Phase 3: Contextual Embeddings

James Furtado[61177], Ricardo Gonçalo[60519], and Diogo Silva[53058]

NOVA School of Science and Technology | FCT NOVA
https://www.di.fct.unl.pt/

1 Introduction

The initial phase of this project involves a study of the code implemented for phase 3. Subsequently, we incorporate a cross-encoder model from the Sentence Transformer library into our project, aiming to enhance the model's ability to understand semantic relationships within text.

For the downstream task of learning to rank, we carefully select a query, a relevant document, and a non-relevant document, utilizing only the detailed description field. The cross-encoder model input is then formatted to align with the requirements of the next sentence prediction task.

Moving forward, we extract the embedding of the CLS token from the last layer and employ the same architecture in the second phase to train a ranking model based on the CLS output embedding. The evaluation process mirrors that of phase 2, providing insights into the model's performance.

The subsequent phases delve into visualizing layer embeddings, analyzing their similarity, and exploring self-attention head visualization. Tokenization is scrutinized as we examine how the tokenizer splits text into tokens. Token embeddings from the first and last layers are extracted, plotted with distinct colors, and critically analyzed for patterns and insights.

To further understand the model's behavior, we compute and visualize the similarity between input and output embeddings across layers. The evaluation extends to comparing the similarities matrix for various query-document pairs, offering a nuanced perspective on the model's performance in capturing semantic relationships.

Finally, self-attention head visualization provides a detailed look into the attention weights of each head. These visualizations are critically analyzed to uncover how different heads contribute to the model's understanding of context and relationships within the text.

Throughout this report, our aim is not only to implement the tasks above but also to critically analyze the results, promoting a deep understanding of the model's behavior and its implications for various natural language processing tasks.

2 Input Formation and Tokenization

The selected cross-encoder model was the one found in: `https://huggingface.co/cross-encoder/qnli-electra-base`

Given our selected cross-encoder model, ce_model.predict computes a relevance score for each document with respect to the query, where higher scores indicate greater relevance. The input to the predict method is a list of pairs, where each pair consists of a query and a document description. The model predicts the relevance score for each pair.

The scores contain the predicted relevance scores for the three documents. Higher scores generally indicate higher predicted relevance. The scores obtained below determine the model's ranking of relevance among the three documents for the given query.

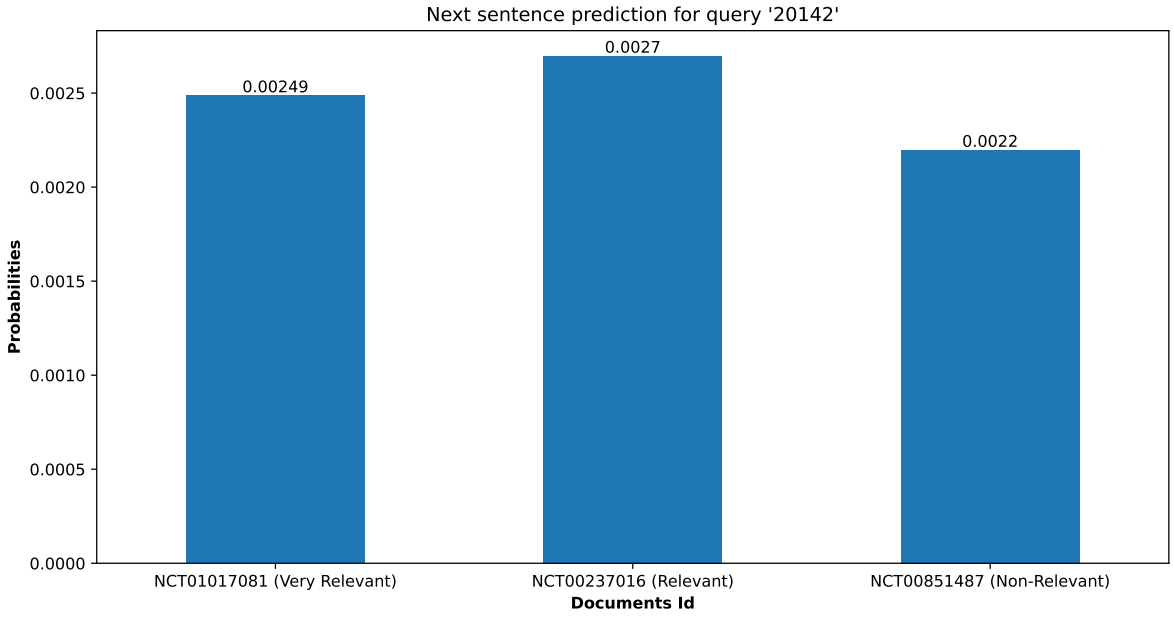

Fig. 1. Next sentence prediction histogram for query "20142"

3 Downstream task: Learning to rank

Now we report how the the new search model we develop on this phase compares with the ones we've developed on the others.

The results are summarized in the figure 2 and 3. We present in this charts three models. The first one is the best model within those our group has developed on the previous phases the **LMJM Language Model**, followed by two models developed on this phase with two different encoders. The **BioBert Bidirectional Encoder** that was trained in biomedical domain and **qnli-electra-base CrossEncoder** trained from data on wikipedia and getting good metrics from the tests[1].

The results shows that the Bidirectional Encoder wins on every single metric and is more precision and has better recall. This didn't surprise us since the encoder was trained on biomedical data. Also we can notice that CrossEncoder model although less performant than the Bidirectional one still outperforms the LMJM Language Model which is great and also expected since we moved to a more precise and a better way of representing words and documents, although it uses more power.

The last thing we want to mention is how the new model works, although it was explained by the professor during the labs how it was supposed to work we want to emphasize the parameters we used and how we implemented it.

3.1 Bert Search Model

To develop the new model, we use the LMJM model to choose calculate p10 on every query and select a set of them for training and another for testing. The way we did was, sort the queries by p10 and choose a few values at every 4-5 query in different positions on the ranking.

Then for each query we generated a set of query-document pairs using the hard negatives strategy introduced during the last phase. We then used google colab to generate CLS tokens for all the pairs using both encoders. Back to our project we just trained a logistic regression model to predict the relevance of a query-document pair giving the CLS token as the x values. The result of the predictions for both the test set and the training set of the qnli-electra-base CrossEncoder are reported on the figures 4 and 5 respectively.

[1] We only included the Bidirectional encoder because we choose it first thinking it was a CrossEncoder. We didn't use consine similarity when generating the model with the Bidirectional Encoder we just generated CLS as we did with the CrossEncoder.

Last but not least we just took the weights of the logistic regression (coef_) and for every query-document pair we would calculate the ranking score multiplying the weights to the CLS token of the pair.

For the results on the figures we used the query-document pair of the test queries.

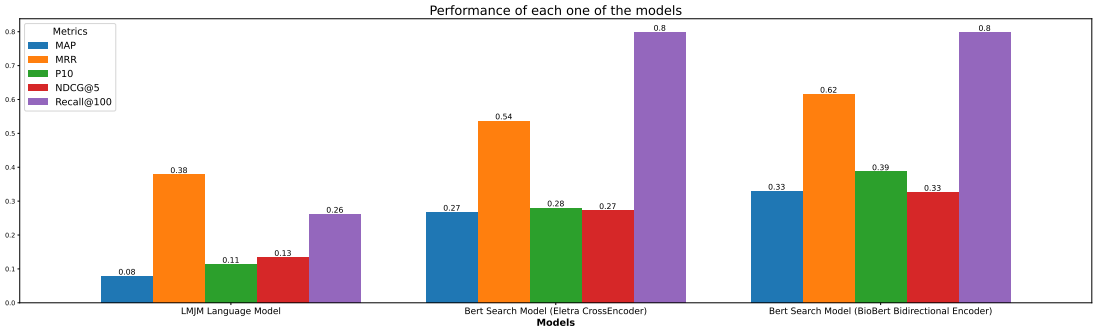

Fig. 2. Metrics Summary for each model

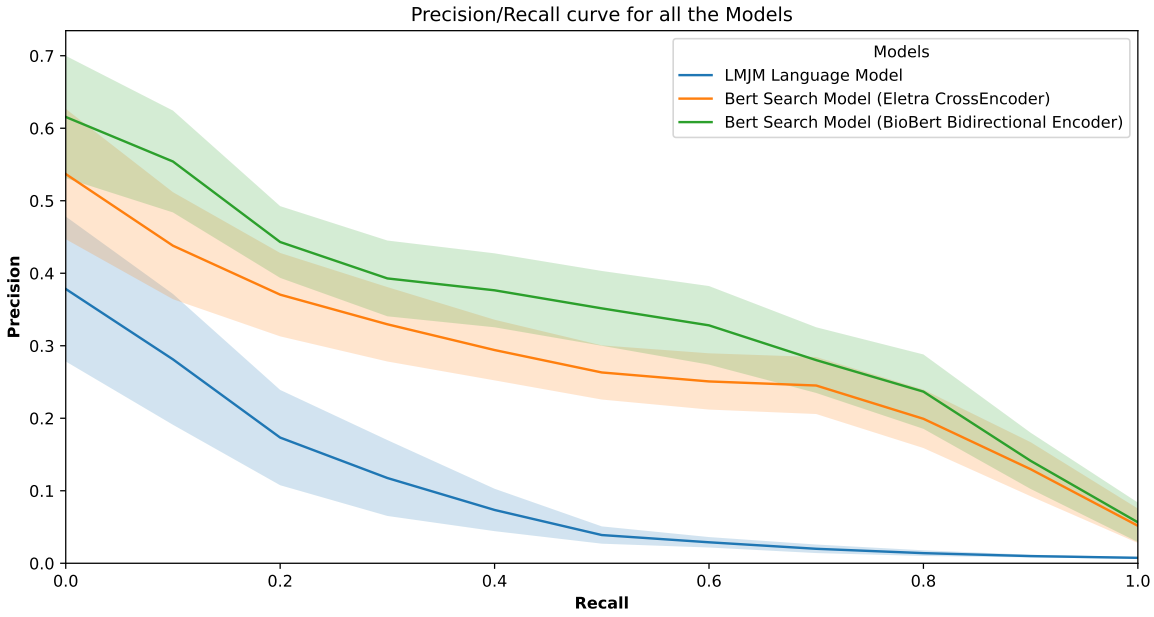

Fig. 3. Precision and recall curve for all the models

	precision	recall	f1-score	support
neg	0.78	0.88	0.83	1748
pos	0.36	0.21	0.27	556
accuracy			0.72	2304
macro avg	0.57	0.55	0.55	2304
weighted avg	0.68	0.72	0.69	2304

Fig. 4. Evaluation report of test data

	precision	recall	f1-score	support
neg	0.79	0.97	0.87	4765
pos	0.84	0.34	0.49	1908
accuracy			0.79	6673
macro avg	0.82	0.66	0.68	6673
weighted avg	0.80	0.79	0.76	6673

Fig. 5. Evaluation report of train data

4 Layer embeddings visualization

When plotting tokens in a scatterplot, we may observe clusters or patterns that reflect the model's learned representations, however there isn´t a very noticable cluster or groups of tokens in the embeddings.

When comparing the token embeddings of the last layer and the first layer wee observe a small difference , which is to be expected since tokens from last layer exhibit more semantic or contextual variance compared to tokens from earlier layers. This is because information has passed through multiple layers, capturing more complex relationships. Also, the last layer captures more high level features and context relevant to the overall task. Since the model used employs self-attention mechanisms, the tokens in the last layer may have attended to a broader context of tokens in the input sequence.

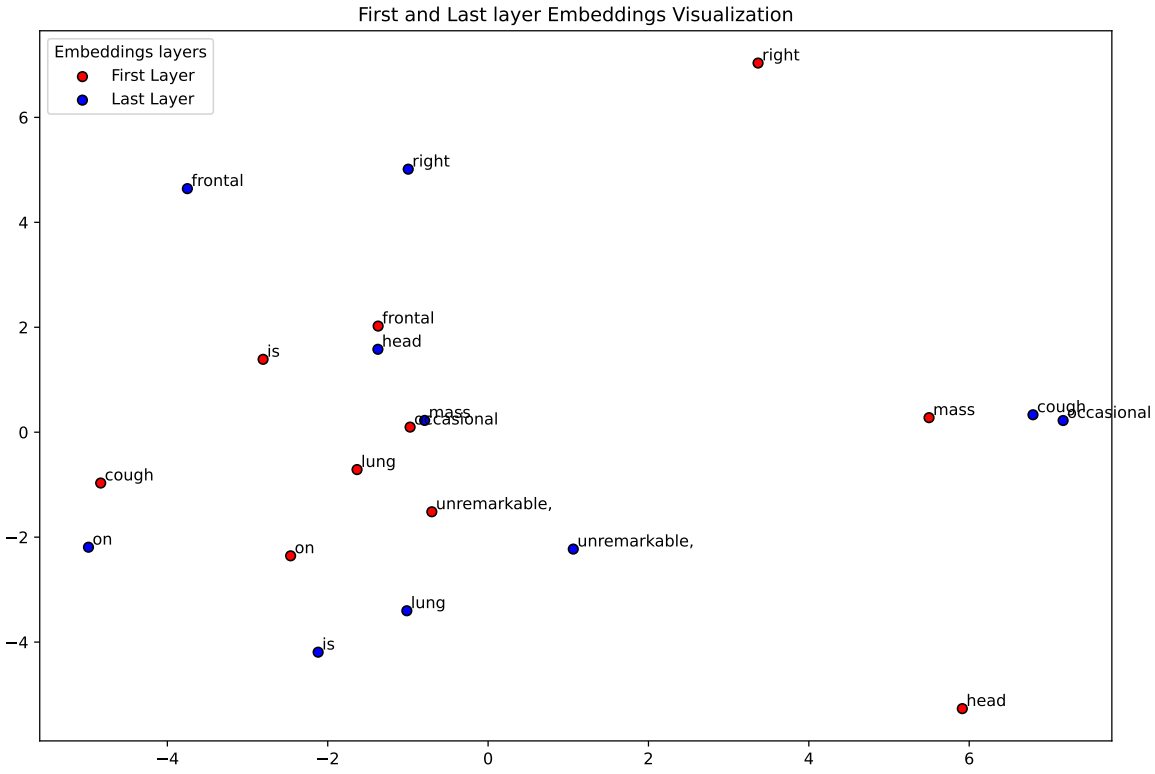

Fig. 6. First and Last layer Embeddings Scatterplot

5 First and Last layer Embeddings Visualization

We can observe from the cosine similarity heatmap between the input and output embeddings of the seconda layer that the w matrixes are probably not capturing most of the relations between the words since the diagonal are high, however they are doing something because they are not 1, we can also observe that the values of input and output embeddings are generally low.

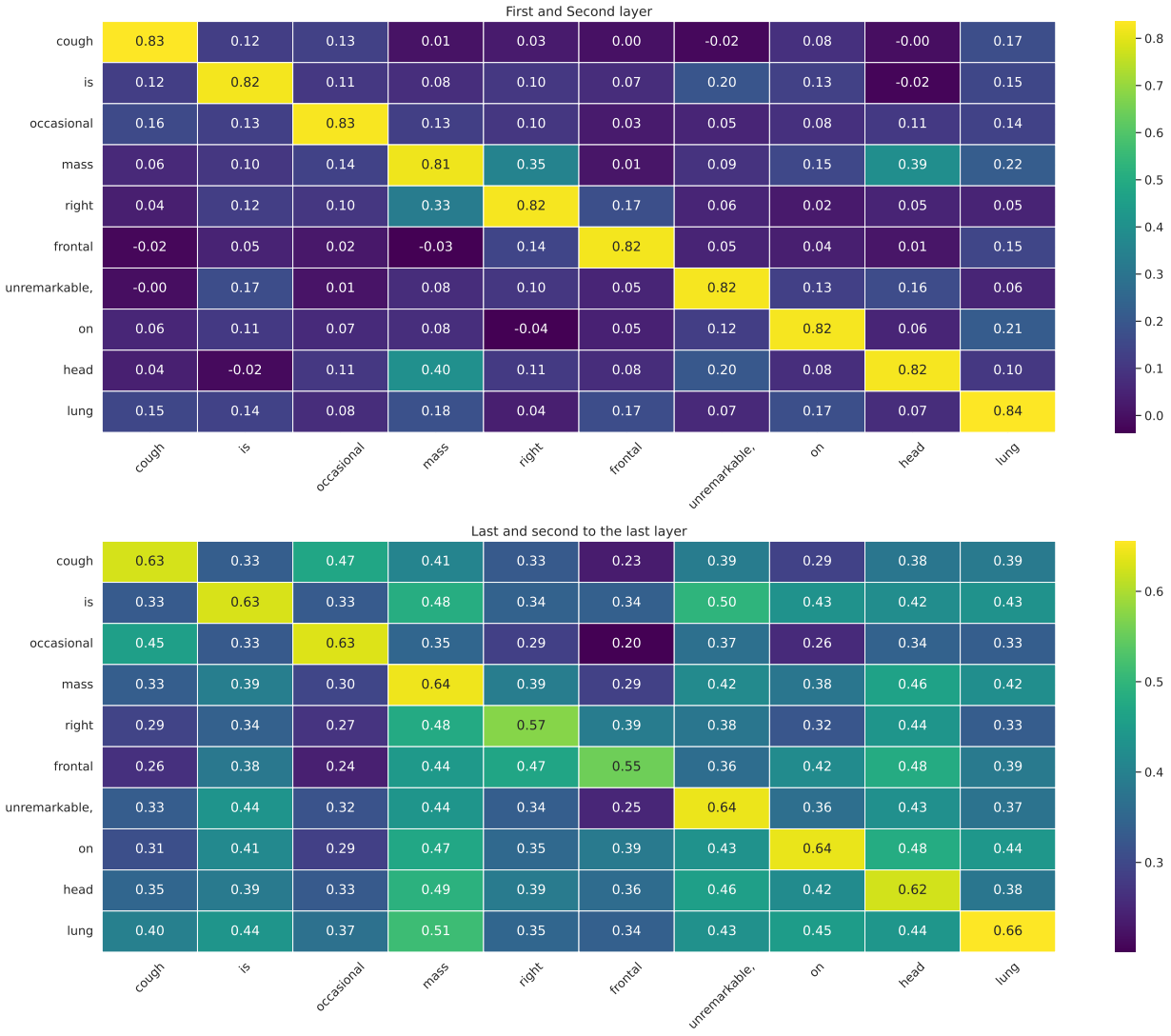

Fig. 7. Heatmap of similarity between first and second layer and between last and second to last layer

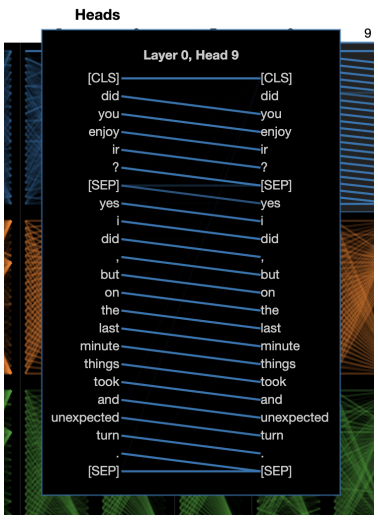

Fig. 8. Attention weights visualization of head 9 layer 0.

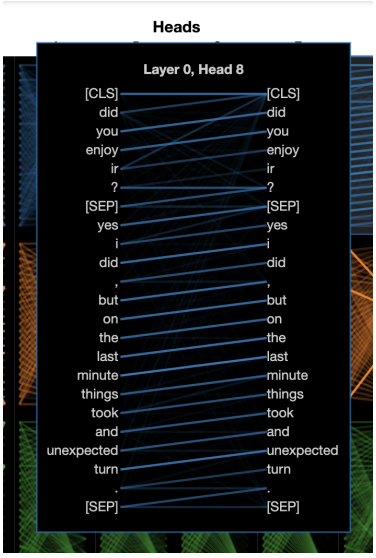

Fig. 9. Attention weights visualization of head 8 layer 0.

6 Self-attention head visualization

Because of complications with colab and because of the deadline we weren't able to calculate the similarities for the queries so we used to nice sentences for the cross encoder and decided to bring up some of the most interesting headings we found.

The sentences we used for the CrossEncoder are: "*Did you enjoy IR?*" and "*Yes I did, but on the last minute things took and unexpected turn.*".

The attentions we see on the figures 8, 9, 10 and 11 shows some interesting things. For example, it seems that on figure 8 the head is capturing the notion of **after** like we can see by the more brighter lines, on the figure 9 on the notion of **before**, on the figure 10 on the **identity** and finally on the figure 11 on a few particular words like *enjoy*, *did*, *minutes*, etc...

This are some conclusion that may be truth and what we have stated are our best guest. It's very hard to try to estimate this notions/relations the w matrixes of a particular head is capturing with as little information as we had, but these are some possible answers.

7 Conclusion

To conclude we just wanted to say that we really enjoyed the process of slowly introducing improvements to our little "search engine". We also had a chance to see a glimpse of what the state of the art looks like which makes us feel really

Fig. 10. Attention weights visualization of head 8 layer 3.

Phase 3

Fig. 11. Attention weights visualization of head 0 layer 1.

greatful. We hope to get more chances to study about the topic on the rest or our master degree.

Finally we wanted to state that our project was structured a little different from the rest of the phases, because we developed part of it on our local machines and another on the google colabs. For short one of the notebook named **ir-project-colab-notebook.ipynb** has code that calculates the cls tokens and most of the figure we presented here on the other hand the notebook named **project.ipynb** has the code that loads the cls that where generated by the other notebook on colab, runs the logist regression and does the evalution of the models.