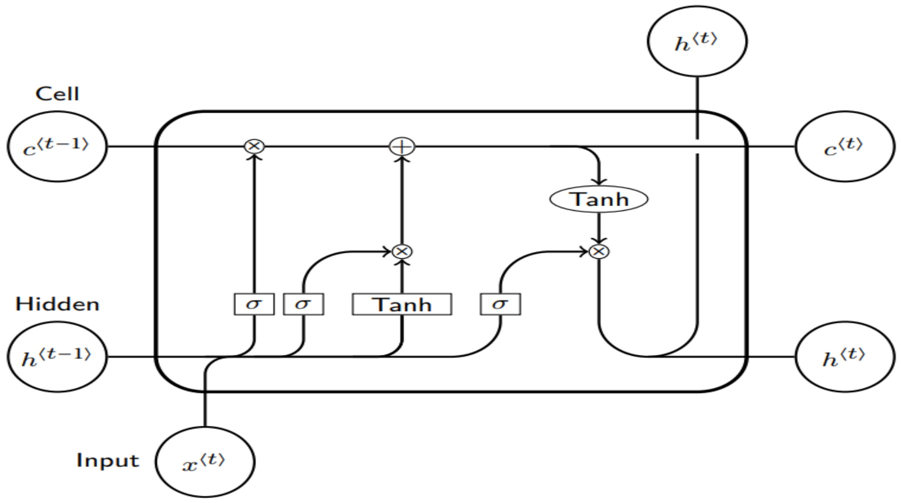


Figure 2.2: LSTMs Architecture

As shown the LSTMs consist of:

- Cell state
- Hidden state

Cell state: act as a transport highway that transfer relative information all the way down the sequence chain. “Memory of the network”

Hidden state consist of 3 gates

- Forget gate
- Input gate
- Output gate

LSTM’s initial step is to filter the input sequence entering the neural network using a “forget gate” which decides which parts of the sequence it needs to remember. The “forget gate layer” has a sigmoid layer applied on $ht-1$(previous hidden state) and xt(input sequence) and outputs a value between 0 and 1, the closer to 0 means to forget, and the closer to 1 means to keep.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

The next step is to decide what new information we are going to store in the cell state. Firstly, another sigmoid layer called the “input gate layer” decides which values we’ll update. Secondly, a tanh layer creates a vector of new candidate values, Ct (new cell state), that could be added to the state. In the next step, we’ll combine these two to create an update to the state.

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) c_t \quad = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

It’s now time to update the old cell state, $Ct-1$, into the new cell state Ct. The previous steps already decided what to do, We multiply the previous state by ft(forget gate output), removing the invaluable information that we decided to forget earlier. Then we add it*C t. This is the new candidate values, scaled by how much we decided to update each state value.

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

Finally, we need to decide what we’re going to output. This output will be based on our cell state but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we’re going to output. Then, we put the cell state through $tanh$ (to push the values to be between (−1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the selected values.

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) h_t \quad = o_t \circ \sigma_h(c_t)$$

One of the lSTMs networks advantage is it learns when to forget.
But also it is slow to train so at the large datasets it doesn’t respect the time consuming.

1.2.3 Gated Recurrent Unit (GRU):

GRUs are improved versions of a standard recurrent neural network. They were designed to solve the vanishing gradient problem by using two gates called “update gate” and “reset gate”; those two gates are two vectors filtering the amount of information passed to the output. What makes them special is that they are trained to keep information from the entering sequence for a long time.

- Update gate: The update gate helps the model to determine how much of the past information (from previous time steps) needs to be passed along to the future.

  When xt(Input Sequence) is plugged into the network unit, it is multiplied by its weight $W(z)$. The same goes for $h(t-1)$ which holds the information for the previous t-1 units and is multiplied by its weight $U(z)$. Both results are added together and a sigmoid activation function is applied to limit the result between 0 and 1.

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$

- Reset Gate: This gate is used from the model to decide how much of the past information we need to forget. This formula is the same as the one for the update gate. The difference comes in the weights and the gate’s usage.

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$

- Current memory content: We introduce a new memory content that will use the reset gate to store the relevant information from the past.

$$h_t = tanh(W x_t + r_t \circ U h_t - 1)$$

• Final memory at the current time step: The network needs to calculate the ht vector which holds information for the current unit and passes it down to the network. To do that we need the update gate. It determines what to collect from the current memory content ht and what from the previous steps $h(t-1)$.

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

The GRU neural network is a special variant of the recurrent neural network, which can maintain a longer-term information dependence and has been widely used in industry However, GRU still has the disadvantages of slow convergence and low learning efficiency.

CHAPTER 2: RELATED WORK

Maha Hekial et al. [1] have used an ensemble model, combining Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) models, to predict the sentiment of Arabic tweets. Their model achieves an F1-score of 64.46%, which outperforms the state-of-the-art deep learning model’s F1-score of 53.6%, on the Arabic Sentiment Tweets Dataset (ASTD).

Nora Al-Twairesh et la. [2] presented the details of collecting and constructing a large dataset of Arabic tweets. The techniques used in cleaning and pre-processing the collected dataset are explained. A corpus of Arabic tweets annotated for sentiment analysis was extracted from this dataset. The corpus consists mainly of tweets written in Modern Standard Arabic and the Saudi dialect. The corpus was manually annotated for sentiment, it contains 17,573 tweets labelled with four labels for sentiment: positive, negative, neutral and mixed.

Hazem Hajj et la. [3] pre-trained BERT specifically for the Arabic language in the pursuit of achieving the same success that BERT did for the English language. The performance of AraBERT is compared to multilingual BERT from Google and other state-of-the-art approaches. The results showed that the newly developed AraBERT achieved state-of-the-art performance on most tested Arabic NLP tasks. The pretrained araBERT models are publicly available on github.com/aub-mind/araBERT hoping to encourage research and applications for Arabic NLP

Abdulmohsen Al-Thubaity et la. [4] proposed the Saudi dialect sentiment lexicon (SauDiSenti) for sentiment analysis of Saudi dialect tweets. SauDiSenti comprises 4431 words and phrases manually extracted by two Saudi annotators from previously labelled datasets of Saudi dialect tweets, they evaluated the performance of SauDiSenti on a new dataset labelled by two annotators, comprising 1500 tweets evenly distributed over three classes: positive, negative, and neutral.

Mazen El-Masri et la. [5] presented a new tool that applies sentiment analysis to Arabic text tweets using a combination of parameters. Those parameters are (1) the time of the tweets, (2) preprocessing methods like stemming and retweets, (3) n-grams features, (4) lexicon-based methods, and (5) machine-learning methods. Users can select a topic and set their desired parameters. The model detects the polarity (negative, positive, both, and neutral) of the topic from the recent related tweets and display the results. The tool is trained with 8000 randomly selected and evenly-labelled Arabic tweets. The experiments show that the Naive Bayes machine-learning approach is the most accurate in predicting topic polarity.

Abdul Munem Nerabie et la. [6] addressed those limitations by proposing a novel Arabic social media text corpus that is enriched with complete PoS information, including tags, lemmas, and synonyms. The proposed corpus constitutes the largest manually annotated Arabic corpus to date, with more than 5 million tokens, 238,600 MSA texts, and words from Arabic social media dialect, collected from 65,000 online users' accounts. Furthermore, the proposed corpus was used to train a custom Long Short-Term Memory deep learning model and showed excellent performance in terms of sentiment classification accuracy and F1-score. The obtained results demonstrate that the use of a diverse corpus that is enriched with PoS information significantly enhances the performance of social media analysis techniques and opens the door for advanced features such as opinion mining and emotion intelligence

Muhammad Abdul-Mageed et la. [7] they compared their models to four other models: mBERT, XLMRBase, XLM-RLarge, and AraBERT. They note that XLM-RLarge is ~ 3.4× larger than any of our own models (~ 550M parameters vs. ~ 160M). They offer two main types of evaluation: on (i) individual tasks, which allows them to compare to other works on each individual dataset (48 classification tasks on 42 datasets), and (ii) ARLUE clusters (six task clusters).

Jacob Devlin et la. [8] improved the fine-tuning based approaches by proposing BERT: Bidirectional Encoder Representations from Transformers. BERT alleviates the previously mentioned unidirectionality constraint by using a “masked language model” (MLM) pre-training objective, inspired by the Cloze task (Taylor, 1953). The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked arXiv:1810.04805v2 [cs.CL] 24 May 2019 word based only on its context. Unlike left-toright language model pre-training, the MLM objective enables the representation to fuse the left and the right context, which allows us to pretrain a deep bidirectional Transformer. In addition to the masked language model, they also use a “next sentence prediction” task that jointly pretrains text-pair representations.

Sarah Alhumoud [9] analyzed and visualized the influence of coronavirus (COVID-19) using machine learning and deep learning methods to quantify the sentiment shared publicly correlated with the actual number of cases reported over time. On the analysis of 10 Million Arabic tweets, results show that deep learning techniques using an ensemble model outperformed machine learning using SVM with an accuracy of 90% and 77% respectively. It also outperformed the individual deep learning models.

Khaled B. Shaban et la. [10] used a deep learning approach for the sentiment classification problem on Arabic text. Three architectures were proposed and derived for: DNN, DBN and Deep Auto Encoders. The features vector used the sentiment scores from ArSenL lexicon. LDC ATB dataset was used to evaluate the models, comparing their accuracy and F1 scores. It was found that, Deep Auto encoder model gives better representation of the input sparse vector. They also proposed a forth model, RAE, which was the best deep learning model according to our results, although it requires no sentiment lexicon. The results show around 9% improvement in average F1 score over the best reported results in literature on the same LDC ATB dataset in the sentiment classification task for Arabic.

CHAPTER 3: PROPOSED MODELS

3.1 DATASET

We all know that the datasets are very important especially in Bert so to make BERT handle a variety of down-stream tasks; our input representation is able to unambiguously represent both a single sentence and a pair of sentences in one token sequence. Throughout this work, a “sentence” can be an arbitrary span of contiguous text, rather than an actual linguistic sentence.

A “sequence” refers to the input token sequence to BERT, which may be a single sentence or two sentences packed together so we use WordPiece embedding but this way is not very efficient and it takes some time to handle this kind of data and the cleaning that where the pre-trained model came with this that focus on applying transfer learning by fine-tuning large pretrained language models for downstream NLP/NLU tasks with a relatively small number of examples, resulting in notable performance improvement for these tasks. This approach takes advantage of the language models that had been pre-trained in an unsupervised manner (or sometimes called self-supervised).

However, this advantage comes with drawbacks, particularly the huge corpora needed for pre-training, in addition to the high computational cost of days needed for training (latest models required 500+ TPUs or GPUs running for weeks. These drawbacks restricted the availability of such models to English mainly and a handful of other languages. To remedy this gap, multilingual models have been trained to learn representations for +100 languages simultaneously, but still fall behind single-language models due to little data representation and small language - specific vocabulary.

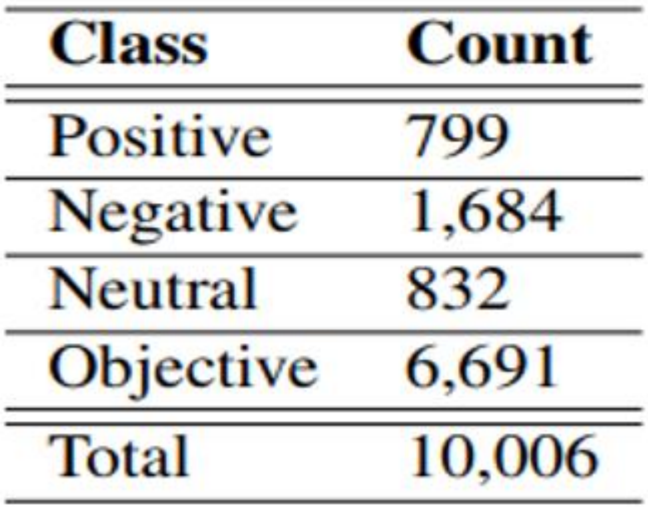
In this paper we used two datasets:

- 40000-Egyptian-tweets (Ammar Mohammed et la. 2019)
- and 'SarcasmAndSentimentDetectionInArabic' which was a competition of workshop on 19-20 April, 2021 that focus on sarcasm and semantic analysis of Arabic language which is **ArSarcasm.**

3.1.1 ArSarcasm:

ArSarcasm is a new Arabic sarcasm detection dataset. The dataset was created using previously available Arabic sentiment analysis datasets ((SemEval_2017)and (ASTD)) and adds sarcasm and dialect labels to them. The dataset contains 10,547 tweets, 1,682 (16%) of which are sarcastic.

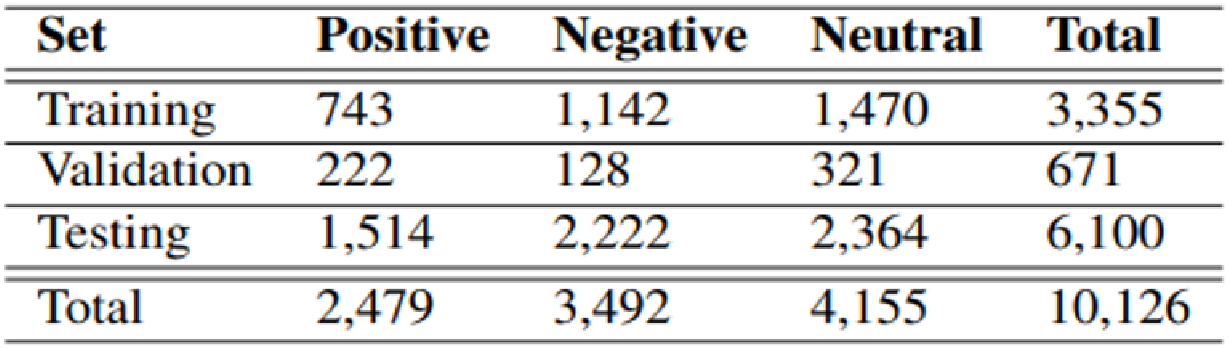
The reason for this choice is that sarcasm is highly subjective and always mentioned as one of the main reasons that degrades sentiment analysers' performance. ASTD dataset consists of 10,006 tweets labelled as shown in Table 1. The dataset contains tweets that date back to the period between 2013 and 2015. The tweets are mostly in Egyptian dialect and they were annotated using Amazon's Mechanical Turk. In our work, since we are aiming to annotate for sarcasm, we decided to eliminate the objective class and we took our sample from the other subjective classes.



Class	Count
Positive	799
Negative	1,684
Neutral	832
Objective	6,691
Total	10,006

Figure 3.1: ASTD statistics

The other dataset we are using is the one provided in SemEval's 2017 task for Arabic SA. This dataset consists of 10,126 tweets distributed over different sets as shown in Table 2. The data was annotated using CrowdFlower4 crowd-sourcing platform. The new dataset contains 10,543 tweets, most of which were taken from SemEval's dataset.



Set	Positive	Negative	Neutral	Total
Training	743	1,142	1,470	3,355
Validation	222	128	321	671
Testing	1,514	2,222	2,364	6,100
Total	2,479	3,492	4,155	10,126

Figure 3.2: SemEval 2017 Task 4-A dataset statistics

3.1.2 Dataset Statistics

The new dataset contains 10,547 tweets, 8,075 of them were taken from SemEval's dataset while the rest (2,472 tweets) were taken from ASTD. Each of the tweets has three labels for sarcasm, sentiment and dialect. Table 3 shows the statistics of the new dataset, where we can see that 16% of the data is sarcastic (1,682 tweets).
And for the data after being labeled:

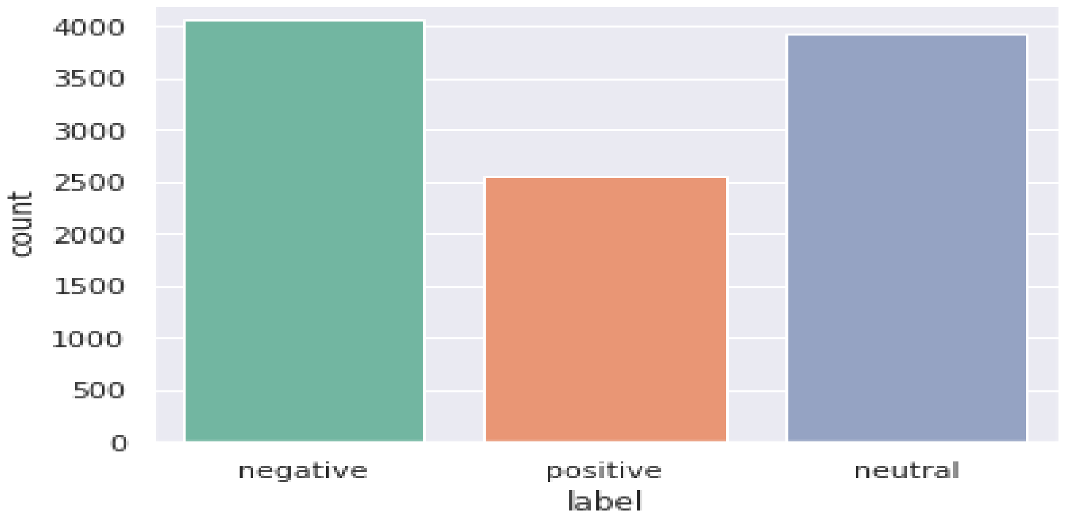


Figure 3.3: 40000-Egyptian-tweets & ArSarcasm distribution

The new annotation shows that most of the data is either in MSA or the Egyptian dialect, while there are few examples of the Maghrebi dialect. Figure 1 shows the ratio of sarcasm in the tweets belonging to each dialect. Maghrebi dialect has the largest percentage, but this is an outlier due to the small number of Maghrebi tweets (only 32 tweets). Thus, sarcasm is more prominent in the Egyptian dialect with 34% of the Egyptian tweets being sarcastic.

Also, from the table, it is noticeable that the Egyptian dialect comprises most of the sarcastic tweets and for this task we split the data to 0.3 test and validation as we will explain it later.

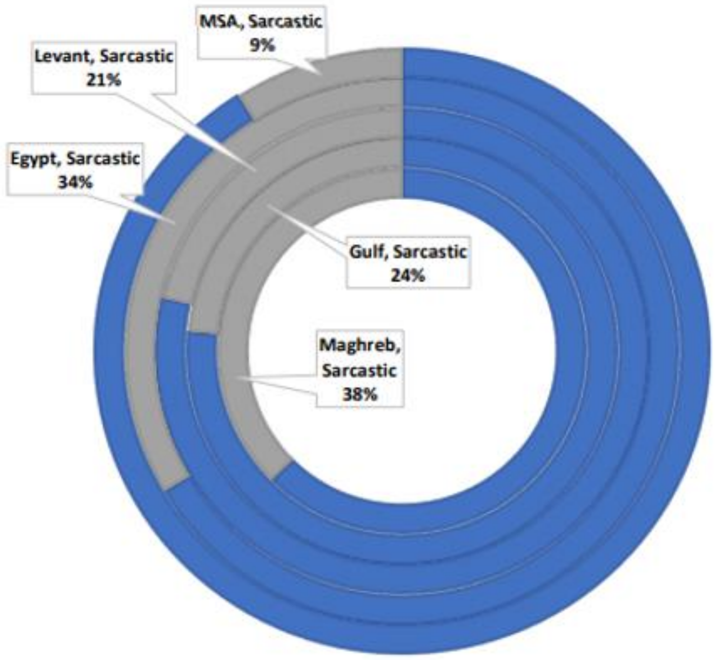


Figure 3.4: Ratio of sarcasm over the dialects

3.2 PRE-PROCESSING

Currently, social media plays an important role in daily life and routine. Millions of people use social media for different purposes. Large amounts of data flow through online networks every second, and these data contain valuable information that can be extracted if the data are properly processed and analyzed. However, most of the processing results are affected by preprocessing difficulties. The preprocessed Arabic text is stored in structured database tables to provide a useful corpus to which, information extraction and data analysis algorithms can be applied.

Challenges in processing Arabic Text:

The complexities and characteristics of the Arabic language have made processing Arabic text, a major challenge, wherein researchers must deal with several difficulties, such as ambiguity, diglossia, and the difficulties in reading and understanding the Arabic script:

1. the Arabic letter changes according to its position in the word,
2. no capitalization or dedicated letter,
3. the complex structure of the word and morphology.
4. Another challenge is the normalization of inconsistency in the use of certain letters, dialect words, or diacritical marks.

Social Media Arabic Text:

Social media Arabic text has special characteristics compared to those of Classical Arabic (CA), Modern standard Arabic (MSA), and Dialect Arabic. Such text can be a mixture of all these varieties, and additionally, it may contain non-Arabic words, samples, notations, and orthographic features such as spelling mistakes, repeated letters, or express emotive words. Therefore, in addition to tackling the complexity challenges of the Arabic language, other issues such as cleaning noise from the text, changing a word to its normal form, dealing with the dialect language must be handled. Consequently, preprocessing social media Arabic text poses a great challenge to the research and application developers.

Cleaning:

we aim to clean the social media Arabic text using NLP tools at the same time while preserving meaning so we did the following:

- Replace # with word “هاشتاج”.
- Replace urls with word “رابط”.
- Replace @ mentions with word “مستخدم”.
- Removing the digits and non-Arabic letters.
- Removing elongation.
- Keeping the stopwords for contextual meaning.
- Removing redundant punctuations.
- Removing special characters.
- Keeping Emojis.

3.3 PROPOSED METHODOLOGY

3.3.1 Word Embedding:

There are some definitions for what Word Embeddings are, but in the most general notion, word embeddings are the numerical representation of words, usually in a shape of a vector. Being more specific, word embeddings are unsupervisedly learned word representation vectors whose relative similarities correlate with semantic similarity. In computational linguistics they are often referred as distributional semantic model or distributed representations. The theoretical foundations of word embeddings can be traced back to the early 1950's and in particular in the works of Zellig Harris, John Firth, and Ludwig Wittgenstein.

Naturally, every feed-forward neural network that takes words from a vocabulary as input and embeds them as vectors into a lower dimensional space, which it then fine-tunes through back-propagation, necessarily yields word embeddings as the weights of the first layer, which is usually referred to as Embedding Layer.

The earliest attempts at using feature representations to quantify (semantic) similarity used handcrafted features. A good example is the work on semantic differentials [Osgood, 1964]. The early 1990's saw the rise of automatically generated contextual features, and the rise of Deep Learning methods for Natural Language Processing (NLP) in the early 2010's helped to increase their popularity, to the point that, these days, word embeddings are the most popular research area in NLP.

Types of Word Embedding:

- Classical Neural Language Model
- Word2Vec (Google 2013)
- Global Vectors (GloVe) (Stanford, 2014)

1. Classical Neural Language Model

The classic neural language model proposed by (Bengio et al., 2003) consists of a one-hidden layer feed-forward neural network that predicts the next word in a sequence.

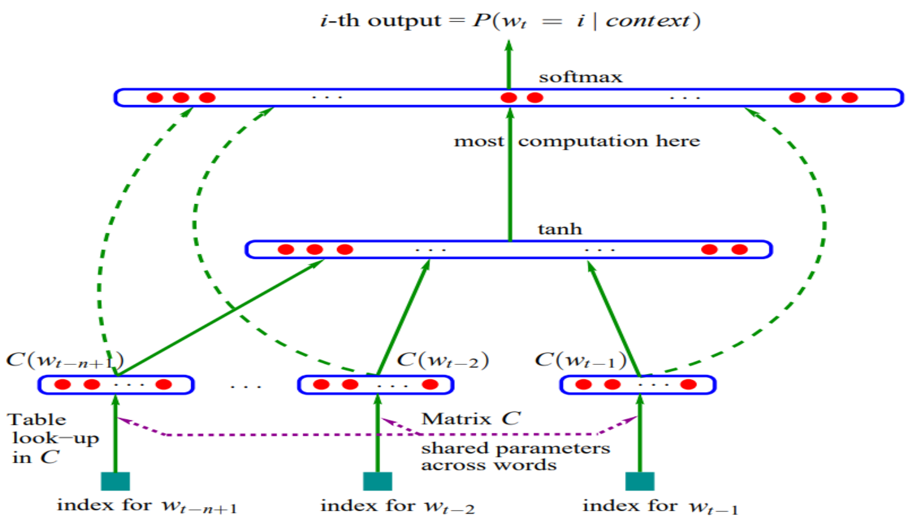


Figure 5.1: Classic neural language model (Bengio et al., 2003)

Bengio et al. are one of the first to introduce what we now refer to as a word embedding, a real-valued word feature vector in RR. Their architecture forms very much the prototype upon which current approaches have gradually improved. The general building blocks of their model, however, are still found in all current neural language and word embedding models…These are:

- **Embedding Layer**: a layer that generates word embeddings by multiplying an index vector with a word embedding matrix.

- **Intermediate Layer(s)**: one or more layers that produce an intermediate representation of the input, e.g. a fully-connected layer that applies a non-linearity to the concatenation of word embeddings of n previous words.

- **Softmax Layer**: the final layer that produces a probability distribution.

2. Word2Vec (Google 2013)

Word2vec is a particularly computationally efficient predictive model for learning word embeddings from raw text. It comes in two flavors, the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model. Algorithmically, these models are similar, except that CBOW model learns to predict a center word given context words, while the skip-gram does the inverse and predicts source context-words from the target words.

- Continuous Bag-of-Words Model

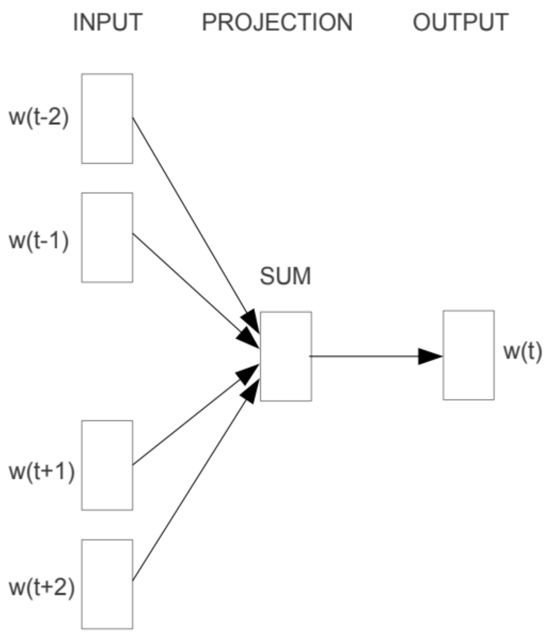


Figure 5.2: CBOW model architecture

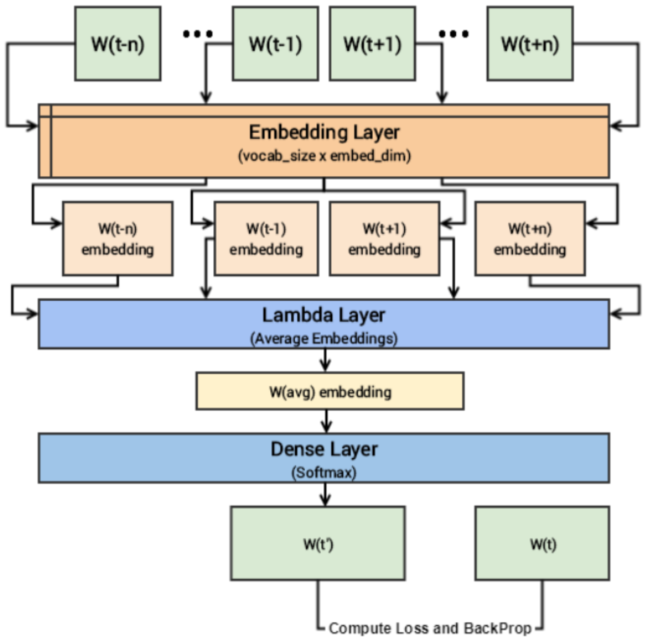


Figure 5.3: CBOW deep learning model

3. Global Vectors (GloVe) (Stanford, 2014)

Both CBOW and Skip-Grams are “predictive” models, in that they only take local contexts into account. Word2Vec does not take advantage of global context. GloVe embeddings by contrast leverage the same intuition behind the co-occuring matrix used distributional embeddings, but uses neural methods to decompose the co-occurrence matrix into more expressive and dense word vectors. While GloVe vectors are faster to train, neither GloVe or Word2Vec has been shown to provide definitively better results rather they should both be evaluated for a given dataset.

Word Embeddings and Their Challenges:

- Inability to handle unknown or OOV (out-of-vocabulary) words
- No shared representations at sub-word levels
- Scaling to new languages requires new embedding matrices

Since those drawbacks can be an obstacle in the understanding of the semantic, Bidirectional Encoder Representations from Transformer (BERT) was represented as the optimal selection for the task for its ability to outdraw those challenges.

3.3.2 Loss function

Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally.

For example, you may have a 2-class (binary) classification problem with 100 instances (rows). A total of 80 instances are labeled with Class-1 and the remaining 20 instances are labeled with Class-2.

This is an imbalanced dataset and the ratio of Class-1 to Class-2 instances is 80:20 or more concisely 4:1.

You can have a class imbalance problem on two-class classification problems as well as multi-class classification problems. Most techniques can be used on either.

The remaining discussions will assume a two-class classification problem because it is easier to think about and describe.

Most classification data sets do not have exactly equal number of instances in each class, but a small difference often does not matter.

There are problems where a class imbalance is not just common, it is expected. For example, datasets like those that characterize fraudulent transactions are imbalanced. The vast majority of the transactions will be in the “Not-Fraud” class and a very small minority will be in the “Fraud” class.

Another example is customer churn datasets, where the vast majority of customers stay with the service (the “No-Churn” class) and a small minority cancel their subscription (the “Churn” class).

When there is a modest class imbalance like 4:1 in the example above it can cause problems.

A major setback of the data set is its imbalance, where class (x) represented the majority of the data set. To remedy this, we implemented a custom loss function, Focal Loss.

3.3.2.1 Focal Loss:

A function that addresses class imbalance during training in tasks like object detection. Focal loss applies a modulating term to the cross entropy loss in order to focus learning on hard negative examples. It is a dynamically scaled cross entropy loss, where the scaling factor decays to zero as confidence in the correct class increases. Intuitively, this scaling factor can automatically down-weight the contribution of easy examples during training and rapidly focus the model on hard examples.

Formally, the Focal Loss adds a factor $1 - p(t)^{\gamma}$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples$(p_t > .5)$, putting more focus on hard, misclassified examples. Here there is tunable *focusing* parameter $\gamma \geq 0$.

$$FL(p_t) = -(1 - p_t)^{\gamma} log(p_t)$$

3.3.2 Hyperparameter Sweeps:

Sweep provided by W&B, an open source library used for Build better models more efficiently with Weights & Biases experiment tracking.

Hyperparameters play a crucial role in determining a machine learning model’s performance, hyperparameter Sweeps offer efficient ways of automatically finding the best possible combination of hyperparameter values (number of epochs, batch size, number of layers, number of nodes in each layer, etc.) for your machine learning model with respect to a particular dataset.

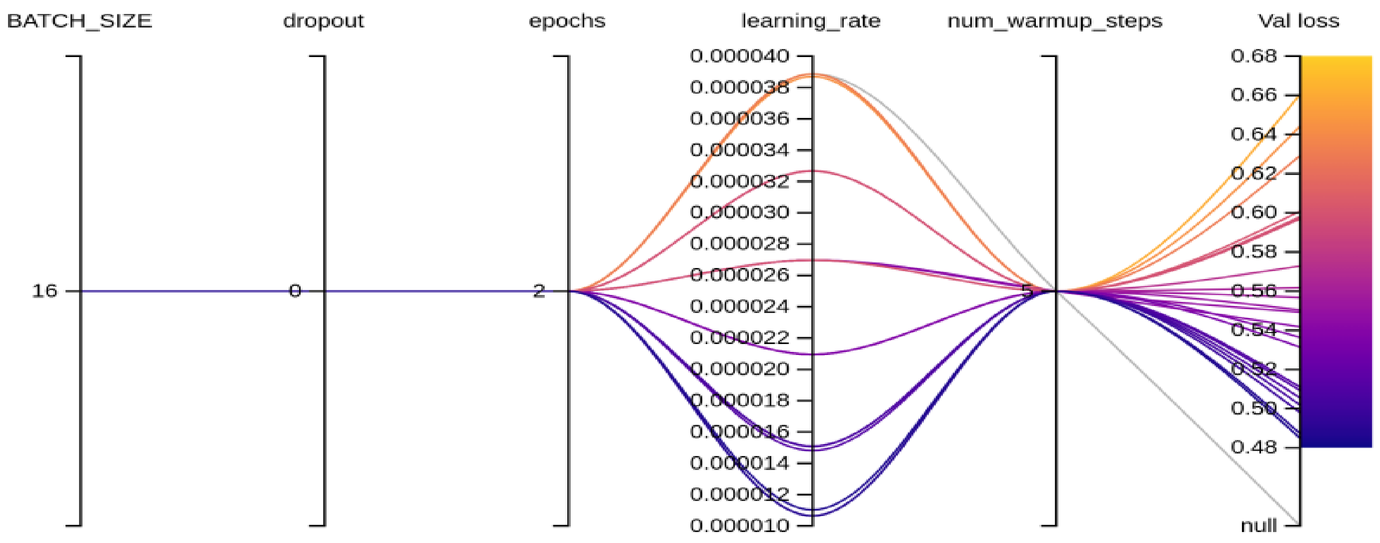


Figure 5.4: The optimal Hyperparameters

3.3.3 Label Smoothing

Label Smoothing is a regularization technique that introduces noise for the labels. This accounts for the fact that datasets may have mistaken in them, so maximizing the likelihood of $log\ p(y \mid x)$ directly can be harmful. Assume for a small constant ϵ the training set label y is correct with probability $1 - \epsilon$ and incorrect otherwise. Label Smoothing regularizes a model based on a softmax with k output values by replacing the
hard 0 and 1 classification targets with targets of $\epsilon\ /k - 1$ and $1 - \epsilon$

Label Smoothing was used here to overcome the both overfitting and overconfidence problems due to the little amount of data and class imbalance.

3.3.4 Models

The introduction of Bidirectional Encoder Representation from Transformers (BERT) (Devlin et al., 2019) led to a revolution in the NLP world. Since then, many other models have been released such as ELECTRA (Clark et al., 2020), GPT-1/2/3 (Radford et al., 2019; Brown et al., 2020) and RoBERTa (Liu et al., 2019). Those models helped achieve state-of-the-art results on different tasks such as sentiment analysis, named entity recognition (NER), sentence completion and others. However, those models were trained mostly on English data while others included data from other languages such as the multilingual BERT (Devlin et al., 2019).

A number of Arabic language models (LMs) has been developed, and we have used three of these models to extract the semantic from the Arabic tweets.

ARBERT Model:

AraBERT (Antoun et al., 2020), which is trained with the same architecture as BERT (Devlin et al., 2019) and uses the BERTBase configuration. AraBERT is trained on 23GB of Arabic text, making ~ 70M sentences and 3B words, from Arabic Wikipedia, the Open Source International dataset (OSIAN) (Zeroual et al., 2019) (3.5M news articles from 24 Arab countries), and 1.5B words Corpus from El-Khair (2016) (5M articles extracted from 10 news sources). (Antoun et al. 2020) evaluate AraBERT on three Arabic downstream tasks. These are

- sentiment analysis from six different datasets: HARD (Elnagar et al., 2018), ASTD (Nabil et al., 2015), ArsenTDLev (Baly et al., 2019), LABR (Aly and Atiya, 2013), and ArSaS (Elmadany et al., 2018).
- NER, with the ANERcorp (Benajiba and Rosso, 2007).
- Arabic QA, on Arabic SQuAD and ARCD (Mozannar et al., 2019) datasets. Another Arabic LM that was also introduced in ArabicBERT (Safaya et al., 2020), which is similarly based on BERT architecture. ArabicBERT was pre trained on two datasets only, Arabic Wikipedia and Arabic OSCAR (Suarez et al., 2019).

Since both of these datasets are already included in AraBERT, and Arabic OSACAR1 has significant duplicates, they compare to AraBERT only. GigaBERT (Lan et al., 2020), an Arabic and English LM designed with code-switching data in mind, was also introduced.

ARBERT was trained on 61GB of MSA text (6.5B tokens) from the following sources:

- Books (Hindawi). They collect and preprocess 1,800 Arabic books from the public Arabic bookstore Hindawi.

- El-Khair. This is a 5M news articles dataset from 10 major news sources covering eight Arab countries from El-Khair (2016).

- Gigaword. They use Arabic Gigaword 5th Edition from the Linguistic Data Consortium (LDC). The dataset is a comprehensive archive of newswire text from multiple Arabic news sources.

- OSCAR. This is the MSA and Egyptian Arabic portion of the Open Super-Large Crawled Almanac corpus (Suarez et al., 2019), a huge multilingual subset from Common Crawl6 obtained using language identification and filtering.

- OSIAN. The Open Source International Arabic News Corpus (OSIAN) (Zeroual et al., 2019) consists of 3.5 million articles from 31 news sources in 24 Arab countries.

- Wikipedia Arabic. They download and use the December 2019 dump of Arabic Wikipedia. They use WikiExtractor7 to extract articles and remove markup from the dump.

ARBERT was pre trained as follows, they follow (Devlin et al. (2019)'s pre-training setup. To generate each training input sequence, they use the whole word masking, where 15% of the N input tokens are selected for replacement. These tokens are replaced 80% of the time with the [MASK] token, 10% with a random token, and 10% with the original token. They use the original implementation of BERT in the TensorFlow framework.9 As mentioned, they use the same network architecture as BERTBase:

- 12 layers
- 768 hidden units
- 12 heads

for a total of ∼ 163M parameters. They use a batch size of 256 sequences and a maximum sequence length of 128 tokens (256 sequences × 128 tokens = 32, 768 tokens/batch) for 8M steps, which is approximately 42 epochs over the 6.5B tokens. For all their models, they use a learning rate of 1e−4.

XLM-T Model:

XLMT, is a framework for using and evaluating multilingual language models in Twitter. This framework features two main assets: (1) a strong multilingual baseline consisting of an XML-R (Conneau et al., 2020) model pre-trained on millions of tweets in over thirty languages, alongside starter code to subsequently fine-tune on a target task; and (2) a set of unified sentiment analysis Twitter datasets in eight different languages. This is a modular framework that can easily be extended to additional tasks, as well as integrated with recent efforts also aimed at the homogenization of Twitter-specific datasets (Barbieri et al., 2020)

XLM-T was pre-trained as follows, they used the Twitter API to retrieve 198M tweets posted between May'18 and March'20, which are their source data for LM pre-training. They only considered tweets with at least three tokens and with no URLs to avoid bot tweets and spam advertising. Additionally, they did not perform language filtering, aiming at capturing a general distribution.

In terms of opting for pre-training a LM from scratch or building upon an existing one, they follow (Gururangan et al. 2020) and (Barbieri et al. 2020) and continue training an XML-R language model from publicly available checkpoints, which they selected due to the high results it has achieved in several multilingual NLP tasks (Hu et al., 2020). They use the same masked LM objective, and train until convergence in a validation set.

MARBER Model:

The most notable among these is MARBERT, which gave the best accuracy of 88%.

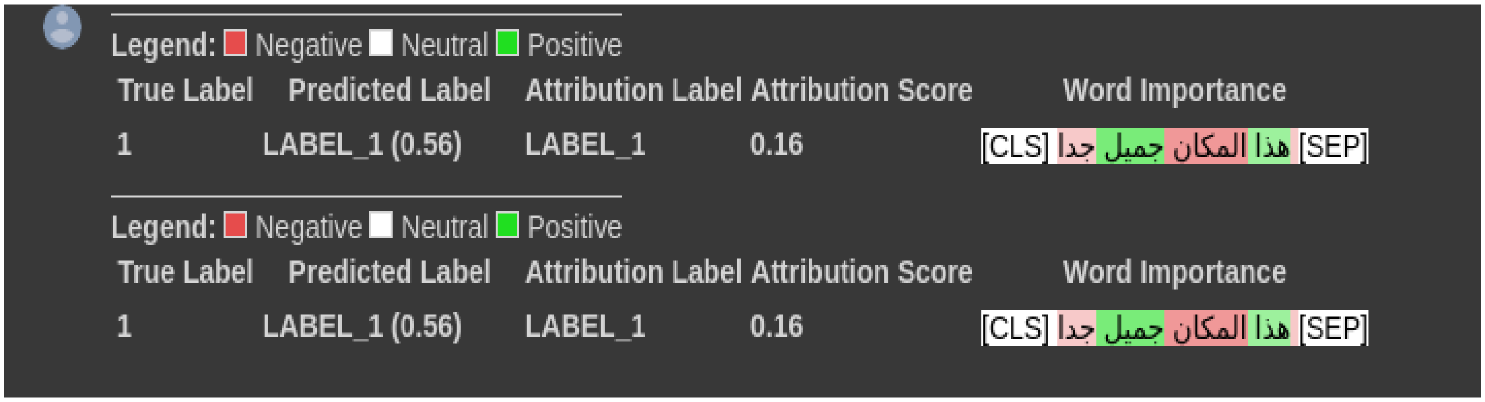
As known, Arabic has a large number of diverse dialects. Most of these dialects are under-studied due to the rarity of resources. Multilingual models such as mBERT and XML-R are trained on mostly MSA data, which is also the case for AraBERT and ARBERT. As such, these models are not best suited for downstream tasks involving dialectal Arabic. To treat this issue, they use a large Twitter dataset to pre-train a new model.

To pre-train MARBERT, they randomly sample 1B Arabic tweets from a large in-house dataset of about 6B tweets. They only include tweets with at least three Arabic words, based on character string matching, regardless whether the tweet has non-Arabic string or not. That is, they do not remove non-Arabic so long as the tweet meets the three Arabic word criteria. The dataset makes up 128GB of text (15.6B tokens).

Pre-training. They use the same network architecture as BERTBase, but without the next sentence prediction (NSP) objective since tweets are short. They use the same vocabulary size (100K wordPieces) as ARBERT, and MARBERT also has ~ 160M parameters. They train MARBERT for 17M steps (~ 36 epochs) with a batch size of 256 and a maximum sequence length of 128.

3.3.5 Captum Interepertability

Captum is a model interpretability and understanding library for PyTorch. Captum means comprehension in Latin and contains general purpose implementations of integrated gradients, saliency maps, smoothgrad, vargrad and others for PyTorch models. It has quick integration for models built with domain-specific libraries such as torchvision, torchtext, and others, it can be used to improve and troubleshoot models by facilitating the identification of different features that contribute to a model’s output in order to design better models and troubleshoot unexpected model outputs.



Legend: 🟥 Negative ⬜ Neutral 🟩 Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
1	LABEL_1 (0.56)	LABEL_1	0.16	[CLS] هذا المكان جميل جدا [SEP]

Legend: 🟥 Negative ⬜ Neutral 🟩 Positive

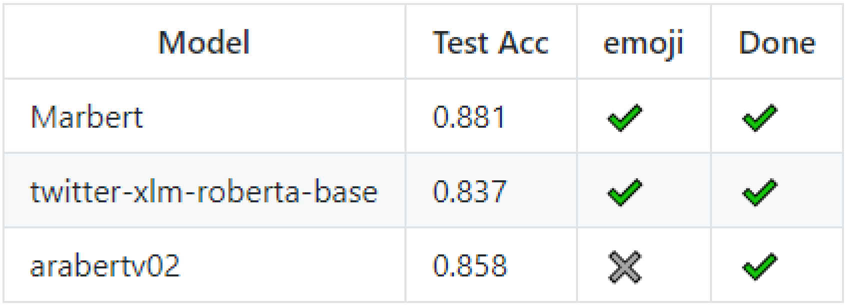
True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
1	LABEL_1 (0.56)	LABEL_1	0.16	[CLS] هذا المكان جميل جدا [SEP]

Figure 5.5: Word importance in classification

CHAPTER 4: EXPERIMENTS AND RESULTS

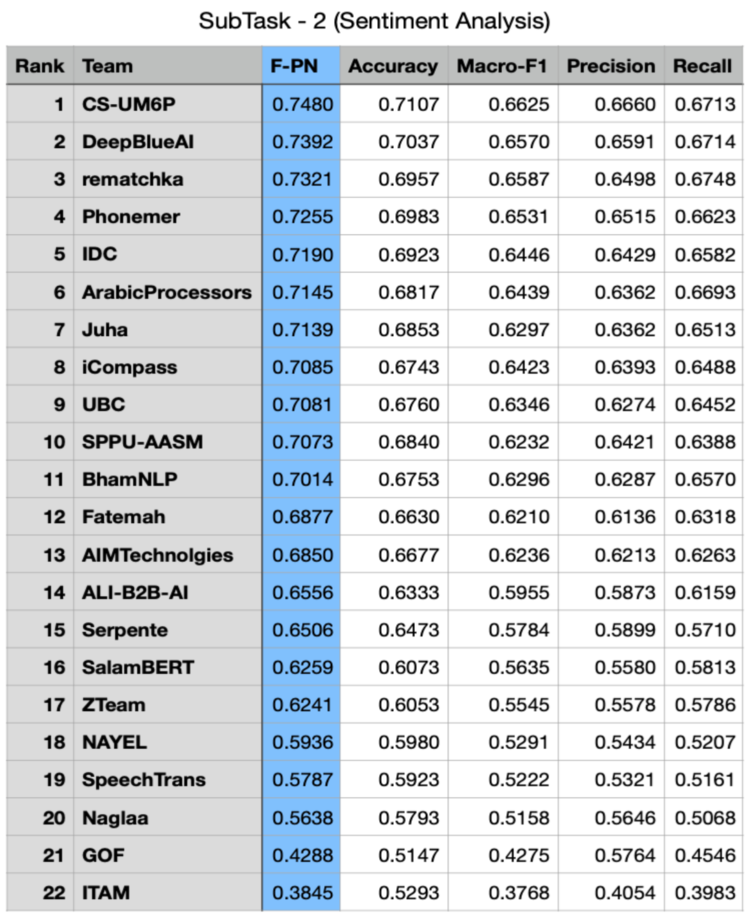
As we explained how the three models work let's see how the results are:

- For the two dataset together we have a high accuracy results from the three models shown in this table. We have marbert model as the highest as to train ARBERT, we use the same architecture as BERT-base: 12 attention layers, each has 12 attention heads and 768 hidden dimensions, a vocabulary of 100K WordPieces, making ~163M parameters and To train MARBERT, we randomly sample 1B Arabic tweets from a large in-house dataset of about 6B tweets. We only include tweets with at least *3 Arabic words*, based on character string matching, regardless whether the tweet has non-Arabic string or not. That is, we do not remove non-Arabic so long as the tweet meets the 3 Arabic word criterions. The dataset makes up 128GB of text (15.6B tokens). We use the same network architecture as ARBERT (BERT-base), but without the next sentence prediction (NSP) objective since tweets are short.



Model	Test Acc	emoji	Done
Marbert	0.881	✓	✓
twitter-xlm-roberta-base	0.837	✓	✓
arabertv02	0.858	✗	✓

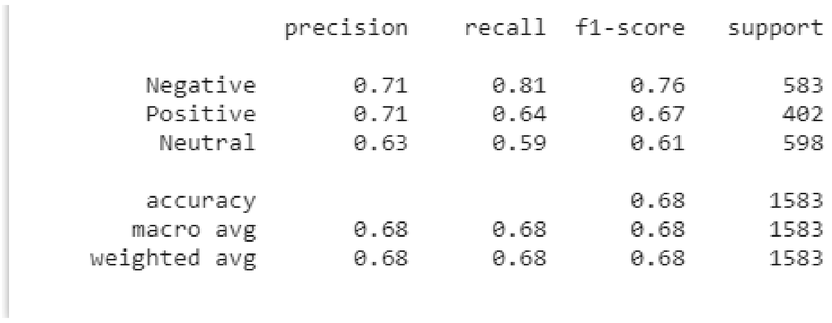
- As for the ArSarcam dataset that used in the competition (SarcasmAndSentimentDetectionInArabic) ,we manage to get all the accuracy of this competition and the highest accuracy was 0.71 as shown in the table.



SubTask - 2 (Sentiment Analysis)

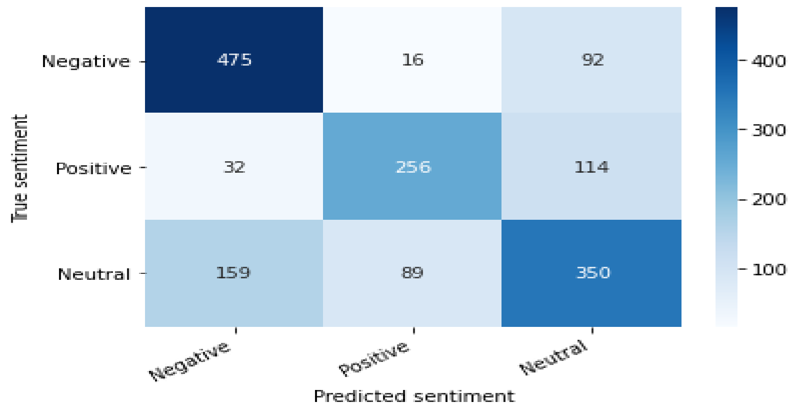
Rank	Team	F-PN	Accuracy	Macro-F1	Precision	Recall
1	CS-UM6P	0.7480	0.7107	0.6625	0.6660	0.6713
2	DeepBlueAI	0.7392	0.7037	0.6570	0.6591	0.6714
3	rematchka	0.7321	0.6957	0.6587	0.6498	0.6748
4	Phonemer	0.7255	0.6983	0.6531	0.6515	0.6623
5	IDC	0.7190	0.6923	0.6446	0.6429	0.6582
6	ArabicProcessors	0.7145	0.6817	0.6439	0.6362	0.6693
7	Juha	0.7139	0.6853	0.6297	0.6362	0.6513
8	iCompass	0.7085	0.6743	0.6423	0.6393	0.6488
9	UBC	0.7081	0.6760	0.6346	0.6274	0.6452
10	SPPU-AASM	0.7073	0.6840	0.6232	0.6421	0.6388
11	BhamNLP	0.7014	0.6753	0.6296	0.6287	0.6570
12	Fatemah	0.6877	0.6630	0.6210	0.6136	0.6318
13	AIMTechnolgies	0.6850	0.6677	0.6236	0.6213	0.6263
14	ALI-B2B-AI	0.6556	0.6333	0.5955	0.5873	0.6159
15	Serpente	0.6506	0.6473	0.5784	0.5899	0.5710
16	SalamBERT	0.6259	0.6073	0.5635	0.5580	0.5813
17	ZTeam	0.6241	0.6053	0.5545	0.5578	0.5786
18	NAYEL	0.5936	0.5980	0.5291	0.5434	0.5207
19	SpeechTrans	0.5787	0.5923	0.5222	0.5321	0.5161
20	Naglaa	0.5638	0.5793	0.5158	0.5646	0.5068
21	GOF	0.4288	0.5147	0.4275	0.5764	0.4546
22	ITAM	0.3845	0.5293	0.3768	0.4054	0.3983

- So if we can see that the ***twitter-xlm-roberta-base*** is the lowest accuracy even so it’s continue pre-training that has a large corpus of Twitter in multiple languages it need more data especially when handling Arabic text and emojis.

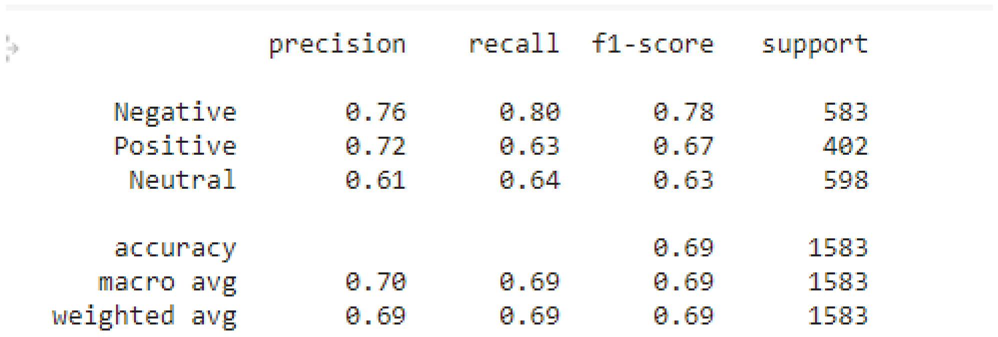


```
              precision    recall  f1-score   support

    Negative       0.71      0.81      0.76       583
    Positive       0.71      0.64      0.67       402
     Neutral       0.63      0.59      0.61       598

    accuracy                           0.68      1583
   macro avg       0.68      0.68      0.68      1583
weighted avg       0.68      0.68      0.68      1583
```

- And for its confusion matrix that shows between the true sentiment and the predicted sentiment:

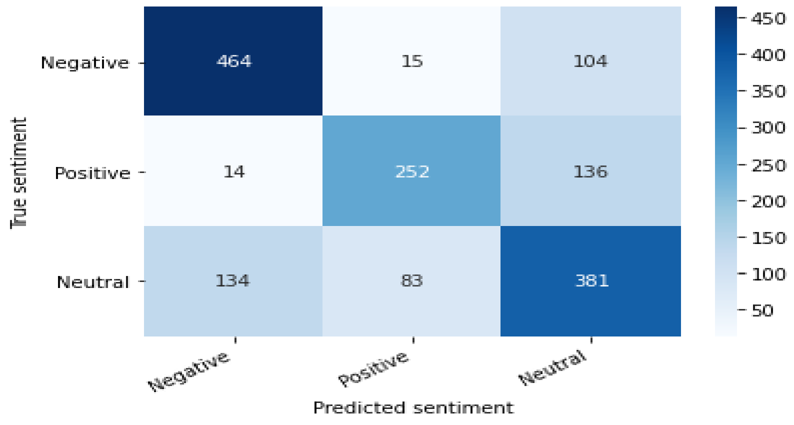


- And for the next model that has better accuracy that XLM is ***Arabert*** with 0.69 accuracy and we explained that it uses the same architecture as BERT-base.

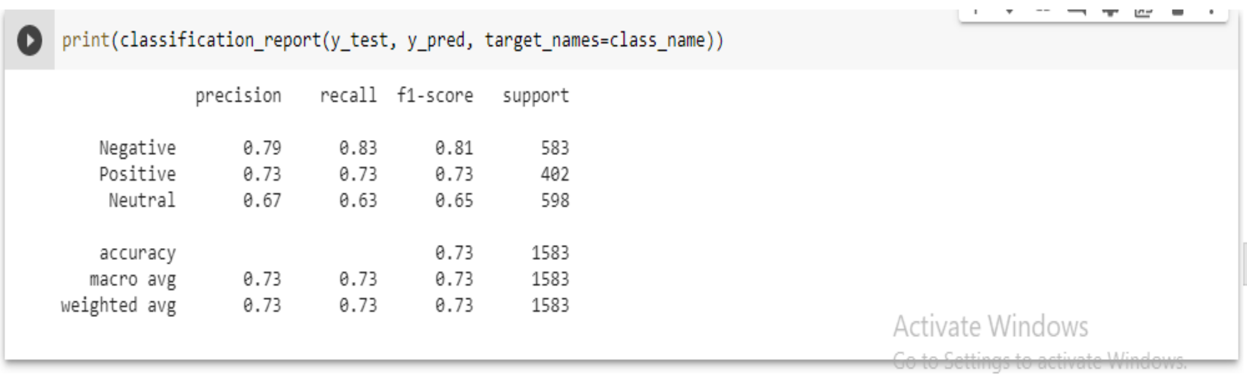


```
                 precision    recall  f1-score   support

    Negative          0.76      0.80      0.78       583
    Positive          0.72      0.63      0.67       402
     Neutral          0.61      0.64      0.63       598

    accuracy                              0.69      1583
   macro avg          0.70      0.69      0.69      1583
weighted avg          0.69      0.69      0.69      1583
```

- And for its confusion matrix that show how the result was better than the previous model.



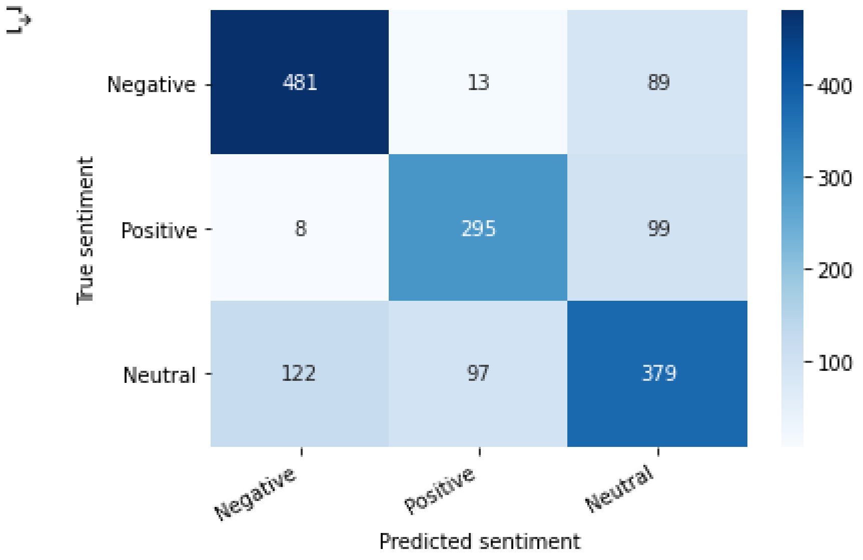
- For the last one, **Marbert** shows the best accuracy for this kind of data and that why it is not using the Bert-base architecture. This model include tweets with at least *3 Arabic words*, based on character string matching.



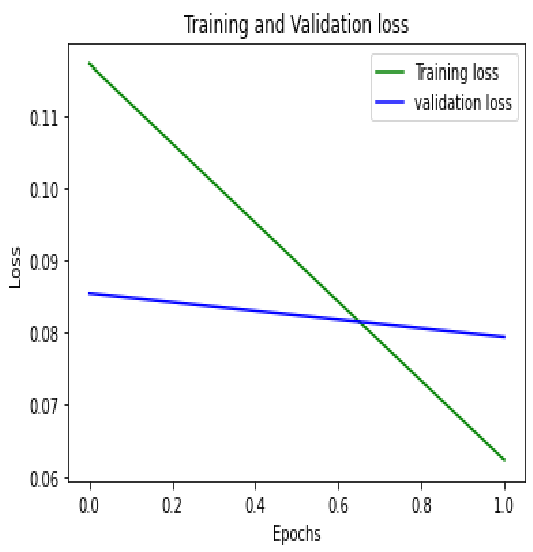
```
print(classification_report(y_test, y_pred, target_names=class_name))
```

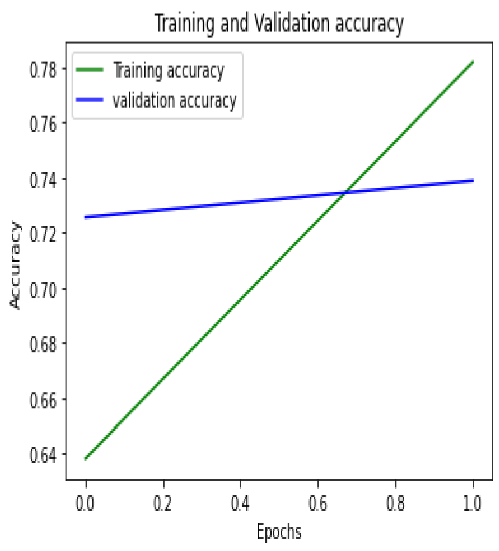
```
              precision    recall  f1-score   support

    Negative       0.79      0.83      0.81       583
    Positive       0.73      0.73      0.73       402
     Neutral       0.67      0.63      0.65       598

    accuracy                           0.73      1583
   macro avg       0.73      0.73      0.73      1583
weighted avg       0.73      0.73      0.73      1583
```

- And for its confusion matrix, we see that it's a lot better than other models even better result than the competition.

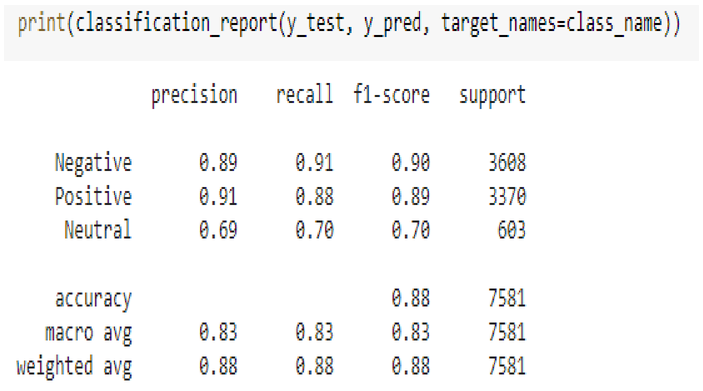


- And for the training loss vs. validation loss over the number of epochs and the graph of training accuracy vs. validation accuracy over the number of epochs the two graphs shows how they changed over (0,2) Epochs.

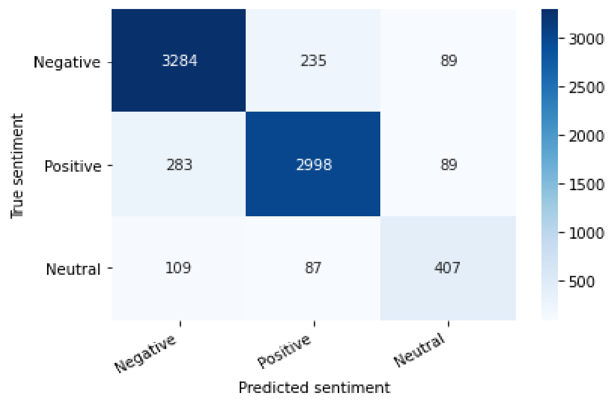




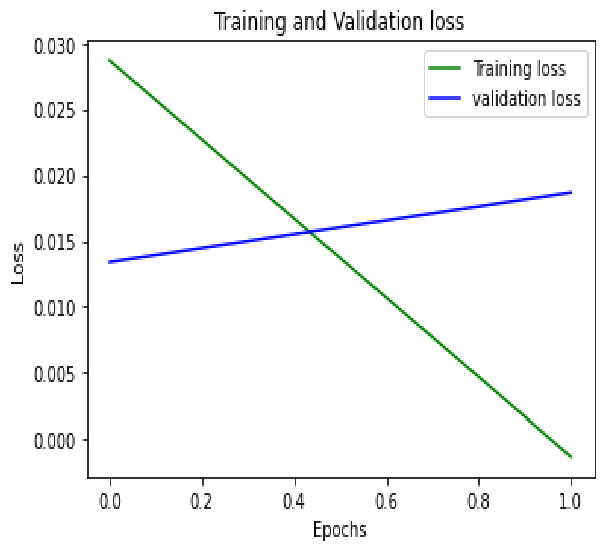
- The result shows us how the model is better than the competition so we used it with a bigger dataset (40000-Egyptian-tweets) and the accuracy was satisfying and higher so as its confusion matrix.

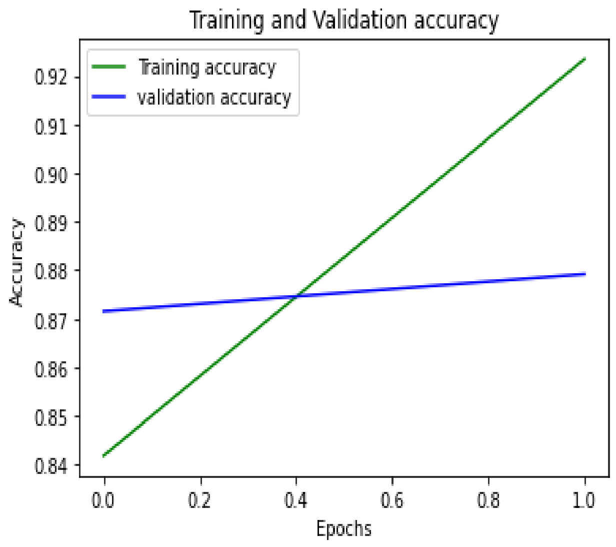


```
print(classification_report(y_test, y_pred, target_names=class_name))

              precision    recall  f1-score   support

    Negative       0.89      0.91      0.90      3608
    Positive       0.91      0.88      0.89      3370
     Neutral       0.69      0.70      0.70       603

    accuracy                           0.88      7581
   macro avg       0.83      0.83      0.83      7581
weighted avg       0.88      0.88      0.88      7581
```



- And for the loss and accuracy, the loss is has substantially declined as we increase the amount of trained data.





CHAPTER 5: CONCLUSION and FUTURE WORK

Conducting research about best strategies for classification processing steps to identify optimal strategies for emoji sentiment analysis, we found that using ‘Bidirectional Encoder Representations from Transformers (BERT)’ was the best approach Where we make use of transformers that use an attention mechanism to extract the finest semantics from the Arabic tweets.

Regarding the competition, after applying the pre-processing and feeding the MarBERT model with the *ArSarcasm dataset,* which has a collection of arabic tweets containing a variety of emojis, we found that the model has produced better results than its counterparts and both training and validation loss started to adapt to the changes of the pre-training, as explained in the diagrams of the results section including MarBert, Arabert, and XML-RB. MarBert's results were the most refined of all, especially when it came to the loss domain.

The Model	**Accuracy**
With using the competition data	
MARBERT	0.73
XML-Rb	0.68
AraBert	0.69
+ using 40000-Egyptian-tweets	
MARBERT	0.88
XML-Rb	0.84
AraBert	0.86

As for the future work:

- We may use the model in applications like social media monitoring, customer support management, and analyzing customer feedback.
- Increase the number of encoders, decoders, and Number of attention heads.
- According to the confusion matrix results, we can increase the diversity of training examples in the Dataset to improve the accuracy.

REFERENCES

[1] Heikal, Maha, Marwan Torki, and Nagwa El-Makky. "Sentiment analysis of Arabic Tweets using deep learning." *Procedia Computer Science* 142 (2018): 114-122.

[2] Al-Twairesh, Nora, et al. "Arasenti-tweet: A corpus for arabic sentiment analysis of saudi tweets." *Procedia Computer Science* 117 (2017): 63-72.

[3] Wissam Antoun, Fady Baly, and Hazem Hajj. "Arabert: Transformer-based model for arabic language understanding." *arXiv preprint arXiv:2003.00104* (2020).

[4] Al-Thubaity, Abdulmohsen, Qubayl Alqahtani, and Abdulaziz Aljandal. "Sentiment lexicon for sentiment analysis of Saudi dialect tweets." *Procedia computer science* 142 (2018): 301-307.

[5] El-Masri, Mazen, et al. "A web-based tool for Arabic sentiment analysis." *Procedia Computer Science* 117 (2017): 38-45.

[6] Nerabie, Abdul Munem, et al. "The Impact of Arabic Part of Speech Tagging on Sentiment Analysis: A New Corpus and Deep Learning Approach." *Procedia Computer Science* 184 (2021): 148-155.

[7] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

[8] Abdul-Mageed, Muhammad, AbdelRahim Elmadany, and El Moatez Billah Nagoudi. "ARBERT & MARBERT: deep bidirectional transformers for Arabic." *arXiv preprint arXiv:2101.01785* (2020).

[9] Alhumoud Sarah. "Arabic Sentiment Analysis using Deep Learning for COVID-19 Twitter Data." *International Journal of Computer Science and Network Security* (2020): 132-138.

[10] Al Sallab, Ahmad, et al. "Deep learning models for sentiment analysis in Arabic." *Proceedings of the second workshop on Arabic natural language processing*. 2015.

1. Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pretraining text encoders as discriminators rather than generators. In ICLR.

2. Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

3. Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

4. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

5. Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. Arabert: Transformer-based model for arabic language understanding. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, pages 9–15.

6. Imad Zeroual, Dirk Goldhahn, Thomas Eckart, and Abdelhak Lakhouaja. 2019. OSIAN: Open Source International Arabic News Corpus - Preparation and Integration into the CLARIN-infrastructure. In Proceedings of the Fourth Arabic Natural Language Processing Workshop, pages 175–182, Florence, Italy. Association for Computational Linguistics.

7. Ashraf Elnagar, Yasmin S Khalifa, and Anas Einea. 2018. Hotel Arabic-Reviews Dataset Construction for Sentiment Analysis Applications. In Intelligent Natural Language Processing: Trends and Applications, pages 35–52. Springer.
8. Mahmoud Nabil, Mohamed Aly, and Amir F Atiya. 2015. Astd: Arabic sentiment tweets dataset. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2515–2519.

9. Ramy Baly, Alaa Khaddaj, Hazem Hajj, Wassim ElHajj, and Khaled Bashir Shaban. 2019. ArSentDLEV: A multi-topic corpus for target-based sentiment analysis in Arabic levantine tweets. arXiv preprint arXiv:1906.01830.

10. Mohamed Aly and Amir Atiya. 2013. LABR: A Large Scale Arabic book Reviews Dataset. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), volume 2, pages 494–498.

11. AbdelRahim Elmadany, Hamdy Mubarak, and Walid Magdy. 2018. ArSAS: An Arabic Speech-Act and Sentiment Corpus of Tweets. OSACT, 3:20.

12. Yassine Benajiba and Paolo Rosso. 2007. ANERsys 2.0: Conquering the NER Task for the Arabic Language by Combining the Maximum Entropy with POS-tag Information. In IICAI, pages 1814–1823.

13. Hussein Mozannar, Karl El Hajal, Elie Maamary, and Hazem Hajj. 2019. Neural Arabic Question Answering. In Proceedings of the Fourth Arabic Natural Language Processing Workshop, Florence, Italy. Association for Computational Linguistics.

14. Ali Safaya, Moutasem Abdullatif, and Deniz Yuret. 2020. KUISAIL at SemEval-2020 task 12: BERTCNN for offensive speech identification in social media. In Proceedings of the Fourteenth Workshop on Semantic Evaluation, pages 2054–2059, Barcelona (online). International Committee for Computational Linguistics.

15. Pedro Javier Ortiz Suarez, Benoıt Sagot, and Laurent Romary. 2019. Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructure. In 7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7). Leibniz-Institut fur Deutsche Sprache.

16. Wuwei Lan, Yang Chen, Wei Xu, and Alan Ritter. 2020. An Empirical Study of Pre-trained Transformers for Arabic Information Extraction. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4727– 4734.
17. Imad Zeroual, Dirk Goldhahn, Thomas Eckart, and Abdelhak Lakhouaja. 2019. OSIAN: Open Source International Arabic News Corpus - Preparation and Integration into the CLARIN-infrastructure. In Proceedings of the Fourth Arabic Natural Language Processing Workshop, pages 175–182, Florence, Italy. Association for Computational Linguistics.

18. Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In Findings of the Association for Computational Linguistics:

EMNLP 2020, pages 1644–1650, Online. Association for Computational Linguistics

19. Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzman, Edouard Grave, Myle Ott, Luke Zettle moyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8440– 8451, Online. Association for Computational Linguistics.

20. Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multitask benchmark for evaluating cross-lingual generalisation. In International Conference on Machine Learning, pages 4411–4421. PMLR.

21. Suchin Gururangan, Ana Marasovic, Swabha ´ Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining:Adapt language models to domains and tasks. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8342–8360, Online. Association for Computational Linguistics.

22. Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A Neural Probabilistic Language Model. The Journal of Machine Learning Research, 1137–1155.

23. Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

24. Barbieri, Francesco, Luis Espinosa Anke, and Jose Camacho-Collados. "XLM-T: A Multilingual Language Model Toolkit for Twitter." *arXiv preprint arXiv:2104.12250* (2021).

25. Mohammed, Ammar, and Rania Kora. "Deep learning approaches for Arabic sentiment analysis." *Social Network Analysis and Mining* 9.1 (2019): 1-12.

26. Bengio, Yoshua, and Jean-Sébastien Senécal. "Adaptive importance sampling to accelerate training of a neural probabilistic language model." *IEEE Transactions on Neural Networks* 19.4 (2008): 713-722.

27. Andrew A. Naguib, Yassen H. Mehrez, Mina M. Tawfik, Marc E. Erian. Towards featureless and language-independent recognition and synthesis of writing style: application on Arabic and English poem meter classification (Extended)

28. *Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.*

29. Debnath, Alok, et al. "Semantic textual similarity of sentences with emojis." *Companion Proceedings of the Web Conference 2020*. 2020.

30. Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).