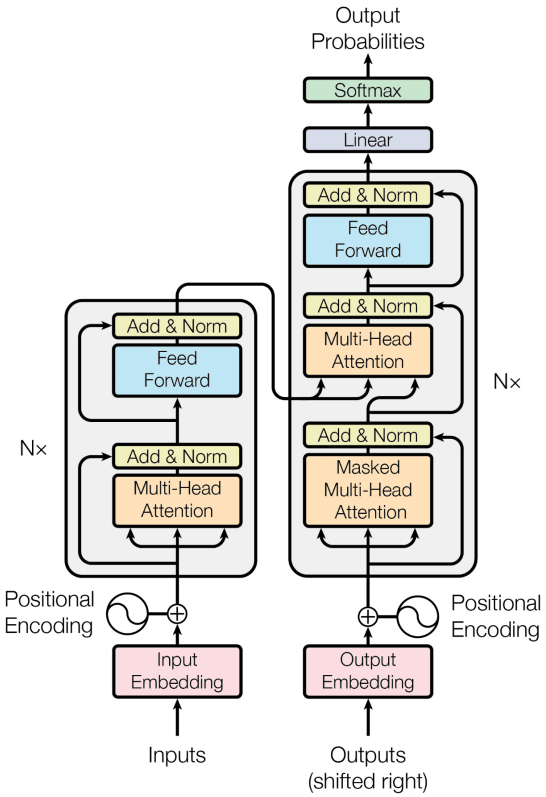
Report on Card 2 - Prática: Introdução a Modelos Transformers (I)

Marcus Vinicius Oliveira Nunes

Activity Description

Transformers are a type of neural network specialized in NLP(Natural Language Processing), which can translate, create, enhance and do other interesting things with text. While CNNs(Convolutional Neural Networks) are specialized in computer vision, Transformers are specialized in NLP.

Differently from other models, transformers use the attention mechanism that instead of, on the translation tasks for example, translating words sequentially It tries to understand the context of each other and the relation between them through the query, key and value mechanism. Transformers can train a big set of data in parallel, which can enhance and boost both the training process and the results since they show a great output.



Seção 8 - Transformers, Bert, GPT e mais

This section presents some of the things that a transformer model can do and how useful they can be, the first one being the questioning and answering that they can do. First I download the model from the huggingface.com. It's a portuguese model developed by Pierre. I instantiate an object model from the model's class.

```
[2] import transformers
    from transformers import pipeline

[3] model = pipeline("question-answering", model="pierreguillou/bert-base-cased-squad-v1.1-portuguese")
```

Next I create some kind of text or copy it from some article and make a question based on the article. Both are stored in variables. After that, I call my object and get an answer storing it into a variable. My experiments can be seen below.

```
[4] text = "Steven Paul Jobs (São Francisco, 24 de fevereiro de 1955 – Palo Alto, 5 de outubro de 2011)[2] foi um inventor, empresário e
    pergunta = "Quem foi Steven Paul Jobs?"
    resposta = model(question=pergunta, context = text)

[5] print(pergunta)
    print(resposta)

    Quem foi Steven Paul Jobs?
    {'score': 0.6625030040740967, 'start': 102, 'end': 166, 'answer': 'inventor, empresário e magnata americano no setor da informática'}

[6] pergunta2 = "Quando nasceu Steven Jobs?"
    resposta2 = model(question=pergunta2, context= text)

    print(resposta2)

    {'score': 0.8030586838722229, 'start': 33, 'end': 56, 'answer': '24 de fevereiro de 1955'}
```

Another cool feature that transformers have is the ability of filling holes of texts. If a text is missing a word, transformers can predict what word can fill that hole, another way to call this is fill mask.

An example can be seen below, predictions by the model can be seen. [MASK] is the hole in the text, and predictions can be seen in the output space. Like "Busan é uma cidade da Coreia do [Mask]", the model predicted both "Coreia do Sul" and "Coreia do Norte"

```
mask = pipeline("fill-mask", model="neuralmind/bert-base-portuguese-cased")
```

```
[8] text = mask("Existe uma chance do imperador cair do[MASK]")

    for x in range(len(text)):
      print(text[x])
```

```
{'score': 0.4944414794445038, 'token': 8242, 'token_str': 'céu', 'sequence': 'Existe uma chance do imperador cair do céu'}
{'score': 0.20068956911563873, 'token': 8271, 'token_str': 'cavalo', 'sequence': 'Existe uma chance do imperador cair do cavalo'}
{'score': 0.11563429236412048, 'token': 4899, 'token_str': 'trono', 'sequence': 'Existe uma chance do imperador cair do trono'}
{'score': 0.025017688050866127, 'token': 18537, 'token_str': 'Céu', 'sequence': 'Existe uma chance do imperador cair do Céu'}
{'score': 0.009439321234822273, 'token': 6711, 'token_str': 'avião', 'sequence': 'Existe uma chance do imperador cair do avião'}
```

```
[9] text = mask(["Existe uma chance do imperador cair do[MASK]", "Busan é uma cidade da Coreia do [MASK]"])

    for x in range(len(text)):
      print(text[x])
```

```
[{'score': 0.4944414794445038, 'token': 8242, 'token_str': 'céu', 'sequence': 'Existe uma chance do imperador cair do céu'}, {'score
[{'score': 0.8622966408729553, 'token': 1422, 'token_str': 'Sul', 'sequence': 'Busan é uma cidade da Coreia do Sul'}, {'score': 0.12
```

Another useful undertaking that can be performed by a transformer model is the summarization of large texts. I took a large text from wikipedia on Busan, a South Korean city and told the model to summarize it. The text got way smaller.

Summarization

```
summarize = pipeline("summarization")
```

```
[11] text = """Busan (hangul: 부산; hanja: 釜山) (pronúncia: [pu.sen]), anteriormente conhecida como Pusan e agora oficialmente Cidade

     resumo = summarize(text, max_length= 70, min_length = 20)
```

```
[12] print(resumo)
```

```
[{'summary_text': ' Busan is a segunda cidade mais populosa da Coreia do Sul (atrás da capital Seul) Sua área é de 770,04 km² .
```

Text generation is another really useful feature. I used it to make texts larger, although the outputs didn't make any sense, it was nice to see.

Text Generation

```
generator = pipeline("text-generation", model="pierreguillou/gpt2-small-portuguese")
```

```
[14] text = "A China está se superando cada vez mais na área da tecnologia, até ultrapassando os EUA."
     resultado = generator(text, max_length =60, do_sample = True)
     print(resultado)
```

```
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncat
[{'generated_text': 'A China está se superando cada vez mais na área da tecnologia, até ultrapassando os E
```

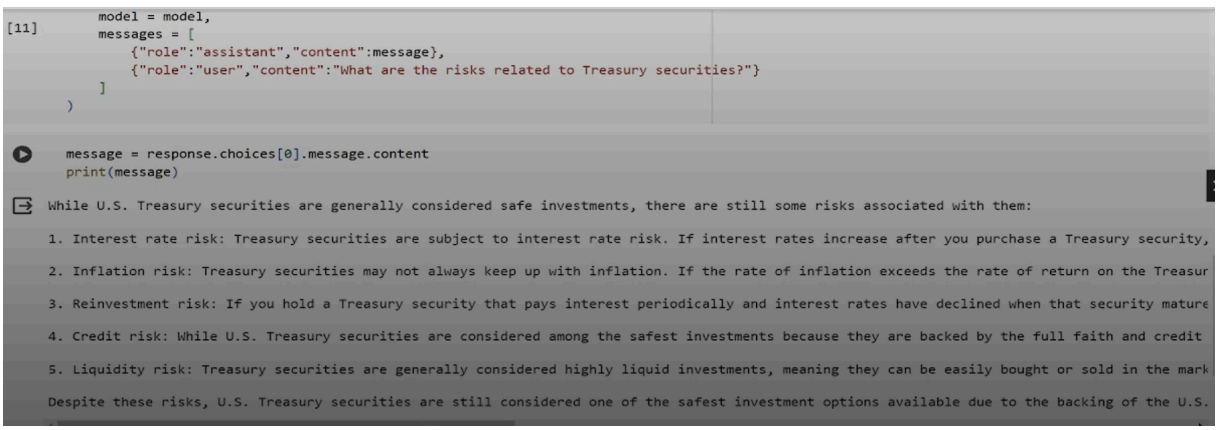
```
[15] text2 = "George Lucas criou o universo Star Wars"
     resultado = generator(text2, max_length =200, do_sample = True)
     print(resultado)
```

```
[{'generated_text': 'George Lucas criou o universo Star Wars e o universo de Star Wars Odyssey, um dos mui
```

It’s also possible to use the Generative Pretrained Transformer/GPT from the main site they provide. Unfortunately due to problems with the API I couldn’t run the code and had only to pay attention on what the teacher was saying but basically the first thing to be done was load the model and authenticate the token.

Next step is to set the model and put the content, it works like the chat in Chat GPT you put what you want there and GPT will answer as it can be seen below.

By doing this I can put context in my conversation with GPT, it will consider the first statement to answer the second one.



Conclusion

Transformers are pretty much the new way of dealing with Natural Language Processing, they can generate, summarize, fill and answer questions. They have better performance and train faster than other models using self-attention.

References

https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf