02 - Data Preprocessing

August 3, 2020

1 EA Assignment 02 - Data Preprocessing

Authored by: Álvaro Bartolomé del Canto (alvarobartt @ GitHub)

We will start this Jupyter Notebook with a little recap from the previous one named `01 - Data Exploration.ipynb` where we explored the available data and extracted some conclusion and useful details that may be useful during this Jupyter Notebook, so please, check the previous Notebook before proceeding.

So on, we will be using the same `Loading Data` and `Cleaning Data` Jupyter cells in order to load the data and clean it (since there were some invalid/duplicated values), respectively.

1.1 Loading Data

[1]:
```python
import glob
```

[2]:
```python
directories = glob.glob('../documents_challenge/*')
directories
```

[2]:
```
['../documents_challenge/Wikipedia',
 '../documents_challenge/Conference_papers',
 '../documents_challenge/APR',
 '../documents_challenge/PAN11']
```

[3]:
```python
data = list()
```

[4]:
```python
%%time

for directory in directories:
    context = directory.split('/')[-1].lower()

    for subdir in glob.glob(f"{directory}/*"):
        lang = subdir.split('/')[-1].lower()

        for file in glob.glob(f"{subdir}/*"):
            data.append({
                'lang': lang,
                'context': context,
                'text': open(file, 'r').read()
            })
```

```
CPU times: user 423 ms, sys: 117 ms, total: 540 ms
Wall time: 541 ms
```

[5]:
```
import pandas as pd

data = pd.DataFrame(data)
data.head()
```

[5]:
```
  lang    context                                               text
0   en  wikipedia   Watchmen is a twelve-issue comic book limite…
1   en  wikipedia  The Citigroup Center (formerly Citicorp Cente…
2   en  wikipedia  | birth_place = | death_date = | death_place …
3   en  wikipedia  Marbod or Maroboduus (born c. in 30 BC, died …
4   en  wikipedia The Sylvester Medal is a bronze medal awarded …
```

1.2 Cleaning Data

[6]:
```
duplicated_data = data[data.duplicated(subset=('text',), keep='first')]
duplicated_data
```

[6]:
```
      lang            context  \
13651   en  conference_papers
13701   en  conference_papers
13725   en  conference_papers
13752   en  conference_papers
13790   en  conference_papers
…      …                  …
21451   en              pan11
22259   es              pan11
22726   es              pan11
22821   es              pan11
23072   es              pan11

                                                    text
13651  This approach naturally involves an agglomerat…
13701  We can see that all these proposals have in co…
13725  Contribution of conceptual vectors to lexical …
13752  Since version 2, relations as derivationally r…
13790  This article describes conceptual\n vectors th…
…                                                  …
21451   \n\nFor Juanita, who had spent all day sewing…
22259  La hermosa canción, que canta Margarita mientr…
22726   Volvió a los dos meses, muerto de hambre, mal…
```

```
22821    (N. de la E.)\n\n[32] Nochebuena chiquita, as…
23072    Otro elemento se reitera igualmente en sus no…

[116 rows x 3 columns]
```

[7]:
```python
data.drop_duplicates(subset=('text',), keep='first', inplace=True)
data.shape
```

[7]:
```
(23012, 3)
```

[8]:
```python
data = data[data['text'] != 'translation not available']
data.shape
```

[8]:
```
(23011, 3)
```

At this point we have already loaded and cleaned the data as defined in the previous Jupyter Notebook, so now we can proceed with the NLP Data Preprocessing.

1.3 Defining Pre-Processing Steps

As already stated, the preprocessing is one of the most relevant steps in a NLP pipeline, since when we preprocess text we intend to give additional value to the text, which means that we are enriching our raw data in order to help the model out before we feed it.

When it comes to NLP preprocessing there are some common steps since it is usual that the text is not unified into lower case, so it contains both upper and lower characters, a common piece of text contains stopwords such as pronouns, determinants, etc., if the text has been downloaded from Internet it may contain HTML tags, it may also contain multiple spaces or line breaks, etc.

So we will just try to cover that in a really generic way first, but then in a more detailed one, since the stopwords are different depending on the language, a language model can be applied for either stemming or lemmatization, etc.

1.3.1 Text Cleaning

We will use the Python library `unidecode` for the text cleaning so as to transform any string into a unidecoded one, which in our case, this library will just remove all the accents from both Spanish and French text, and maybe from English texts if there's any (by mistake or maybe because the text mentions words in other languages).

[9]:
```python
from unidecode import unidecode
```

Unidecode example/s presented below:

[10]:
```python
unidecode("Même si je travaille chez EA, j'aimerais continuer à étudier")
```

[10]:
```
"Meme si je travaille chez EA, j'aimerais continuer a etudier"
```

```
[11]: unidecode("Aunque trabaje en EA, me gustaría seguir estudiando")
```

```
[11]: 'Aunque trabaje en EA, me gustaria seguir estudiando'
```

As we may have seen, the accents have dissappeared but the vowels which contained those accents are still in there, so they have not been removed.

1.3.2 Regular Expressions

Now we will proceed with defining the regular expresions, which in this case as already mentioned, will be used for: removing the URLs (URL_PATTERN), removing the HTML tags and values (HTML_PATTERN), removing the punctuation signs/marks (PUNCTUATION_PATTERN), to just keep the characters and discard the numbers and any other characters (NUMBER_PATTERN) and to remove multiple spaces (SPACES_PATTERN).

```
[12]: import re

      URL_PATTERN = re.compile(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:
       ↪%[0-9a-fA-F][0-9a-fA-F]))+')
      HTML_PATTERN = re.compile(r'<.*?>|&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});')
      PUNCTUATION_PATTERN = re.compile(r'[^\w\s]')
      NUMBER_PATTERN = re.compile(r'[\d]+')
      SPACES_PATTERN = re.compile(r'[ ]{2,}')
```

Regular Expressions example/s presented below:

```
[13]: URL_PATTERN.findall("Please, visit our web at https://www.ea.com/")
```

```
[13]: ['https://www.ea.com/']
```

```
[14]: HTML_PATTERN.sub(" DELETE ", "<b>Avez-vous déjà joué à Rocket Arena?</b>")
```

```
[14]: ' DELETE Avez-vous déjà joué à Rocket Arena? DELETE '
```

```
[15]: PUNCTUATION_PATTERN.findall("¿Qué tal ha parecido la presentación de EA?, mola,␣
       ↪¿no?")
```

```
[15]: ['¿', '?', ',', ',', '¿', '?']
```

```
[16]: NUMBER_PATTERN.sub(" DELETE ", "I've just spent up to 345 hours playing EA's␣
       ↪Harry Potter Quidditch Word Cup game.")
```

```
[16]: "I've just spent up to  DELETE  hours playing EA's Harry Potter Quidditch Word
      Cup game."
```

```
[17]: SPACES_PATTERN.findall("Les jeux EA?        simplement le meilleur")
```

```
[17]: ['        ']
```

As we have already seen, the regular expressions work as expected since they replace, remove, find, etc. the matching parts of the text to the introduced regular expression in each case.

1.3.3 Stopwords Removal

So we will proceed with the next part, which is the stopword removal part, where we will use a corpus from one of the main NLP libraries for Python which is named NLTK (Natural Language Tool Kit), which contains a corpus of stopwords in multiple languages and, in this case, in the languages we need to solve the problem we are facing to.

[18]:
```
from nltk.corpus import stopwords

spanish_stopwords = stopwords.words('spanish')
english_stopwords = stopwords.words('english')
french_stopwords = stopwords.words('french')
```

Stopwords removal example/s presented below:

[19]:
```
texto = "me gustaría trabajar en ea".split()

for palabra in spanish_stopwords:
    texto = list(filter((palabra.lower()).__ne__, texto))

print(' '.join(texto))
```

```
gustaría trabajar ea
```

[20]:
```
texte = "j'aimerais travailler chez ea".split()

for mot in french_stopwords:
    texte = list(filter((mot.lower()).__ne__, texte))

print(' '.join(texte))
```

```
j'aimerais travailler chez ea
```

[21]:
```
text = "i'd love to work for ea".split()

for word in english_stopwords:
    text = list(filter((word.lower()).__ne__, text))

print(' '.join(text))
```

```
i'd love work ea
```

As we may have seen the stopword removal does not work as well as expected fot both English and French, since those languages tend to use the apostrophe (') to shorten words that appear together, mainly after personal pronouns when they are followed by a verb which starts by a vowel or some other rules.

1.3.4 Additional Steps

Anyway, we will define a step in order to solve it, which will split the characters before and after the apostrophe, and the apostrophe will be removed.

[22]:
```
"j'aimerais travailler chez ea".replace("'", " ")
```

[22]:
```
'j aimerais travailler chez ea'
```

As we can see above, we fixed this issue creating a new rule which should be applied to both English and French texts in order to split the words that contain an apostrphe and removing the apostrophe. Anyway, we will still include this step into the Spanish Preprocessing Pipeline, since some words from other languages may be found so as to ensure that they are all preprocessed the same way.

1.3.5 Stemming (Discarded)

Finally, we will proceed with the Stemming, which is a NLP method to reduce each word to their root, which means that different words with the same root with be transformed to their root so that those words are the same, which can add value to the preprocessing since we have a lot of different words from different language, and this is a way to reduce the size of the input data we will be using to feed the model.

In this case, we will also be using NLTK since it contains stemmers for English, Spanish and French, which should cover all our needs for now. So on, we will just transform each word to its root.

[23]:
```
from nltk.stem import SnowballStemmer

english_stemmer = SnowballStemmer('english')
spanish_stemmer = SnowballStemmer('spanish')
french_stemmer = SnowballStemmer('french')
```

Stemming example/s presented below:

[24]:
```
texto = "algun dia trabajaré para ea".split()
resultado = list()

for palabra in texto:
    resultado.append(spanish_stemmer.stem(palabra))

print(' '.join(resultado))
```

```
algun dia trabaj par ea
```

[25]:
```
text = "someday i'll be working for ea".split()
result = list()

for word in text:
    result.append(english_stemmer.stem(word))
```

```
print(' '.join(result))
```

```
someday i'll be work for ea
```

[26]:
```
texte = "un jour je travaillerai pour ea".split()
resultat = list()

for mot in texte:
    resultat.append(french_stemmer.stem(mot))

print(' '.join(resultat))
```

```
un jour je travaill pour ea
```

Update: Stemming will be removed, since we will not know the language in which an input text is written, and the model is supposed to classify any input text in its context regarless the language, so we do not need anything this specific on the preprocessing part.

1.3.6 Conclusion

So on, in order to conclude, we already defined and tested all the NLP preprocessing steps that we will need to accomplish the task of preprocessing the data so as to feed the model in the next notebook. Anyway, we still need to implement it as some Python interfaces regarding the language, so when data is received, we will just need to apply the defined preprocessing function using the required interface.

1.4 PreProcessing Pipeline

Once we designed all the NLP Preprocessing pipeline steps we will procceed to its implementation over a random sample text so as to see how it works and in order to initially evaluate its performance.

[27]:
```
from random import choice

sample_lang, sample_context, sample_text = data.iloc[choice(range(len(data)))]
sample_lang, sample_context, sample_text
```

[27]:
```
('en',
 'apr',
 'i bought this book because i wanted to have a clear idea on how to build straw
"greb. it is particularly well studied. and makes you want to build straw. well
explained and effectively illustrated this book will certainly refer to my
future site. the author even provided the essential key to the calculation of
structure in the case of small buildings, it is concise but effective.')
```

[28]:
```
sample_text = sample_text.replace('\t', ' ').replace('\n', ' ')
sample_text
```

```
[28]: 'i bought this book because i wanted to have a clear idea on how to build straw
      "greb. it is particularly well studied. and makes you want to build straw. well
      explained and effectively illustrated this book will certainly refer to my
      future site. the author even provided the essential key to the calculation of
      structure in the case of small buildings, it is concise but effective.'
```

```
[29]: sample_text = unidecode(sample_text)
      sample_text
```

```
[29]: 'i bought this book because i wanted to have a clear idea on how to build straw
      "greb. it is particularly well studied. and makes you want to build straw. well
      explained and effectively illustrated this book will certainly refer to my
      future site. the author even provided the essential key to the calculation of
      structure in the case of small buildings, it is concise but effective.'
```

```
[30]: patterns = (
          URL_PATTERN, HTML_PATTERN, PUNCTUATION_PATTERN,
          NUMBER_PATTERN, SPACES_PATTERN
      )

      for pattern in patterns:
          sample_text = pattern.sub(' ', sample_text)

      sample_text
```

```
[30]: 'i bought this book because i wanted to have a clear idea on how to build straw
      greb it is particularly well studied and makes you want to build straw well
      explained and effectively illustrated this book will certainly refer to my
      future site the author even provided the essential key to the calculation of
      structure in the case of small buildings it is concise but effective '
```

```
[31]: sample_text = sample_text.strip().lower()
      sample_text
```

```
[31]: 'i bought this book because i wanted to have a clear idea on how to build straw
      greb it is particularly well studied and makes you want to build straw well
      explained and effectively illustrated this book will certainly refer to my
      future site the author even provided the essential key to the calculation of
      structure in the case of small buildings it is concise but effective'
```

```
[32]: sample_text = sample_text.replace("'", " ")
      sample_text
```

```
[32]: 'i bought this book because i wanted to have a clear idea on how to build straw
      greb it is particularly well studied and makes you want to build straw well
      explained and effectively illustrated this book will certainly refer to my
      future site the author even provided the essential key to the calculation of
```

```
structure in the case of small buildings it is concise but effective'
```

[33]:
```
stopwords = english_stopwords if sample_lang == 'en' else spanish_stopwords if␣
↪sample_lang == 'es' else french_stopwords

sample_text = sample_text.split(' ')

for word in stopwords:
    sample_text = list(filter((word.lower()).__ne__, sample_text))

sample_text = ' '.join(sample_text)
sample_text
```

[33]:
```
'bought book wanted clear idea build straw greb particularly well studied makes
want build straw well explained effectively illustrated book certainly refer
future site author even provided essential key calculation structure case small
buildings concise effective'
```

[34]:
```
sample_text = SPACES_PATTERN.sub(' ', sample_text)
sample_text
```

[34]:
```
'bought book wanted clear idea build straw greb particularly well studied makes
want build straw well explained effectively illustrated book certainly refer
future site author even provided essential key calculation structure case small
buildings concise effective'
```

1.5 PreProcessing Interface

Now, once the research has been made and the preprocessing pipeline has been tested, we will just proceed with the Python implementation of an Interface in order to create a single preprocessing pipeline to preprocess all the data available in the previously loaded dataset.

[35]:
```
BASE_PATTERNS = (
    URL_PATTERN, HTML_PATTERN, PUNCTUATION_PATTERN,
    NUMBER_PATTERN, SPACES_PATTERN
)
```

Note: below you can see that there are some additional stopwords, since in the extra version of this Jupyter Notebook that can be found in `research/02 - Extra Data Preprocessing.ipynb` we have applied a TF-IDF Vectorizer over the preprocessed data generated below. So on, this Jupyter Notebook has been run twice, and the extra version just contains additional resources that are already implemented in this Notebook since this is the final version.

[36]:
```
STOPWORDS = english_stopwords + spanish_stopwords + french_stopwords

ADDITIONAL_STOPWORDS = [
    'much', 'despues', 'first', 'categoria', 'aqui', 'thumb', 'also', 'tres',␣
↪'asi',
    'three', 'one', 'still', 'aquella', 'like', 'aquel', 'mas', 'tal', 'tan',␣
↪'hacia',
    'went', 'two', 'new', 'even', 'would', 'tras', 'could', 'pues', 'without',␣
↪'category',
    'many', 'twoone', 'tambien', 'well', 'solo', 'dos'
]

STOPWORDS += ADDITIONAL_STOPWORDS
STOPWORDS = set(list(STOPWORDS))
```

[45]:
```
class CustomPreProcessor(object):
    """
    Custom PreProcessor

    Preprocesses the introduced raw text to transform it into clean text. This
    preprocessing pipe is regex based.

        >>> from apinlp.nlp.preprocessing import CustomPreProcessor
        >>> preprocessor = CustomPreProcessor()
        >>> print(preprocessor._preprocess("Visit us at https://www.ea.com/"))
        "visit us"
    """

    def __init__(self, strip_accents=True):
        self.strip_accents = strip_accents

        self.patterns = BASE_PATTERNS
        self.additional_patterns = (SPACES_PATTERN,)

        self.stopwords = STOPWORDS

    def _preprocess(self, text):
        """Cleans and applies a preprocessing layer to raw text"""
        text = text.replace('\t', ' ').replace('\n', ' ')

        if self.strip_accents:
            text = unidecode(text)

        for pattern in self.patterns:
            text = pattern.sub(' ', text)

        text = text.strip().lower()
        text = text.replace("'", " ")

        text = [word for word in text.split(' ') if len(word) > 2]
        for word in self.stopwords:
            text = list(filter((word.lower()).__ne__, text))

        text = ' '.join(text)

        for pattern in self.additional_patterns:
            text = pattern.sub(' ', text)

        return text
```

[46]:
```
preprocessor = CustomPreProcessor()
```

[48]:
```
preprocessor._preprocess(text="Visit us at https://www.ea.com/")
```

[48]:
```
'visit'
```

[40]:
```
preprocessor._preprocess(text="Visítanos en https://www.ea.com/")
```

[40]:
```
'visitanos'
```

[41]:
```
preprocessor._preprocess(text="Visitez-nous sur https://www.ea.com/")
```

[41]:
```
'visitez'
```

Future Note: additionally this feature may be included in a Python package so as to create a web service to test the models with real unseen data in order to ease its usage via an API instead of requiring to interact with the Jupyter Notebooks.

1.6 Preprocessed Data Overview

Finally, we will proceed with a simple overview on the preprocessed data, since we are applying our preprocessing interface named `CustomPreProcessor` to every single text in the dataset regardless the language. So on, this means that we are going to create a new column which will contain the preprocessed text, which we will be using later so as to vectorizer it and feed the model.

[42]:
```
%time data['preprocessed_text'] = data['text'].apply(preprocessor._preprocess)
```

```
CPU times: user 7min 21s, sys: 245 ms, total: 7min 21s
Wall time: 7min 21s
```

Now, once the preprocessed_text data has been created, we will just drop the original text column since it is not longer useful in this project, since as we already said, we will vectorize and feed the model with the preprocessed data.

[43]:
```
data.drop(columns=['text'], inplace=True)
data.head()
```

```
[43]:   lang    context                                  preprocessed_text
       0   en  wikipedia  watchmen twelve issue comic book limited serie…
       1   en  wikipedia  citigroup center formerly citicorp center tall…
       2   en  wikipedia  birth_place death_date death_place party conse…
       3   en  wikipedia  marbod maroboduus born died king marcomanni no…
       4   en  wikipedia  sylvester medal bronze medal awarded every yea…
```

Note: so as to work with the Jupyter Notebooks without repeating the same processes over and over we will just dump it into a JSON-Lines (.jsonl) file which will contain the `pandas.DataFrame` as a JSON object on each line of the file; but due to GitHub quotas and limits this file has been included in the .gitignore, so you will not be able to see it. Otherwise, just run this Jupyter Notebook in order to generate it.

```
[44]: data.to_json(path_or_buf='PreProcessedDocuments.jsonl', orient='records',␣
       ↪lines=True)
```