Diplomado Inteligencia Artificial y Ciencia de Datos: Métodos fundamentales, Progra Lógica, Ciencia de Computación e Ingeniería de Computación

Deep Learning: Modern Text Pre Processing. Vocabulary Challenge - Les Miserables.

Carolina Bernal Rodríguez
Fri March, 14th 2025

OVERVIEW

This challenge consisted on creating a vocabulary using the words from the book *Les Miserables* on the Spanish version downloaded on this link. To achieve that Python will be used for coding.

Do statistics, including:

1. How many words in original text (book)
2. How many different words in vocabulary
3. Print the 100 most frequent words in vocabulary
4. Print the 100 least frequent words in vocabulary

Code step by step

Convert from .pdf to .txt and .csv

For this step we need to create a function to transform the .pdf, Python has this module `from PyPDF2 import PdfReader` available , so it's easy to convert the text using `pdf_reader = PdfReader(pdf_file)` and `page = pdf_reader.pages[page_num]  text += page.extract_text()`

Standardize the text to not include upper characters, punctuation and accents

For this you'll need to use regular expressions (Regex) to modify the text like: `text_re = re.sub(r'[^\w\s]', '', text_re) text_re = re.sub(r"[!?-]", "", text_re)` among others.

Create the vocabulary logic

Here I decided to follow the logic both from the presentation of the slides available here https://bit.ly/3x2ifhu but also the idea to create a class that follows the logic from python. On this case the logic below to count and store the words for frequency and add a new one with frequency = 1 is similar to:

```
for s in iterable:
      if s in self.datos:
         self.datos[s] += 1
      else:
         self.datos[s] = 1
```

In this case that logic should be equivalent to:

```
def update(self, iterable):

""" Update the counter with the words find on the iterable """

      if isinstance(iterable, str):

          iterable = re.findall(r'\b\w+\b', iterable.lower()) # Divide word by
word ignoring punctuation

      for word in iterable:

           self.datos[word] = self.datos.get(word, 0) + 1 # if the word exists
then add 1, if not initialize counter on 1
```

Since:

1. If the word already exists in self.data, .get(word, 0) returns its current value and adds 1.
2. If the word doesn't exist, .get(word, 0) returns 0, so it is initialized with 1.
3. The word is stored in the dictionary with its new frequency.

```
# Count of words in text and the number of unique words

print("Count of total words on the text (First Part): ", total_w_cnt)

print("Count of unique words on the text (First Part): ", unique_w_cnt)
```

```
Count of total words on the text (First Part): 109629

Count of unique words on the text (First Part): 13439
```

Top 100 words (sample all words available on .ipynb file):

```
[('de', 5322), ('la', 3917), ('que', 3818), ('el', 3394),
('y',3122),('en', 2835), ('a', 2488), ('se', 1680), ('un', 1600), ('no',
1498), ('los', 1351), ('una', 1316), ('su', 1244), ('por', 936), ('las',
935), ('con', 924), ('habia', 858), ('del', 813), ('al', 756), ('es',
749), ('lo', 719), ('le', 667), ('era', 650), ('como', 572), ('mas', 513),
('para', 504), ('señor', 447), ….
```

Least frequent 100 words (sample all words available on .ipynb file):

```
[('charlesfrancoisbienvenu', 1), ('interesa', 1), ('exactos', 1), ('circulado',
1), ('ocupa', 1), ('reservandole', 1), ('consagrada', 1), ('galanterias', 1),
('sobrevino', 1), ('precipitaronse', 1), ('diezmadas', 1), ('perseguidas', 1),
('acosadas', 1), ('emigro', 1), ('tragicos', 1), ('espantosos', 1),
('emigrados', 1), ('germinar', 1), ('distracciones', 1), ('afectos', 1),
('lacerandole', 1), ('conmoverian', 1), ('hiriesen', 1),...
```

Save to parquet, on this step we also include it on the `class CounterWords:`

```
def save_to_parquet(self, filename="vocabulary.parquet"):

""" Save the dictionary on parquet file """

df = pd.DataFrame(self.datos.items(), columns=["word", "frequency"])

df.to_parquet(filename, engine="pyarrow", index=False)

print(f"File Saved on: {filename}")
```

Conclusions and Learnings

- ➔ Creating a vocabulary from scratch was useful to understand how to use regex to clean the data, also it was important to learn a logic step by step on how to include a new word to the vocabulary and its frequency on the text.
- ➔ Python has useful libraries to store a parquet file by using a pandas dataframe.
- ➔ As expected, articles and word connectors on the text were the ones with the highest frequency.
- ➔ Creating a Class was also insightful since all the functions we need to create can be described as methods of the class, Classes help us in bundling data and functionality together keeping the code clean and more readable. I decided to structure like that since Python has already collections https://docs.python.org/3/library/collections.html
- ➔ Will try the Addendum from the challenge to practice, but on this occasion this was not included since was not completed.