Text Mining and Natural Language Processing

2023-2024

Stefania Rosciano id: 513269

Libero Biagi id: 513037

[Libero & Stefania's Diagnosis]

Introduction

In this project we will showcase the performance of two different models for diagnosis classification based on symptoms written in natural language. The task involves a multiclass classification of symptoms for 24 possible diagnoses.

Classification it's an usual NLP task, for which we decided to use two neural models: a Bi-Directional LSTM (Recurrent Neural Network) because of it recurrent structure, processing inputs through a series of temporal steps where each step's output feeds into the next capturing patterns and relations between words. But with base RNNs there are two main problems:

- Not being able to retain long-term dependencies within our input, as our hidden states only take information coming from the previous time step
- Unidirectionality

The first problem can be solved by introducing Long short term memory (LSTM) cells in our model that will compute a context vector by taking the information coming from the hidden states and the input and applying a series of transformations that allow us to select the important information to retain. As we will also consider the context vector of all the previous steps we are able to capture long-term relations.

The problem of unidirectionality can be easily solved by considering Bi-Directional RNNs in which we will stack two layers of RNNs (can also be LSTM gates) that will process our input in opposite directions, concatenating their outputs to form a unique representation.

The second model we will consider is BERT (Bi-Directional Encoder Representations from Transformers) , a Transformer model, that by nature it's bidirectional by making use of the attention mechanism. BERT in specific it's an encoder only Transformer, meaning that it will only make use of Self-attention. The overall structure for this model it's composed of a series of transformer blocks followed by a fully connected dense layer, the variant we chose to use consists of only 12 transformer blocks containing the Self-Attention heads.

At least we will also consider a LSTM variant model adding an attention layer. Attention has the job of dealing with the problem of the *"bottleneck"* for the LSTM. Because all our representation it's compressed into one final hidden state that contains all the relevant information of the sequence, but with attention we are able to consider not only this last hidden state for the final output, but we compute a final context vector that will be the weighted sum of all our hidden states weighted by the

importance of each of them (in our case of *self-attention*). By computing the scores of our hidden states and the last hidden state we can find which hidden states are the ones that contribute the most. In Transformers, attention mechanisms facilitate bidirectionality and long-term relationship modeling.

The data it's taken from the research paper "*Optimizing classification of diseases through language model analysis of symptoms*". It includes 1200 samples of symptoms each paired with one of 24 diagnoses. The dataset is balanced, with approximately 50 samples per class (around 4.2% per class).

Methodology

1. Preprocessing of the text

Text preprocessing involves:

1. Splitting by whitespace, converting to lowercase, removing stopwords (e.g., *the, a, and*), and punctuations.
2. Lemmatization of verbs using *WordNetLemmatizer* (mapping words to their *lemma*: base dictionary form).
3. Padding sequences on the left to ensure uniform length for LSTM inputs.
4. For BERT, using a pre-trained BERT tokenizer to generate input IDs and attention masks.
5. Mapping target labels to unique integers for classification.

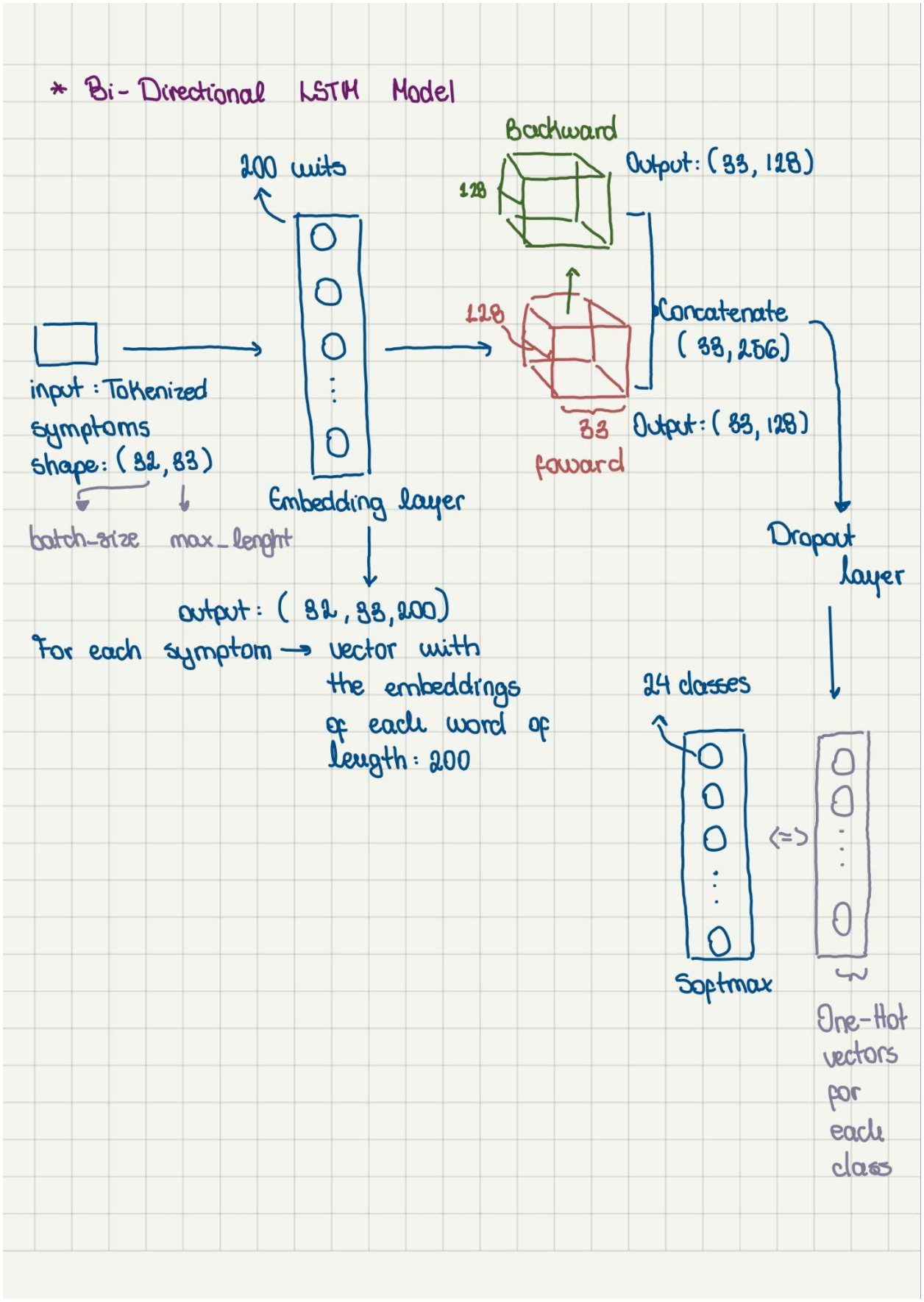
2. Defining the models

- **Bi-Directional LSTM**

The model comprises:

1. An embedding layer (with *mask_zero = True*) to convert inputs into vector representations of dimension *out_dimensions* (tuned as a hyperparameter).
2. A Bi-Directional LSTM layer.
3. Dropout layers for regularization to avoid overfitting.

Adding dense layers post-LSTM worsened performance, possibly due to the model not suffering significantly from the "bottleneck" issue given the small sequence length (33 steps).


* Bi-Directional LSTM Model
Backward
200 units
Output: (33, 128)
128
128
Concatenate
(33, 256)
input: Tokenized
symptoms
shape: (32, 33)
33
foward
Output: (33, 128)
batch_size
max_lenght
Embedding layer
Dropout
layer
output: (32, 33, 200)
For each symptom → vector with
the embeddings
of each word of
length: 200
24 classes
<=>
Softmax
One-Hot
vectors
for
each
class

- **Attention layer**

To enhance the capability of our model to focus on relevant information, we incorporated an attention layer. Following this layer, we employed a dense layer to ensure that the dimensions of the context vector are appropriately aligned before reaching the output layer. This step is crucial, particularly compared to models lacking attention mechanisms. As the output of the attention layer it's a vector of weights computed as the weighted some of the scores with the values vector:

$$\text{context\_vector}(q, V) = \sum_i \text{attention\_weights}(q, V)_i \cdot v_i$$

Introducing a dense layer aids in providing a meaningful representation to these weights, thus refining the model's decision-making process.

For the attention scores (computed between the query and value vector, because we are implementing self-attention the values vector are the previous hidden steps while the query it's the current hidden state) we will use the additive score with the $tanh$ function,

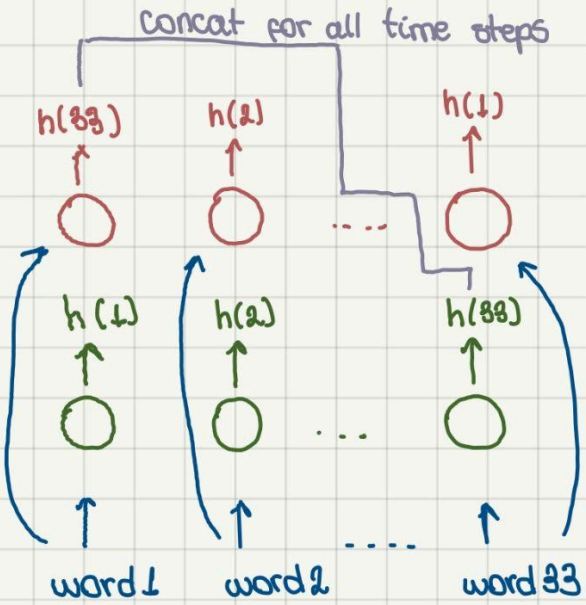
$$\text{score}(q, v) = \tanh(W_q q + W_v v)$$

Which tells us how important two vectors are with respect to each other. In the code this step it's done by the linear projection of the scores to a dimension of 1, in which we will get a scalar (relevance) for each time step. Afterwards these scores we will be transformed into a probability distribution for all time steps using a softmax that will give us the attention_weights

$$\text{attention\_weights}(q, V) = \text{softmax}(\text{score}(q, V))$$

which we will later inspect so we can check which words the model it's paying more attention towards

* The attention layer



Self-Attention : input :

Query
Current h(t)
(32,33, 256)

1. Linearly project

Wq · query

(32, 33, 32)

Value
All other h(t)
(32, 256)

Wv · values

(32, 32)

Add 1 dimension
(32, 1, 32)

tanh(query + values) = scores
↳ Compute the relevance score for each time step

Linearly project

Dimensions $\in \mathbb{R}^{1\times 32}$ ← W_i · scores
= (32, 33, 1)

score = One value for each word

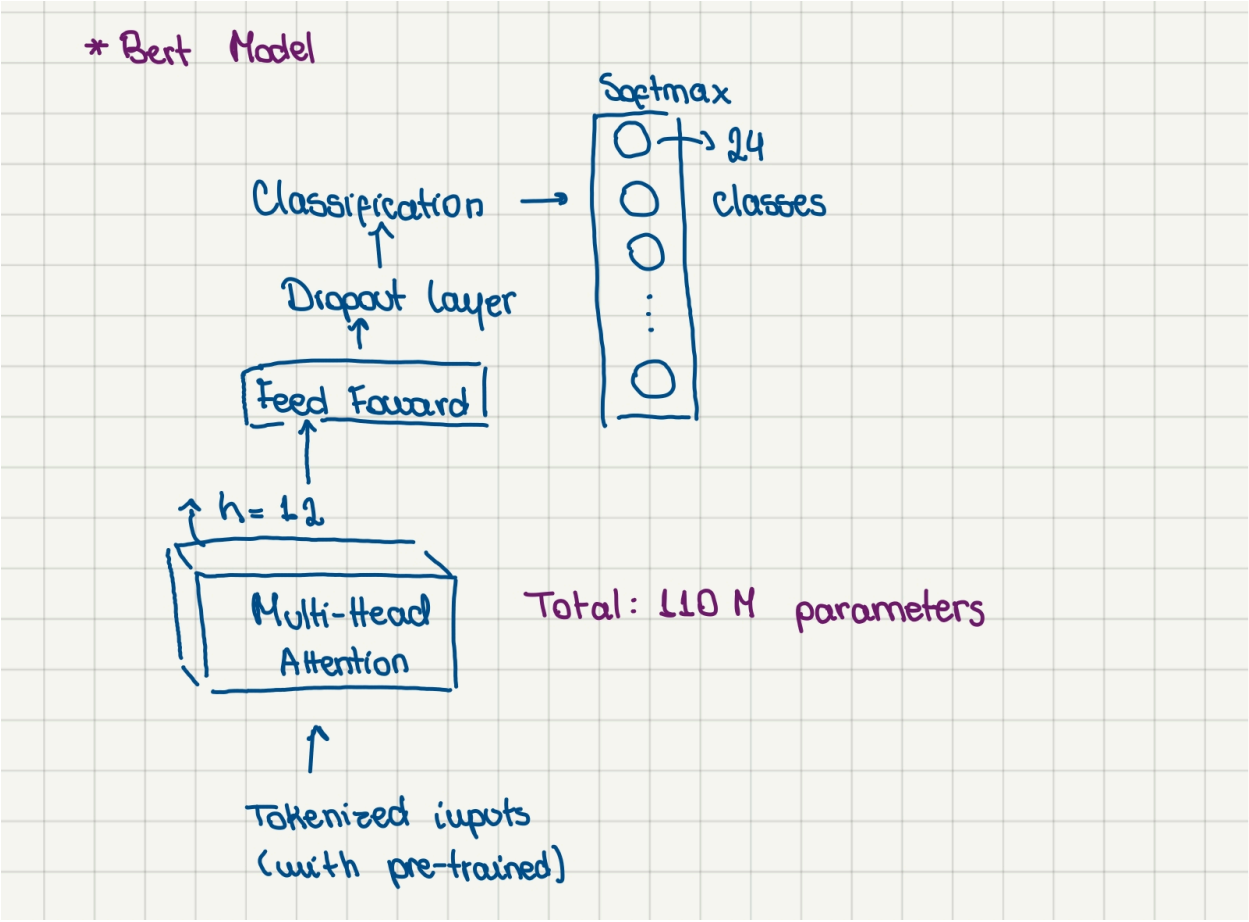
Attention weights

Probabilities : Degree of importance of each word = softmax(score)

↳ Context vector = $\sum$ att_weights · values
output = (32, 256)

- **Fine-Tuning a BERT model**

We will be using the TensorFlow Hub to instantiate the BERT model, converting a PyTorch model to a Keras model, so that all attributes of keras' can be used. The function takes as argument the inputs with corresponding attention masks. Because we are fine-tuning the model, we will add our classification layer specific for our task but the weights will not be "*frozen*" as we will train the weights of the model again on our data. The model we instantiate it's the *Bert Base Model Uncased,* trained with Wikipedia and Bookcorpus for the objective of masked language modeling and next sentence prediction, even though these tasks are not the most aligned for our use case, we hope that the model will still be able to perform fairly well on our "*simple*" dataset. This model has 12 attention heads and in total has 110 million parameters.



3. Hyperparameter Optimization

For hyperparameter optimization and model selection we decided to perform a 5-fold-cross-validation for both the models.

In the cross-validation we are splitting only the train set into validation, leaving the test set we splitted at the beginning unseen as we will check the performance in this test set only with respect to our best model found after cross-validation.

Hyperparameters for LSTM

1. **Embedding Dimension**: 50, 100, 200 dimensions.
2. **Number of LSTM Units**: 32, 64, 128 units (doubled for Bi-Directional LSTM).
3. **Learning Rate**: Uniform distribution between 0.0001 and 0.01.
4. **Dropout Rate**: Uniform distribution between 0.3 and 0.6.

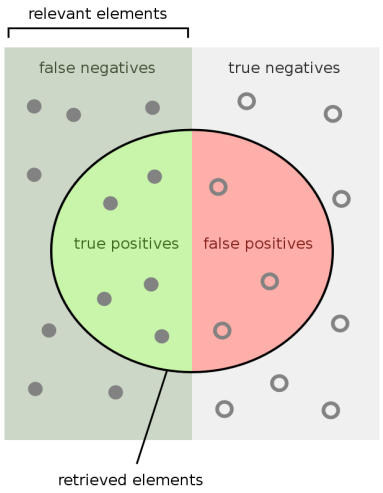
Hyperparameters for BERT

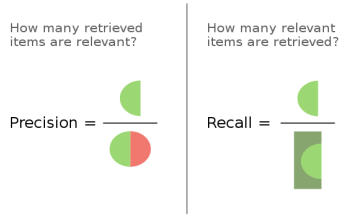
1. **Learning Rate:** Same distribution as LSTM.
2. **Dropout Rate**: Same distribution as LSTM.

Result and Analysis

We will present our results while also comparing them with respect to the ones obtained on the previously mentioned paper.
The main metrics that we are considering are:

- Precision, number of true positive over all the positives (considers false positives)
- Recall, number of true positives over all the positive predictions (also considering false negatives)
- F1-Score: The harmonic mean of precision and recall, because our dataset it's completely balanced the macro, and weighted F1-Scores are equal, so we will consider the macro scores
- Accuracy, the percentage of correct predictions





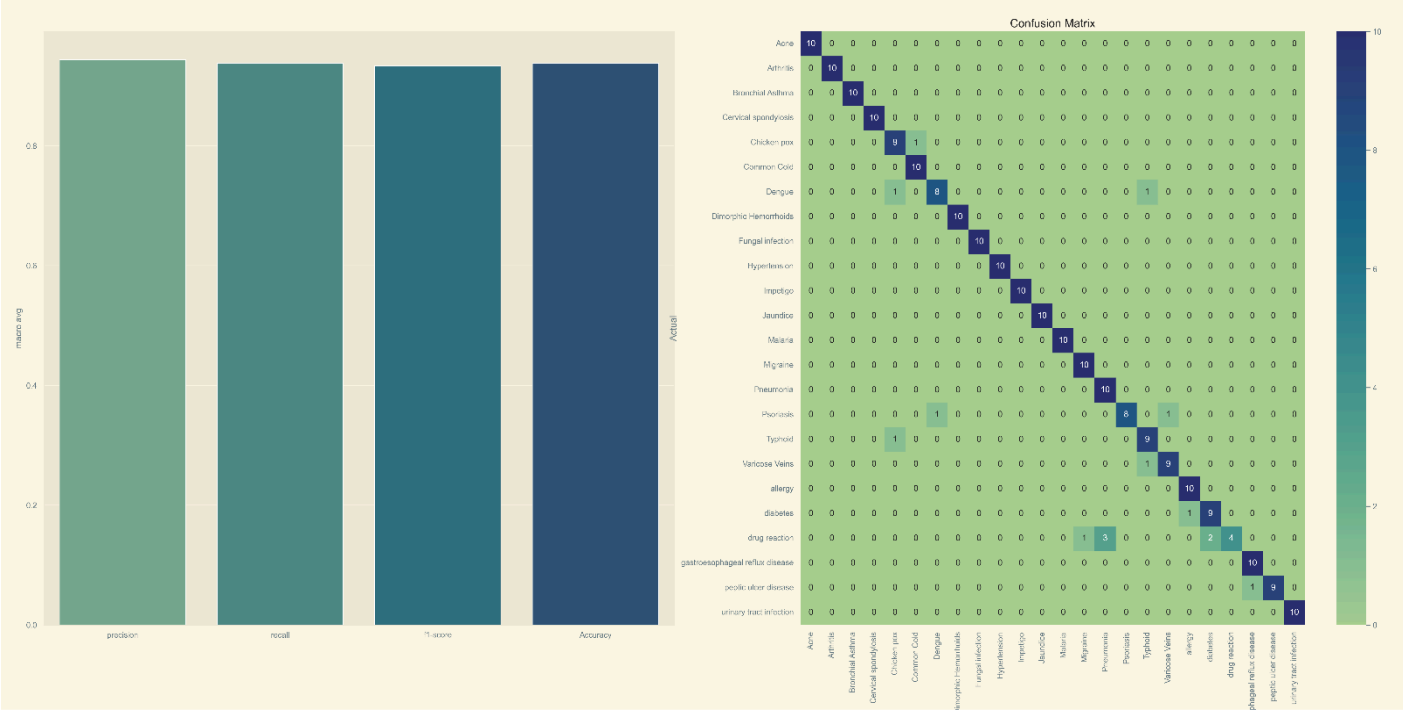
For the Bi-Directional LSTM:

We found that the best hyperparameters were:

- Embedding Dimension : 200
- N_units of the LSTM : 32
- Learning rate : 0.006860548790690411
- Dropout rate : 0.46132499448222186

	Acne	Arthritis	Bronchial Asthma	Cervical spondylosis	Chicken pox	Common Cold	Dengue	Dimorphic Hemorrhoids	Fungal infection	Hypertension	Impetigo	Jaundice	Malaria	Migraine
precision	1	1	1	1	0.8181818	0.9090909	0.8888888	1	1	1	1	1	1	0.9090909090909091
recall	1	1	1	1	0.9	1	0.8	1	1	1	1	1	1	1
f1-score	1	1	1	1	0.8571428	0.9523809	0.8421052	1	1	1	1	1	1	0.9523809523809523
support	10	10	10	10	10	10	10	10	10	10	10	10	10	10

	Pneumonia	Psoriasis	Typhoid	Varicose Veins	allergy	diabetes	drug reaction	gastroesophageal reflux disease	peptic ulcer disease	urinary tract infection	accuracy	macro avg	weighted avg
precision	0.769230769	1	0.8181818	0.9	0.9090909	0.8181818	1	0.90909090909090	1	1	0.9375	0.9437095	0.9437095312095312
recall	1	0.8	0.9	0.9	1	0.9	0.4	1	0.9	1	0.9375	0.9375	0.9375
f1-score	0.869565217	0.888889	0.8571428	0.9	0.9523809	0.8571428	0.5714285	0.95238095238095	0.947368421(	1	0.9375	0.933346	0.933346
support	10	10	10	10	10	10	10	10	10	10	0.9375	240	240



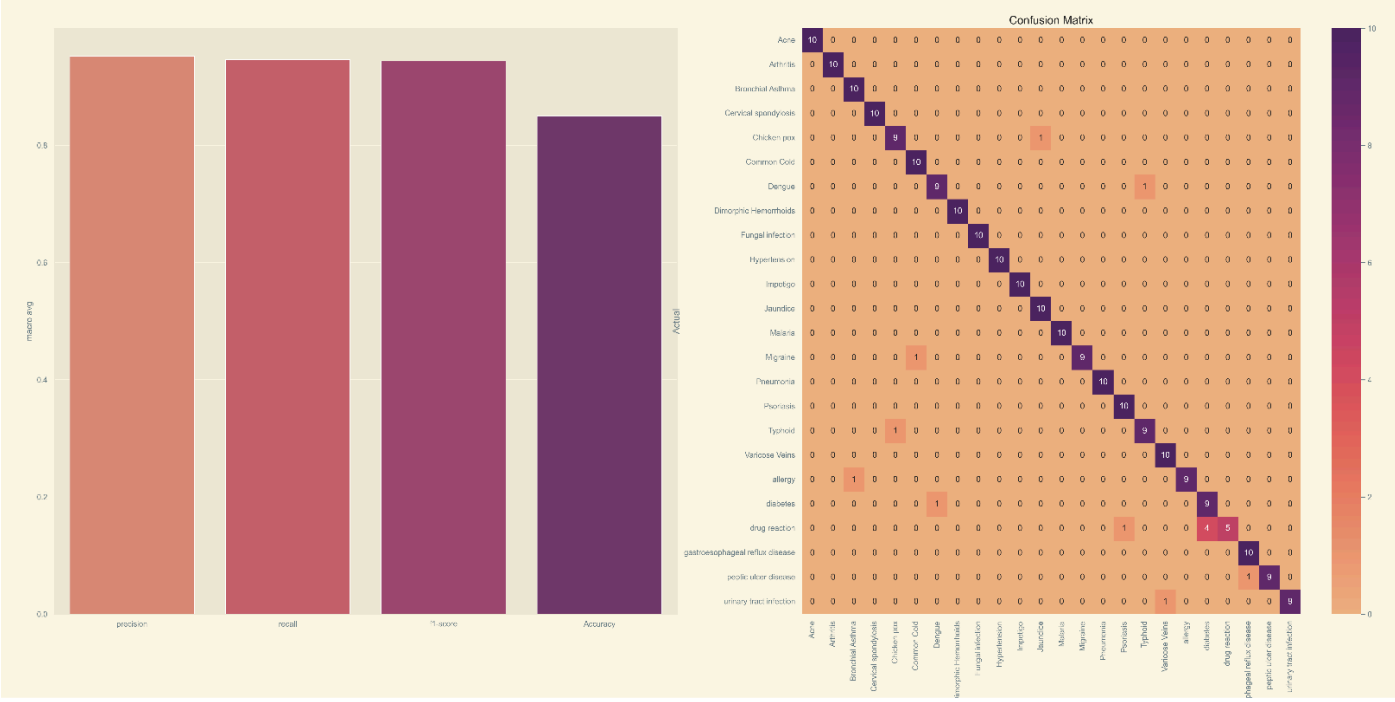
They are close to the ones obtained by the paper, even though they obtained higher scores overall. Probably as they used Hyperopt for hyperparameter optimization and also used a tokenizer specialized to medical terms as stated on the paper "*A preprocessing step involved the meticulous tokenization of these symptom descriptions, achieved through the utilization of a specialized medical tokenizer designed for enhanced contextual understanding of medical terms*"

BiLSTM + Hyperopt	97.08	97.05	97.08	97.37

Bi-DIrectional LSTM with Attention Mechanism

	Acne	Arthritis	Bronchial Asthma	Cervical spondylosis	Chicken pox	Common Cold	Dengue	Dimorphic Hemorrhoids	Fungal infection	Hypertension	Impetigo	Jaundice	Malaria	Migraine
precision	1	1	0.909091	1	0.9	0.909091	0.9	1	1	1	1	0.909091	1	1
recall	1	1	1	1	0.9	1	0.9	1	1	1	1	1	1	0.9
f1-score	1	1	0.952381	1	0.9	0.952381	0.9	1	1	1	1	0.952381	1	0.947368
support	10	10	10	10	10	10	10	10	10	10	10	10	10	10

	Pneumonia	Psoriasis	Typhoid	Varicose Veins	allergy	diabetes	drug reaction	gastroesophageal reflux disease	peptic ulcer disease	urinary tract infection	accuracy	macro avg	weighted avg
precision	1	0.909091	0.9	0.909090909	1	0.692308	1	0.909090909	1	1	0.945833	0.951952	0.951952
recall	1	1	0.9	1	0.9	0.9	0.5	1	0.9	0.9	0.945833	0.945833	0.945833
f1-score	1	0.952381	0.9	0.952380952	0.947368	0.782609	0.666667	0.952380952	0.947368	0.947368421	0.945833	0.943876	0.943876
support	10	10	10	10	10	10	10	10	10	10	0.945833	240	240



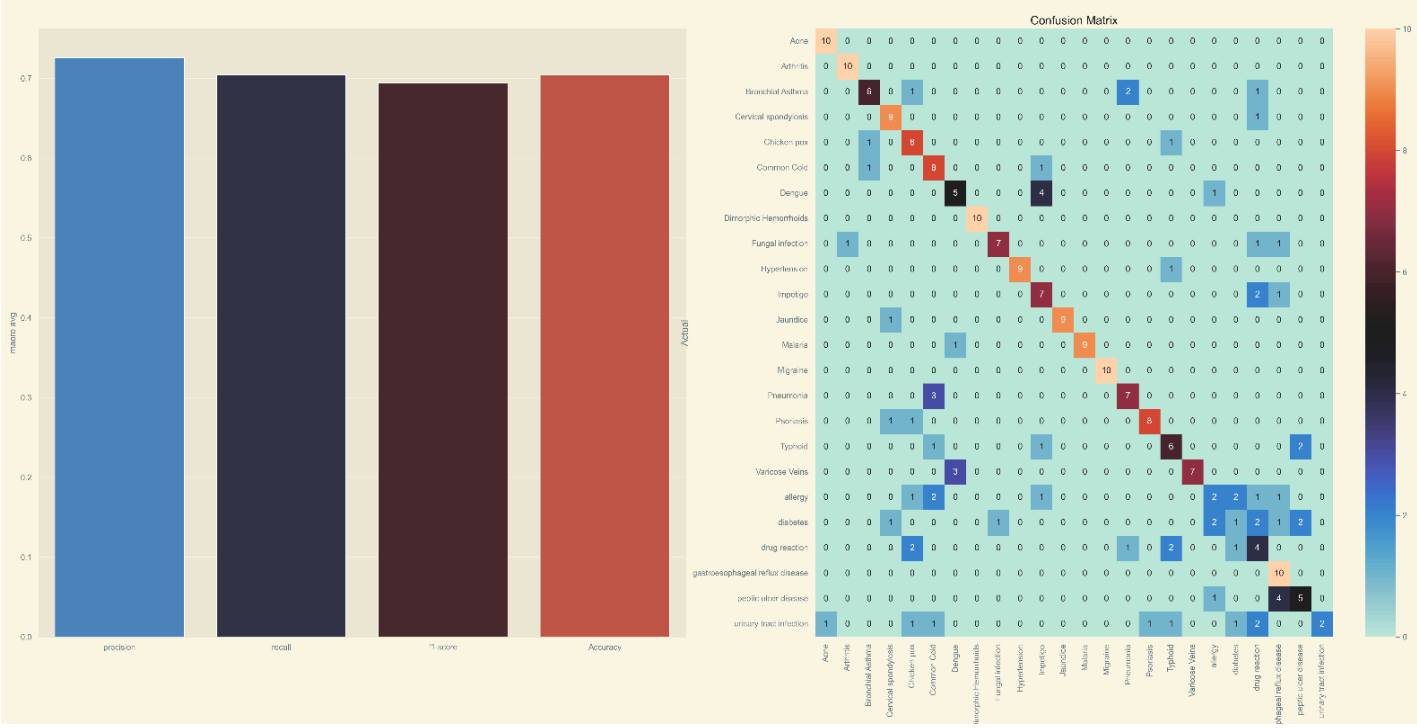
With the attention mechanism we found better results for precision and recall as probably the model with the context vector (weighted sum of each hidden state with the attention weights) was able to just focus on the relevant states for the classification (as we can see by inspecting the attention weights).
In both models though we can see that it was having troubles differentiating drug reactions with diabetes probably due to how the symptoms were written by the patients, in which they could have used similar words.

For BERT:

With the BERT model we found a lot of problems with the instantiation of the model which probably caused some performance problems. Also we chose to fine-tune the model which was probably not the best choice as our dataset it's rather small retraining a large and complex model like BERT probably requires many more examples and epochs as we only consider 50 (same as the paper) because the fitting times were significantly high.
We opted to use the same hyperparameters that were used in the paper (except for the learning rate, in which we used the one found for the LSTM model)

	Acne	Arthritis	Bronchial Asthma	Cervical spondylosis	Chicken pox	Common Cold	Dengue	Dimorphic Hemorrhoids	Fungal infection	Hypertension	Impetigo	Jaundice	Malaria	Migraine
precision	0.909090909	0.9090909	0.75	0.75	0.5714285	0.5333333	0.5555555	1	0.875	1	0.5	1	1	1
recall	1	1	0.6	0.9	0.8	0.8	0.5	1	0.7	0.9	0.7	0.9	0.9	1
f1-score	0.952380952	0.9523809	0.6666666	0.81818181818	0.6666666	0.64	0.5263157	1	0.7777777	0.9473684210	0.5833333	0.9473684	0.9473684	1
support	10	10	10	10	10	10	10	10	10	10	10	10	10	10

	Pneumonia	Psoriasis	Typhoid	Varicose Veins	allergy	diabetes	drug reaction	gastroesophageal reflux disease	peptic ulcer disease	urinary tract infection	accuracy	macro avg	weighted avg
precision	0.7	0.8888888	0.5454545	1	0.3333333	0.2	0.2857142	0.5555555555555	0.5555555	1	0.7041666	0.7257500	0.72575
recall	0.7	0.8	0.6	0.7	0.2	0.1	0.4	1	0.5	0.2	0.7041666	0.7041666	0.7041666666666667
f1-score	0.7	0.8421052	0.5714285	0.82352941176	0.25	0.1333333	0.3333333	0.7142857142857	0.5263157	0.3333333333	0.7041666	0.6938947	0.693894748755429
support	10	10	10	10	10	10	10	10	10	10	0.7041666	240	240



Performances for the BERT model on the paper:

MCN-BERT + AdamP	99.58	99.13	99.28	99.18	669.50
MCN-BERT + AdamW	98.33	98.18	98.23	98.39	688.69

They also fine-tuned BERT with two different optimizers.

Conclusion

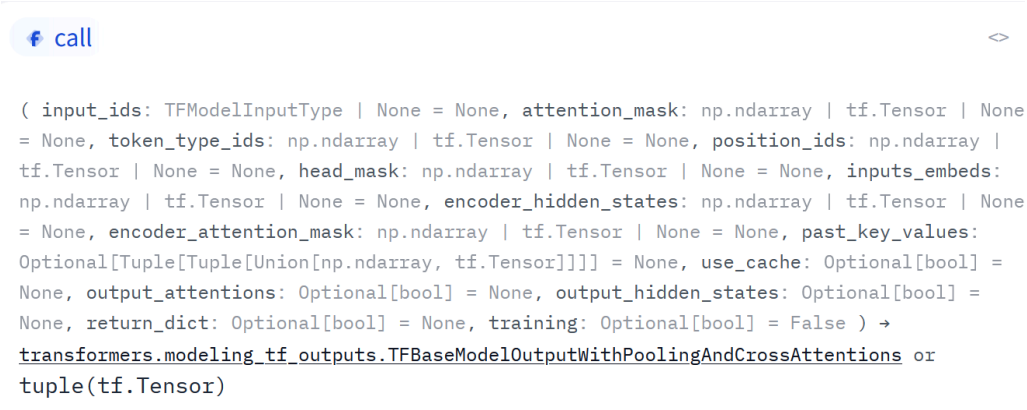
This project demonstrated the efficacy of Bi-Directional LSTM and BERT models for symptom classification tasks. While the LSTM models performed well, the BERT model's performance was limited by the small dataset and high computational requirements. We also demonstrate the importance of hyperparameter optimization as we found better performances with respect to our base model (85%

accuracy).

Some problems:

With the Bi-Directionall LSTM we didn't find many problems, such as the creation and training of the model. With the attention on the other hand we encounter some problems with keras' attention layer, so that's why we decided to instantiate our own attention layer, as the keras' one was having problems with the inputs and the attention masks were not implemented. Even though on our own implementation we weren't able to take into consideration the masks of the padded inputs, after the training of the model we inspected the output of the attention weights as it was giving very low scores to all the padded inputs pointing out that it's working correctly

The biggest problems we faced were with the fine-tuning of the bert model, first of all because we wanted to use keras' we had to transform the PyTorch model into a Keras' one this gave us the warning that some of the weights of the model were not being transferred correctly to the keras' model but we still managed to train the model and also the model we used was not trained specifically for classification (as mentioned above) but when we tried to use a model specifically for sequence classification it wasn't available on keras'. Another problem was that the function takes as argument the inputs with corresponding attention masks. During the model's creation we found a problem that there was a mismatch of keras' versions with respect to the model provided by hugging face, but we managed to solve it by changing the inputs of the BERT model.



Also we wanted to apply cross-validation also with BERT but we weren't able to as the training times were incredibly high so we decided to just take the same hyperparameters that were mentioned in the paper.

Group Organization:

This project was done in pairs. Most of the code and the report were done by Stefania Rosciano while the slides were done by Libero Biagi.

References:

Hassan, E., Abd El-Hafeez, T. and Shams, M.Y. (2024) *Optimizing classification of diseases through language model analysis of symptoms*, *Nature News*. Available at: https://www.nature.com/articles/s41598-024-51615-5#Sec33 (Accessed: 23 June 2024).