Report on Card 1 - Prática: Fundamentos de NLP (I)

Marcus Vinicius Oliveira Nunes

Activity Description

This card is about Natural Language Processing, a branch in the artificial intelligence field that aims to train AI models that can comprehend text, words and speaking. This field is more present in our life than we can notice, auto correct used in google docs for instance is made of a NLP(Natural Language Processing) model. The other things that a NLP model can do is classify email as spam or not, compare if two or more texts are alike, make suggestions of next words that can come after another, turn large texts smaller etc. Although it's considered a hard field, it has machine learning algorithms that can make it easier.

Here are the steps made to teach a machine learning algorithm how to comprehend language:

1. **Segmentation:** This step is responsible to break the text into sentences;
2. **Tokenization:** This step breaks the sentences into words;
3. **Stop Words:** Removes the stop words like "the", "and", "to" because they are not necessary;
4. **Stemming:** This step removes the prefixes and suffixes from words;
5. **Speech Tagging:** This step tags words into nouns, verbs, prepositions etc;
6. **Name entity Tagging:** This step introduces the models to a group of words;
7. **Machine Learning:** Training.

Seção 3 - NLP com Spacy

Spacy is a natural language processing library for python, which contains models that work with this field, for the video it was downloaded a large model which has more features than lighter ones.

The first thing that needs to be done is import spacy and instantiate a new object that loads the large model I want to work with. As it shows down below, nlp is an instantiated object.

Tinkering with Spacy

```
[ ] import spacy
```

```
[ ] nlp = spacy.load('pt_core_news_lg')
```

```
[ ] print(type(nlp))
```

```
<class 'spacy.lang.pt.Portuguese'>
```

Before making use of the features that Spacy gives to me, it's necessary to apply the pipeline to them, which contains these steps. Note that it's possible to add new steps and remove them as well.

```
#By calling this method I can check all the steps that are in the pipeline and make part of it

print(nlp.pipe_names)
```

```
['tok2vec', 'morphologizer', 'parser', 'lemmatizer', 'attribute_ruler', 'ner']
```

Here is the result, texts are turned into "spacy.tokens.doc".

```
[ ] document = nlp("Hoje de noite decidi gastar mais de 20 reais com jogos no Brasil, gastei menos
    document2 = nlp("Alegaçõese extraordinárias requerem evidências extraordinárias. Carl Sagan:
```

```
len(document.vocab)
```

```
384
```

```
[ ] print(type(document))
```

```
<class 'spacy.tokens.doc.Doc'>
```

```
[ ] for token in document:
      print(token.text)
```

```
Hoje
de
noite
decidi
gastar
mais
de
20
reais
com
jogos
no
Brasil
,
gastei
menos
de
20
%
do
meu
salário
então
não
me
incomodou
muito
.
A
Inglaterra
é
melhor
```

Spacy can also do other cool stuff like check token's properties like they are upper case, lower case, stop words etc.

```
[ ] #By calling some methods I can check if a certain token fits a certain property

    print("Tokens:", [token.text for token in document])
    print("Stop word:",[token.is_stop for token in document])
    print("Alfanumérico:", [token.is_alpha for token in document])
    print("Maiúsculo:",[token.is_upper for token in document])
    print("Minúsculo:",[token.is_lower for token in document] )
```

```
Tokens: ['Hoje', 'de', 'noite', 'decidi', 'gastar', 'mais', 'de', '20', 'reais', 'com', 'jog
Stop word: [False, True, False, False, False, True, True, False, False, True, False, True, F
Alfanumérico: [True, True, True, True, True, True, True, False, True, True, True, True, True
Maiúsculo: [False, False, False, False, False, False, False, False, False, False, False, Fal
Minúsculo: [False, True, True, True, True, True, True, False, True, True, True, True, False,
```

```
[ ] print("Tokens:", [token.text for token in document])
    print("Formato:",[token.shape_ for token in document])
```

```
Tokens: ['Hoje', 'de', 'noite', 'decidi', 'gastar', 'mais', 'de', '20', 'reais', 'com', 'jog
Formato: ['Xxxx', 'xx', 'xxxx', 'xxxx', 'xxxx', 'xxxx', 'xx', 'dd', 'xxxx', 'xxx', 'xxxx', '
```

I can also use Pos tagging to check other properties, Pos tagging will give me information about each token, telling me if it's a noun, verb, adjective etc.

```
for token in document:
  print(token.text, " - ", token.pos_, " - ", token.dep_, " -
```

```
Hoje - NOUN - nsubj - hoje - Xxxx
de - ADP - case - de - xx
noite - NOUN - nmod - noite - xxxx
decidi - VERB - ROOT - decidir - xxxx
gastar - VERB - xcomp - gastar - xxxx
mais - ADV - advmod - mais - xxxx
de - ADP - fixed - de - xx
20 - NUM - nummod - 20 - dd
reais - NOUN - obj - real - xxxx
com - ADP - case - com - xxx
jogos - NOUN - obl - jogo - xxxx
```

Another useful thing that Spacy can be used for is to compare the similarity between, two sentences or even tokens.

A similarity finder can tell me level of similarity between two sentences

```
newdocument = nlp("Ele gosta muito de comer macarrão")
newdocument2 = nlp("Ela gosta muito de comer macarrão")

#The level of similarity here is pretty high, the only difference here is the word "Ele"
print(newdocument.similarity(newdocument2))
```

```
0.9586358242850286
```

```
# We can even get the similarity between two tokens

document3 = nlp("A palavra é comprimento ou cumprimento?")
tokenA = document3[3]
print(tokenA)
tokenB = document3[5]
print(tokenB)
print(tokenA.similarity(tokenB))
```

```
comprimento
cumprimento
0.5804340243339539
```

Another really interesting tool that can be used is the matcher, with it it's possible to find and extract information in texts, all that needs to be done is set a pattern of what I would like to be found and let it work. In the classes the teacher made a matcher that finds phone numbers in texts, without mattering the way it's written.

Matching, finding patterns

```
[ ] from typing import Match
    from spacy.matcher import Matcher

    document4 = nlp("Você pode ligar para (51) - 9964656570 ou (11) 12344988")

    matcher = Matcher (nlp.vocab)
    pattern = [{"ORTH": "("}, {"SHAPE":"dd"}, {"ORTH": ")"}, {"ORTH": "-", "OP": "?"}, {"IS_DIGIT'
    matcher.add("telefone", [pattern])
    matches = matcher(document4)
    for id, inicio, fim in matches:
      print(document4[inicio:fim])
```

```
(51) - 9964656570
(11) 12344988
```

Spacy also has a cool feature to display things in a more pleasant and better way.

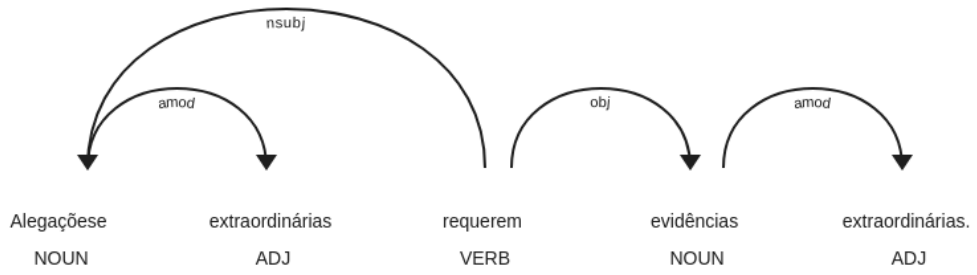
Hoje de noite decidi gastar mais de 20 reais com jogos no Brasil LOC , gastei menos de 20% do meu salário então não me incomodou muito. A Inglaterra LOC é melhor

```
[ ] displacy.render(document2, style="ent",jupyter= True)
```

Alegaçõese ORG extraordinárias requerem evidências extraordinárias. Carl Sagan PER :' https://carlsagan.com MISC '

```
document2.user_data['title'] = "Exemplo"

displacy.render(document2, style="dep",jupyter= True, options ={'compact': False, 'distance':
```



Seção 4 - NLP com NLTK

Like Spacy, NLTK is a NLP library. It's described by the teacher as older and pretty famous. Unlike Spacy, it has a more verbose import section. These are all the imports and downloads I had to do in order to use the features.

```python
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer, LancasterStemmer
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.tag import pos_tag, pos_tag_sents
import string

nltk.download('punkt_tab')
nltk.download("stopwords")
nltk.download("punkt")
nltk.download("tagsets")
nltk.download("wordnet")
nltk.download("average_perceptron_tagger")
nltk.download("maxent_ne_chunker")
nltk.download("words")
nltk.download('tagsets_json')
nltk.download('averaged_perceptron_tagger_eng')
nltk.download('maxent_ne_chunker_tab')
```

Like with Spacy and probably not different to other machine learning libraries, the first thing I had to do was tokenize the text. The next thing I did was removing the stop words that are not considered necessary for the process. They just occupy space. I also removed the punctuations.

```python
Text = "O sol brilha forte no céu. As crianças correm pelo parque. Os pássaros cantam nas árvores. O vento sopra suavemente."
```

```python
print(Text)
```

```
O sol brilha forte no céu. As crianças correm pelo parque. Os pássaros cantam nas árvores. O vento sopra suavemente.
```

```python
sentencas = sent_tokenize(Text, language = "portuguese")
```

```python
print(tokens)
```

```
['O', 'sol', 'brilha', 'forte', 'no', 'céu', '.', 'As', 'crianças', 'correm', 'pelo', 'parque', '.', 'Os', 'pássaros', 'cantam', 'nas', 'árvores', '.', 'O', 'vento', 'sop
```

```python
stops = stopwords.words("portuguese")
print(stops)
```

```
['a', 'à', 'ao', 'aos', 'aquela', 'aquelas', 'aquele', 'aqueles', 'aquilo', 'as', 'às', 'até', 'com', 'como', 'da', 'das', 'de', 'dela', 'delas', 'dele', 'deles', 'depois
```

```python
palavras_sem_stop = [p for p in tokens if p not in stops]
print(Text)
print(palavras_sem_stop)
```

```
O sol brilha forte no céu. As crianças correm pelo parque. Os pássaros cantam nas árvores. O vento sopra suavemente.
['O', 'sol', 'brilha', 'forte', 'céu', '.', 'As', 'crianças', 'correm', 'parque', '.', 'Os', 'pássaros', 'cantam', 'árvores', '.', 'O', 'vento', 'sopra', 'suavemente', '.
```

```python
print(string.punctuation)
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

```python
palavras_sem_poncuacao = [p for p in palavras_sem_stop if p not in string.punctuation]
print(palavras_sem_poncuacao)
```

```
['O', 'sol', 'brilha', 'forte', 'céu', 'As', 'crianças', 'correm', 'parque', 'Os', 'pássaros', 'cantam', 'árvores', 'O', 'vento', 'sopra', 'suavemente']
```

Just like in Spacy, NLTK also contains Pos tagging, which can be seen down below:

```
[ ] pos = nltk.pos_tag(palavras_sem_poncuacao)
    print(pos)
```

```
[('O', 'NNP'), ('sol', 'NN'), ('brilha', 'NN'), ('forte', 'NN'), ('céu', 'NN'), ('As', 'IN'), ('
```

```
token2 = sent_tokenize(Text)

ntokens = []
for tokensentenca in token2:
  ntokens.append(word_tokenize(tokensentenca))

print(ntokens)
possentenca = pos_tag_sents(ntokens)
print(possentenca)
```

```
[['O', 'sol', 'brilha', 'forte', 'no', 'céu', '.'], ['As', 'crianças', 'correm', 'pelo', 'parque
[[('O', 'NNP'), ('sol', 'NN'), ('brilha', 'NN'), ('forte', 'VBZ'), ('no', 'DT'), ('céu', 'NN'),
```

It also contains Lemmatization and Tagging.

Lemmatization

```
[ ] lemmatizer = WordNetLemmatizer()
    resultado = [lemmatizer.lemmatize(palavra) for palavra in palavras_sem_stop]
    print(palavras_sem_poncuacao)
    print(resultado)
```

```
['O', 'sol', 'brilha', 'forte', 'céu', 'As', 'crianças', 'correm', 'parque', 'Os', 'pássaros'
['O', 'sol', 'brilha', 'forte', 'céu', '.', 'As', 'crianças', 'correm', 'parque', '.', 'Os',
```

```
[ ] Text2_en = "Dom João foi rei de Portugal"
    token3 = word_tokenize(Text2_en)
    tags = pos_tag(token3)
    en = nltk.ne_chunk(tags)
    print(en)
```

```
(S
  (PERSON Dom/NNP)
  (PERSON João/NNP)
  foi/VBD
  rei/FW
  de/FW
  (GPE Portugal/NNP))
```

Machine Learning Models for NLP - Article

This article talks about Machine Learning models for Natural Language Processing, which is a subfield of artificial intelligence that aims to enable computers to understand and generate human spoken language. Another subject that is pointed by the article is the evolution of machine learning models and what is their purpose. Machine learning models are algorithms that learn and predict data based on the relation between input and output data. They get better throughout the training process and learn to make their predictions more accurate.

Some early machine learning models like the **SVM(Support Vector Machine), RNNs(Recurrent Neural Networks)** and **CNNs(Convolutional Neural Networks)** were used for natural language processing, not at the start for the CNNs, which were used primarily for image processing.

Later, Transformer models started to rise and introduced state of the art results. This type of model learns the complex relations between words through self-attention mechanisms. Transformers are known to be used in GPT.

Conclusion

Although it's considered a difficult field due to the complexity of human spoken languages, machine learning models are making it easier to tinker with NLP. Spacy and NLTK provide excellent tools of natural language processing models that can perform really interesting tasks.

References

Natural Language Processing In 5 Minutes | What Is NLP And How Does It Wor…

https://medium.com/@harishdatalab/machine-learning-models-for-nlp-ff4010e7dd06