Final project - Mind of context

Halasi Vanda Réka **Páncsics Zsombor** **Ragács Attila**

Abstract

Our final project is called *Mind of context* which is a text generator capable of generating Hungarian story book. Our aim is an application which is a grammatically correct, intelligent book writer. After literature research we decided to transfer learn an existing model instead of creating a completely new one from scratch.

1 Model selection

As a first step we had to choose our model. The state of the art model is the Transformers neural network structure, the predecessor of "LSTM" networks. The model was published in 2017[**4**] in "Attention is All You Need" paper. And it is used in several ways for NLP task forexample in "GPT3-form". The model consists of encoder and decoder parts. Which parts does include 6-6 equivalent sub part. The structure of the model is shown on the Figure 1.[**4**] This model is capable to understand the context of the text more proper than ever before and it solves the paralellisation problem of the prev. models by using attention mechanism. The task to this layer is to select and tone the important part of the input-sequence. The weights of this layer are calculated while decoding from the context and it has role by the prediction of the output. Because of the parralelisation the information of the word sequence will be lost, if it wont be stored in a different way. To do so positional encoding is applied in Transformers models.[4, 1]

We had only one problem. It did not exist any pre-trained "GPT-like" Transformer model. Lucly fortune smiled on us we found a simplified, already hungarian pre-trained Transformer model, a BERT-base model (Bidirectional Encoder Representation on Transformers). It does called huBert.[5] And we have chosen this model as the "basemodel" of

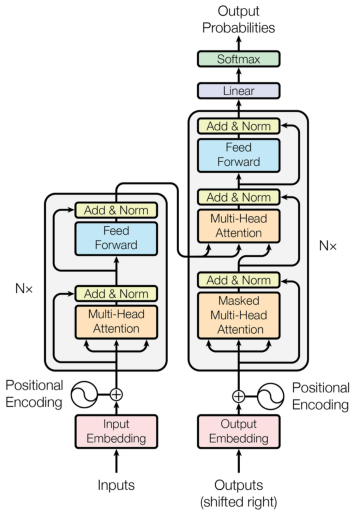

Figure 1: The Transformer - model architecture.

Figure 1: Transformers model structure

our transfer-learning project. It doesnt have the complexity of the GPT2 or 3 models, but it with a smart text-predicting technique it could be able to generate an interesting book...

2 Text preparation

There are two types of *huBERT* model, there is one that was taught with lowercase text and there is one that was taught with uppercase text. We chose the lowercase edition. As searching for the exact format of the text that was used to train the original model, we found that it was broken down into sentences[2]. Taking a look at the *BERT's* documentation we found it has it's own tokenizer, called *PreTrainedTokenizer*[6]. There are several parameters that can be activated in the tokenizer. All the significant special tokens can be found, regarding PAD, EOS, CLS etc. The encoding is implemented as a core part of *LineByLineTextDataset* class which gets a Dataset as a parameter[3].The tokenizer creates the tokens based on the classes implementation. So for

the fine-tuning the tokenizer has been set to a *BertTokenizer.from_pretrained*(”$SZTAKI - HLT/hubert - base - cc$”) and the dataset is being created as a *LineByLineTextDataset* item.

After learning about the original text preparation we started experiment with some variants. We had a hypothesis, that the generation might be better if we created tokens from more sentences. Training the model using tokens from multi

3 Model training

The text-preparation phase were followed by training-testing etaps.To transfer-learn the huBert model we used the built-in trainer module of the Transformers package. The trainings were run on google colab platform with T4 and P100 video cards. A training with P100 vc. did require 5-7 hours to finish 20 epoch on our data set. Our training parameters were: sentence/paragraph broken input text, number of epochs, number of batches. We built 5 models for test purposes. In the beginning each model were trained with random parameter selection than, after testing better and better fine-tuned parameters were applied. Because of the limited time amount of google colab we could not train our models more than 25 epoch, however it comes out that training the model more than 20 epoch leads to overfitting which was reflected by repetative words on the "test output". An other experiment was, that our model learned better by using smaller batches. It could be explained with the small amount of training data which will be more processed with smaller batches than bigger ones.
Our conclusion was: that the best training-paramerets are: 20 training epoch, 32/64 batch size.

4 Text generation

Once we had the fully trained models, it was time to look for some sort of generating method. BERTs are not trained on tasks that would allow us to just feed them text and make it generate text from left to right. Instead we used the MASK task where we replace one token in the sentence with a [MASK] token and use a preexisting pipeline to get predictions from the model regarding what the missing token could be. For example for the sentence 'Today is a great [MASK].' we would get n predictions for the last word (will be important later how many) with their corresponding probabilities.
Usually the best practice is to choose the top prediction, but now we would like to modify our input text and get an output of about the same length. We used 3 different methods in the end, but the main idea is the same. Iterate through certain parts of the input text, mask one token at a time and choose a random prediction from the top 20 (found this number to be optimal) as replacement. Making just enough iterations ensures that the output text will still resemble the original but will change it significantly as well.
In the first method we choose a token randomly from the entire input and did the masking. We did this on half of the tokens hoping to still keep the structure of the original text but make smaller changes that would freshen up the texts. Following this we had a bit more drastic method that relied more on the ability of our models. Instead of only half of the tokens, we targeted those on prime indices (only 2,3 and 5 were used). So this time the selections were fixed and some tokens even got masked multiple times guaranteed. Than we tried a new way of text prediction, by appending new, generated sentences after the original ones. In this method we not only filtered the words, but we used the system to generate fresh new sentences by their own. Finally we went one sentence at a time and using a short left context. This was the idea that resembled traditional text generation most, going from left to right in a way. We did this with a context of 4 sentences, by taking 5 consecutive sentences and only masking tokens in the fifth and final one. This way the very first 4 sentences remained unchanged and then we iterated through these 5-sentence batches, appending the final edited sentence to our output each time.
We feel like the best results came from the first method since the models weren't too developed to form complete sentences that have decent structure. The prime method also did reasonably well, some sentences make sense, some have a correct structure. The apendlast method generated interesting, fresh sentences, however the sentences did not have as much cohesion as before. I could come because of the freedom of the generation, but the missing right context will also add to their effect to the generation. Sadly the context idea

didn't too particularly well, we could explain this by the fact that the first to methods had context from the right (even longer) as well which should strengthen the predictive power of a BERT model. Below are some examples of the generated texts. We could measure the goodness of the predictions only by actually looking at the sentences ourselves, since generating enough text to measure perplexity would have taken days with our resources.

Original text: „Volt egyszer egy kedves, aranyos kislány; aki csak ismerte, mindenki kedvelte, de legjobban mégis a nagymamája szerette: a világ minden kincsét neki adta volna. Egyszer vett neki egy piros bársonysapkát. A kislánynak annyira tetszett a sapka, hogy mindig csak ezt hordta; el is nevezték róla Piroskának. Piroskáék bent laktak a faluban, nagymama pedig kint az erdőben, egy takaros kis házban. Egy szép napon azt mondja Piroskának az édesanyja: - Gyere csak, kislányom!..."
First method „született egyszer egy kedves, barna kislány. aki csak hitte mindenkiről csak legjobban mindig a nagymamája tudta : a világ legnagyobb kincsét nem adta volna. Egyszer kötött neki egy fehér bársonysapkát. A kislánynak annyira tetszett a sapka, hát mindig ő ezt hordta ezt el is nevezték róla Piroskának. Ők bent jártak a faluban, nagymama pedig kint az erdőben, két takaros kis házban. Egy forró nap azt mondja Piroskának az édesanyja : - írd le. . . "
Second (prime) method: „Volt itt egyszer kedves, kicsi kisfiúfia aki mindenhol ismerte, mindenkit szerethette legjobban. a legtöbbet szerettesa enek valamijét azért adta az anyja jutott neki is piroscit A kisfiú azért tetszett neki sapka, mert körülötte mindenki ezt kapta mást nem is szóltak róla. szépen bent élni együtt szegénységben meg az erdőkel aranyos kis kisgyerek aztán ősz napon azzal kérte az apja hogy - gyere csak, fiam . . . "
Third (fresh sentences) method: "..egy kicsi fia született egy király országában s azt gondolta nagy a világ hogy örökké uralkodik egyszer s örökké csak. történet most reggelre is el volt s hogy hogy szép a lánya s felesége volt a neve a leány. az enyém lehet az én lányom vagyok vagyok én nekem a legkisebb is vagy az enyém lesz is vagy. ember a világon nincsen krumpl a szegények mindig jobban vannak mindig reggel s nyáron egyszer egyszer s nyár ellen.a harmadik nap reggel volt reggel reggel volt este volt s este volt volt pedig megint megint éjjel elmúlt megint. mint egy hercegnő a napra a városban a végén s folytatta a maga útját is s a többi többit is. vele mindjárt szépen hozzá egy kicsit mondani is mondani azt hozzá az a kislány a kertben a mit nem látott. több helyen találkozott egymással s az ember a világgal mindig több felé haladt mindig mindig mind egyszer és utoljára pedig." ősz egyszer csak a piros ló megjelenik is hozzá jön hozzá hogy a két arany fia pedig hozzá megy s...."
Last (context) method: „. . . Piroskáék bent laktak a faluban, nagymama pedig kint az erdőben, egy takaros kis házban.takaros kertben erdei csak a nagy szólt nekem a versetj szép kislányom!.már csak Itt itt egy leány nem akar mondani valamit erről világnak.aztán ha holnapután épp hogy elmegynek.Induljunk ki elkészül minden..hőség.. jó halkanon jön még az ugrás s most mindegyek mert talán meghal nagymama!..."

References

[1] Jay Alammart. *Visualizing machine learning one concept at a time.* URL: http://jalammar.github.io/illustrated-transformer/. (accessed: 01.05.2021).

[2] Jacob Devlin. *BERT.* URL: https://github.com/google-research/bert/blob/master/README.md. (accessed: 11.03.2020).

[3] Sylvain Gugger. *huggingfac/transformers.* URL: https://github.com/huggingface/transformers/blob/master/src/transformers/data/datasets/language_modeling.py. (accessed: 21.04.2021).

[4] Ács Judit. *Python NlP Transformers.* URL: https://github.com/bmeaut/python_nlp_2021_spring/blob/main/lectures/09_Transformers_BERT.ipynb. (accessed: 01.05.2021).

[5] David Nemeskey. *huBert.* URL: https://huggingface.co/SZTAKI-HLT/hubert-base-cc. (accessed: 01.05.2021).

[6] The Hugging Face Team. *BERT.* URL: https://huggingface.co/transformers/model%5C_doc/bert.html. (accessed: 7.05.2021).