# Coveo Homework

**Amir Raza**

## 1 Problem

Multiple choice question NLP problem. The task is to create a model that models the probability distribution for a missing word ($[BLANK]$) in a given text, for a given choice of words.

## 2 My code and test probs

**Github code link**: `https://github.com/amir9ume/word_prediction`

Please ask for access. The repository is private currently.

Please find the results.csv file in the shared Github link. There are 6 columns for the 6 choices. Each column contains the prediction probability of that choice.

## 3 Exploratory data analysis

Training data given has 1 column for text, 6 columns for choices, 1 column for the label, and 1 for id. There are 115,527 rows in total. The 6 choices for all text samples, with only 1 correct answer given as label column

At first glance, data seems to have NaN in choices many times. Upon observation, I confirmed that columns choice5 and choice6 are always empty. This is the case for both the train and test sets. We replace these empty spaces with a '$< pad >$' token so that they don't create problems when we model.

On checking the text column, text in the data often has human languages or HTML elements, like div tags, JSON responses. The text contains multi-lingual instances, such as English, French, German (but mostly English and French). Sometimes text also has non-roman script such as Arabic.

$[BLANK]$ token is present at any token position in the text. It can be present at the start, in the middle, or at the end. This implies any model chosen, must be bi-directional. A uni-directional model like GPT is more suitable when prediction is needed only for the final position tokens. BERT or BERT-based models seem suitable for our case as they are bi-directional in nature.

I checked the data set to make sure that there is one $[BLANK]$ in each instance on the text we parse so that there is not a problem later.

Some descriptive statistics of the text:

- Average length of text in the dataset: 56.42
- Maximum length of text in the dataset: 2269
- Minimum length of text in the dataset: 1
- Number of data samples where length of text is greater than 512 tokens: 48 . This number is very insignificant compared to the data size.

I tested to make sure the label is encoded each time as choice1,choice2...choice6. And there are no excess labels such as choice9 or choice20 in the label column.

Preprint. Under review.

## 4 Modeling approach and design choice

**Modelling**

The biggest assumption I make, is that, once I have a decent language model I can predict the missing words based on the probabilities $p(w_t \mid w_{t-1})$. In order to make sure that both English and French are adequately covered by the vocabulary of my model, I opted to work with XLM-Roberta, which is a multi-lingual Transformer model. Hence one single model should be able to predict choice tokens either in English or French. I am assuming that since XLM-Roberta is trained on 2.5T of data from the internet, it should be very good with its word distribution modeling.

I sampled $50\%$ of the training data randomly ($50\%$ of 115k rows). I reduced the data set, with the assumption that a $50\%$ randomly sampled data set would cover enough of the original dataset diversity. I chose to sample only half of the given data, to save on computation time. Ideally, if more time is available, we can fine-tune more with a larger data set. I then split this dataset between training and validation sets in an 80-20 split.

I tokenized the text to retrieve tokens of max length $512$ as just 48 samples in the dataset had more than $512$ tokens. Each of the given choices is also tokenized.

**Design choices**

For the language model task, the model has a language model head on top of the original model. I chose to fine-tune the last layers of the XLM-Roberta model, which was the language model head. Due to computation limitations, I chose not to fine-tune across all layers, as back-propagating through all 12 layers will increase the time requirement. Though ideally, it would be more suited to get better results for newer vocabulary. (rest all previous layers kept frozen)

XLM-Roberta is a multi-lingual model, hence I do not need to check what language an input text is. Model is capable of tokenizing and decoding tokens from both languages (100 languages in total).

Since I was using the pre-trained language model from hugging face, I had to replace the $[BLANK]$ token in the original text with $< mask >$ token, as a cue for the language model task to work.

I first prepared the tokenized text. Sending this tokenized text to the model returns token probabilities at all positions in the text (basically $p(w_t \mid context)$, probability of a word given its context). But I filter these probabilities to only look at the index where the $< mask >$ token was located. This filtered probability is of dimension [vocabulary] and it basically represents the probabilities of all tokens which can be put at the $< mask >$ token position.

I made a small adjustment here. I do not want to find out the probability of all the possible vocabulary tokens which could be put in the $< mask >$ position. Our given problem is comparatively easier, where we already have 6 possible choices. So I only want to find out the best token among the given choices. Hence I looked up probabilities for the choice tokens only.

I tokenize the choices and get their positions in the vocabulary (dim $[250001]$). This makes my task easier, and now in the masked probabilities of the whole vocabulary, I just use the probabilities for the token choices (dim $[6]$).

Using these probabilities, I can get which choice is having the highest probability and this shall be my model's answer. I encode the answer in terms of its position in the one-hot vector of dim $[1, 6]$, and then calculate the cross-entropy loss with the target labels. The cross-entropy loss objective can then be trained for minimization.

## 5 Performance evaluation of the model

Direct out of the box, without any fine-tuning on the coveo dataset, the model gives a cross entropy loss value of 3.60 and an accuracy of $27\%$. A naive approach to predict any of the 6 choices would be $16\%$ accurate. If we always removed the last two columns, then a naive approach would be accurate $25\%$ times. But I do not do that, since the test set may have all 6 acceptable columns instead of having missing values (as was the case with my training data).

In order to improve accuracy, fine-tuning on the data set is required. But I am aware that for such a huge pre-trained model, it is not required to fine-tune for too many epochs. I chose to train for

8 epochs. This is also in part due to computing and time limitations on me, but it gave me enough sense of whether the model approach and design choice are going to work or not.

- **Epoch** 1: Average training loss 3.45, Average Validation loss: 3.42, Validation Accuracy $30\%$
- **Epoch** 2: Average training loss 3.15, Average Validation loss: 3.14, Validation Accuracy $33\%$
- **Epoch** 3: Average training loss 2.88, Average Validation loss: 2.98, Validation Accuracy $36\%$
- **Epoch** 4: Average training loss 2.70, Average Validation loss: 2.77 Validation Accuracy $38\%$
- **Epoch** 5: Average training loss 2.63, Average Validation loss: 2.71 Validation Accuracy $39\%$
- **Epoch** 6: Average training loss 2.32, Average Validation loss: 2.42 Validation Accuracy $43\%$
- **Epoch** 7: Average training loss 2.30, Average Validation loss: 2.36 Validation Accuracy $44\%$
- **Epoch** 8: Average training loss 2.23, Average Validation loss: 2.35 Validation Accuracy $45\%$

I observed that for the 8 epochs, both the training and validation losses went down, without much hyper-parameter tuning, which is promising. I kept a check on the validation loss to make sure that the model is not over-fitting on its training set. The validation accuracy also kept on increasing till $45\%$, by the end of 8 epochs. Further performance improvement could also be seen with better hyper-parameter value tuning, which I have skipped for now, and just used some commonly suggested values.

### 5.1 Prediction bias

Due to the nature of our implementation, the model is not able to handle tokens not already present in the tokenizer library (XLM-Roberta). Given more time, we could try to add these additional tokens to the model vocabulary and then fine-tune the embeddings layer for these new vocabulary additions. This would improve the performance since at the moment many tokens not present in the original library miss out when we do look up for their probabilities $p(w_t \mid context)$ (especially tokens from programming, HTML, or JSON).

I did **qualitative inspection**, where I inspected model outputs for different types of text present in our dataset. The model suffers when there is text not related to human language. If we look at instances of missing word predictions for English and French, the figures, Figure 1 and Figure 2, show that they are working well. But when we observe the prediction for a programming element component, say an HMTL, JSON, or Response body text, Figure 3 shows that the performance is not that good.

But it fails if there is HTML or JSON text describing a response since it does not know how to handle programming elements.

## 6 Discussion of future work

One important thing which would help the most is to add to the tokenizer vocabulary all the choices in this coveo dataset. Since I essentially do a lookup of choice probabilities, if a choice is not present, it misses out. This same model and approach could be trained for lot more epochs to achieve better performance. The train and validation loss values in Section 5 suggest the same. At the moment, I am limited with time and compute, but the current results are promising.

```
_________________
target: tensor([1., 0., 0., 0., 0., 0.])
=================
<s> - do you have any more specific questions around this topic? at the moment i am i
nterpreting it as a starting point for a very wide-ranging discussion. we could certa
inly have a more general discussion if you like, but if there are any key sub-topics
you would like for me to be prepared with, that would be great. - with<mask> to the a
gent panel topic and attach/detach functionality, again, if there's a specific questi
on, please let
regard respect compliment attenti<pad><pad>
regard
=================
```

Figure 1: English word prediction for the $[BLANK]$ position ($< mask >$). Target is showing correct choice position from 1 to 6 choices. The model correctly predicts the first option 'regard' as the answer.

```
target: tensor([0., 1., 0., 0., 0., 0.])
=================
<s> sources et collections. 2. sous collections, sélectionnez la collection dans laquelle vo
us souhaitez ajouter la source. 3. sous sources,<mask> sur ajouter.</s><pad><pad><pad><pad><
pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pa
d><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
<pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><p
ad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad
><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><
pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pa
d><pad><pad><pad>
glo cliquez pé cla<pad><pad>
cliquez
```

Figure 2: French word prediction for the $[BLANK]$ position ($< mask >$) . Target is showing correct choice position from 1 to 6 choices. The model correctly predicts the second option 'cliquez' as the correct answer.

```
target: tensor([0., 0., 1., 0., 0., 0.])
=================
<s> after: <startanalytics> <! -- the excludecoveouseragentprocessor stops the pipeline if t
he request comes from the coveo user agent -- > <! --<processor<mask> :</s><pad><pad><pad><p
ad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad
><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><
pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pa
d><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
<pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><p
ad><pad><pad><pad><pad><pad><pad><pad>
plot spot patch piece<pad><pad>
spot
=================
```

Figure 3: Programming elements prediction. This is a problematic case for the model. Target is showing correct choice position from 1 to 6 choices. The model does not predict the correct choice which is the third option, 'patch'. Instead, it predicts a more commonly used word spot.

## 7 Results

The pre-trained XLM-Roberta model, gives a prediction accuracy of $27\%$ initially. I fine tuned this model for 8 epochs, based on our choice prediction task. Model successfully trained for 8 epochs and the validation accuracy went up to $45\%$ at the end.

This fine-tuned model was used to generate the test results file, 'results.csv'.