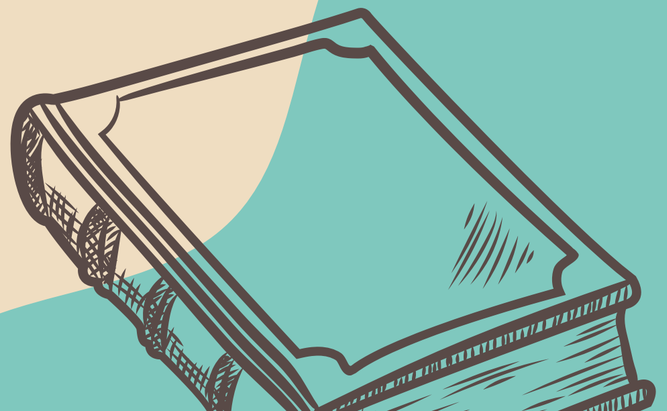
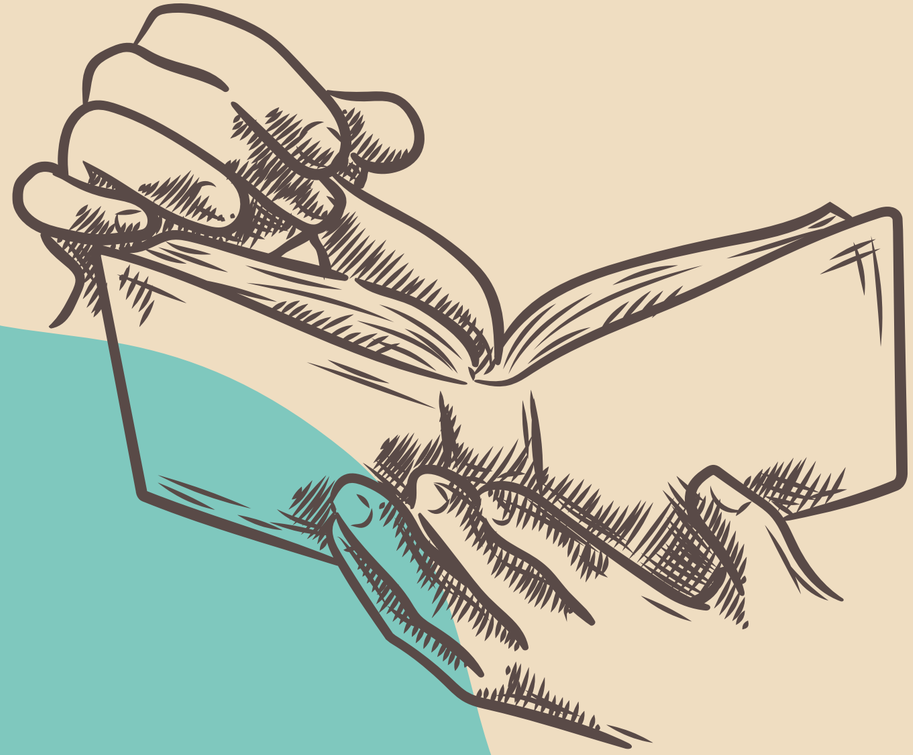
El Akinator literario

NLP: ENTREGA 1

Agustín Jerusalinsky, Natali Lilienthal
y Camila Borinsky

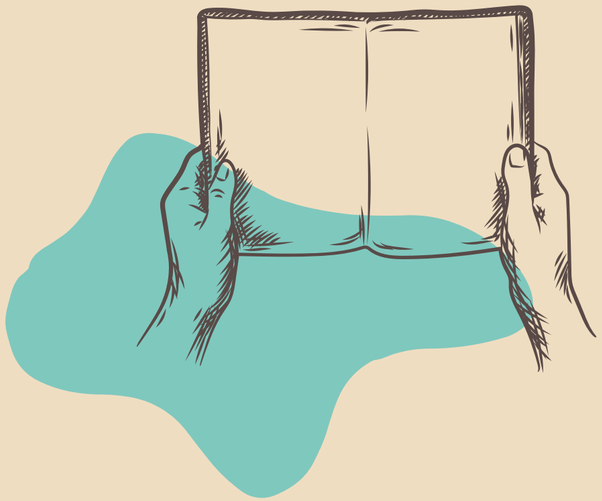


Contenidos



Motivación

Motivación



Resolver debates

Resolver disputas de autoría de obras anónimas, o cartas y publicaciones del ámbito forense.

Detección de plagio

Detectar plagio académico para promover integridad y producción original

Estudio comparativo

Mayor comprensión de estilos literarios. Posibilidad de realizar estudios comparativos.

Motivación

Identificar el autor de un fragmento de texto.

Queremos entrenar un modelo que pueda etiquetar textos según su autor.

Corpus

Parrafos de libros etiquetados con su autor

Entrenamiento

Entrenamiento con el corpus utilizando algúna técnica de NLP.

Modelo

Poder pasarle como entrada textos y que identifique el autor del mismo

Corpus

Corpus

1- Elegir autores

Top 100 Authors yesterday

1. Smollett, T. (Tobias) (9231)
2. Shakespeare, William (8828)
3. Montgomery, L. M. (Lucy Maud) (3790)
4. Alcott, Louisa May (3772)
5. Eliot, George (3743)
6. Forster, E. M. (Edward Morgan) (3720)
7. Melville, Herman (3639)
8. Dumas, Alexandre (3630)
9. Merrill, Frank T. (3372)
10. Rizal, José (3311)
11. Von Arnim, Elizabeth (3293)
12. Gaskell, Elizabeth Cleghorn (3264)
13. Brock, C. E. (Charles Edmund) (3115)
14. Fielding, Henry (2995)
15. Wagner, Richard (2836)
16. Dickens, Charles (2357)
17. Austen, Jane (2304)
18. Wilde, Oscar (1658)
19. Dostoyevsky, Fyodor (1620)
20. Twain, Mark (1590)

Top 100 Gutenberg authors

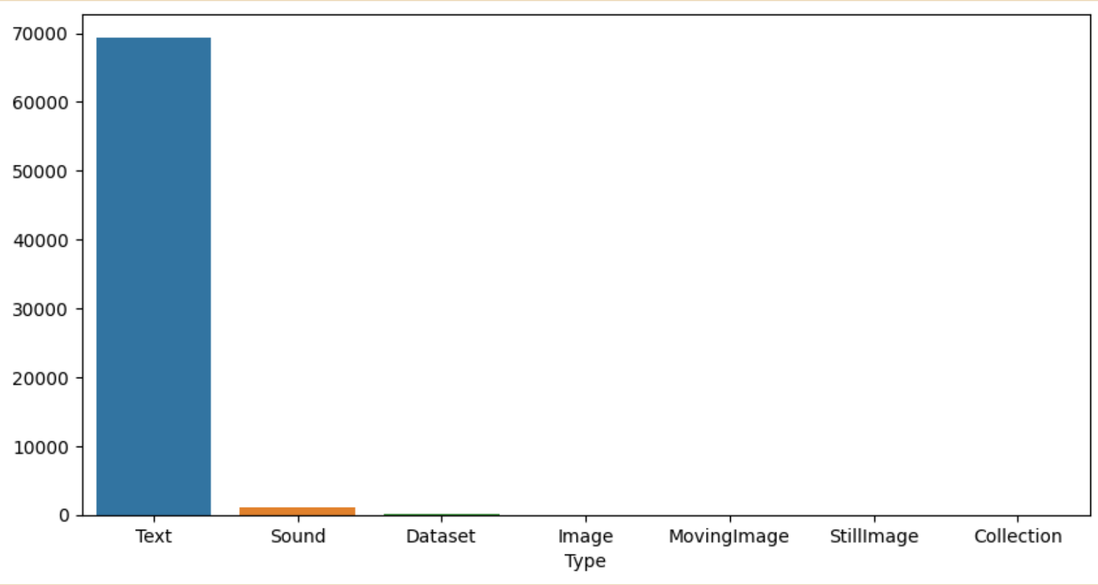
2- Buscar libros de los autores

- Limpieza de dataset completo de libros.
- Filtramos libros de los autores en el top 100.
- Obtenemos los ids de los libros filtrados
- Descargamos texto completo de estos libros usando **gutenbergpy**

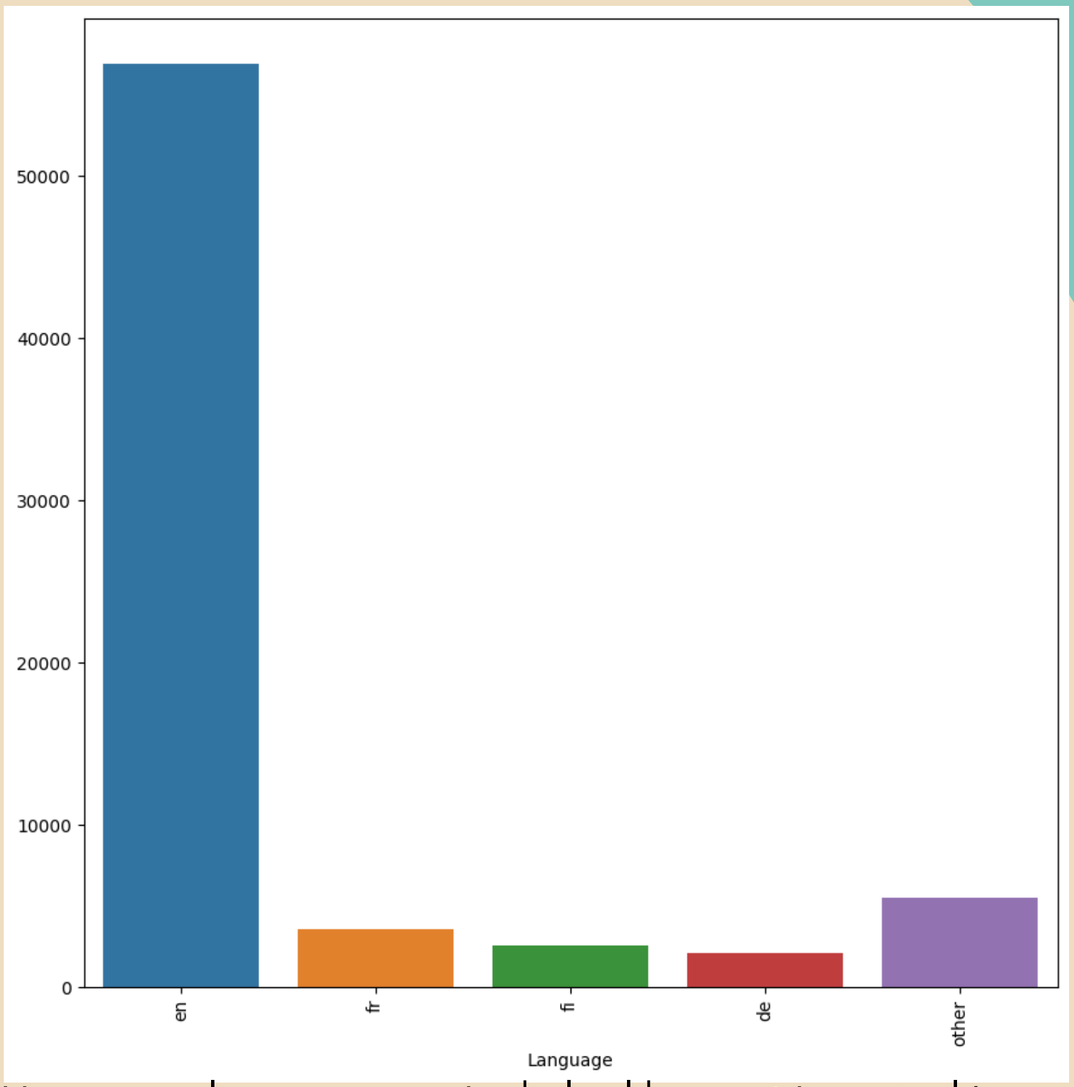
3- Dividir los libros en párrafos

- Separamos los libros en párrafos y contabilizamos los mismos.
- En principio tomamos como separador '\n\n'.
- Si en el futuro encontramos una mejor manera de separar, lo vamos a cambiar.

Corpus
Limpieza del dataset



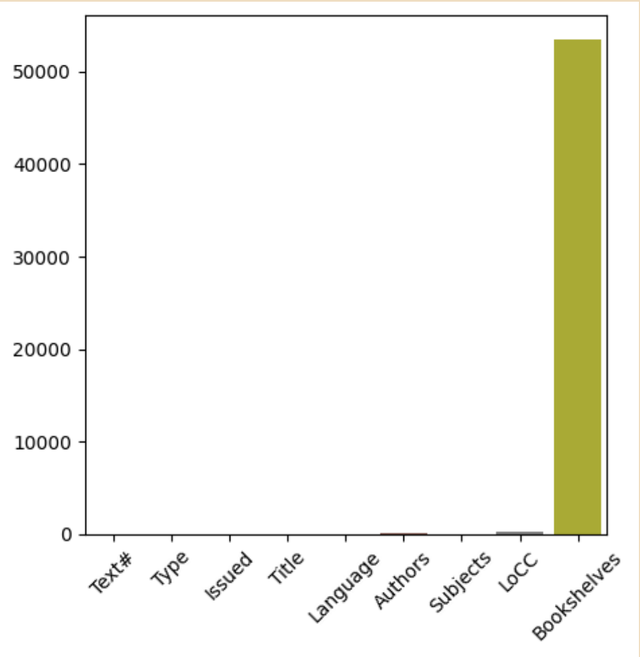
Vemos que la gran mayoría de los libros están en formato texto, nos quedamos solo con estos.



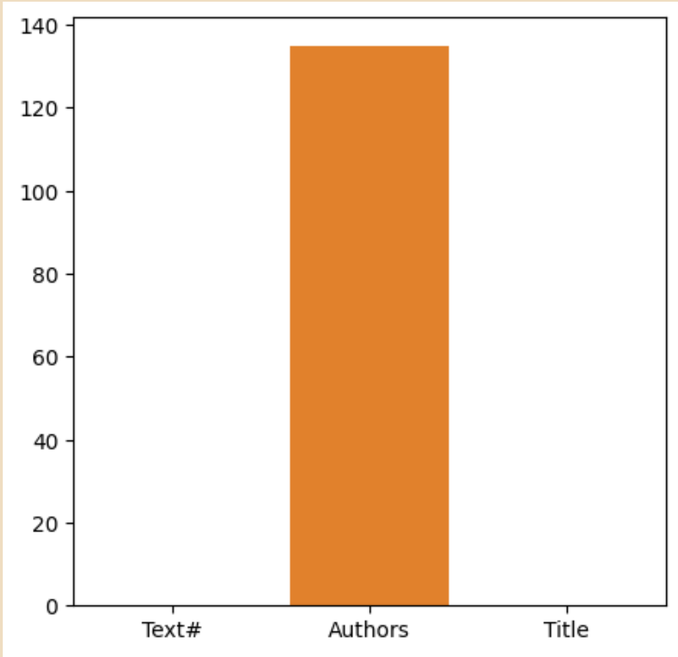
Vemos que la gran mayoría de los libros están en inglés, nos quedamos solo con estos.

Corpus

Datos faltantes



A muchos libros les falta el campo 'Bookshelves'



Analizando faltantes en las columnas relevantes

Corpus

```
df_metadata = pd.read_csv('gutenberg.csv')
#filter type text
df_metadata = df_metadata[df_metadata['Type'] == 'Text']
#get only Text Authors and Title
df_metadata = df_metadata[['Text#','Authors','Title']]
df_metadata = df_metadata.dropna().reset_index(drop=True)
#remove years
df_metadata["Authors"] = df_metadata["Authors"].str.split(',').str[:-1].str.join(',').str.strip()
df_metadata["Authors"]
```

Seleccionamos de todos los libros en **gutenberg.csv** los escritos por los autores en el top 100.

```
authors = pd.read_csv('authors.csv')
#find last '(' and remove everything after
authors['Author'] = authors['Author'].str.split('(').str[:-1].str.join('(').str.strip()
```

```
code_ids = df_metadata_filtered['Text#'].tolist()
books = [download_text(id) for id in code_ids]
```

Descargamos el texto de los libros a utilizar

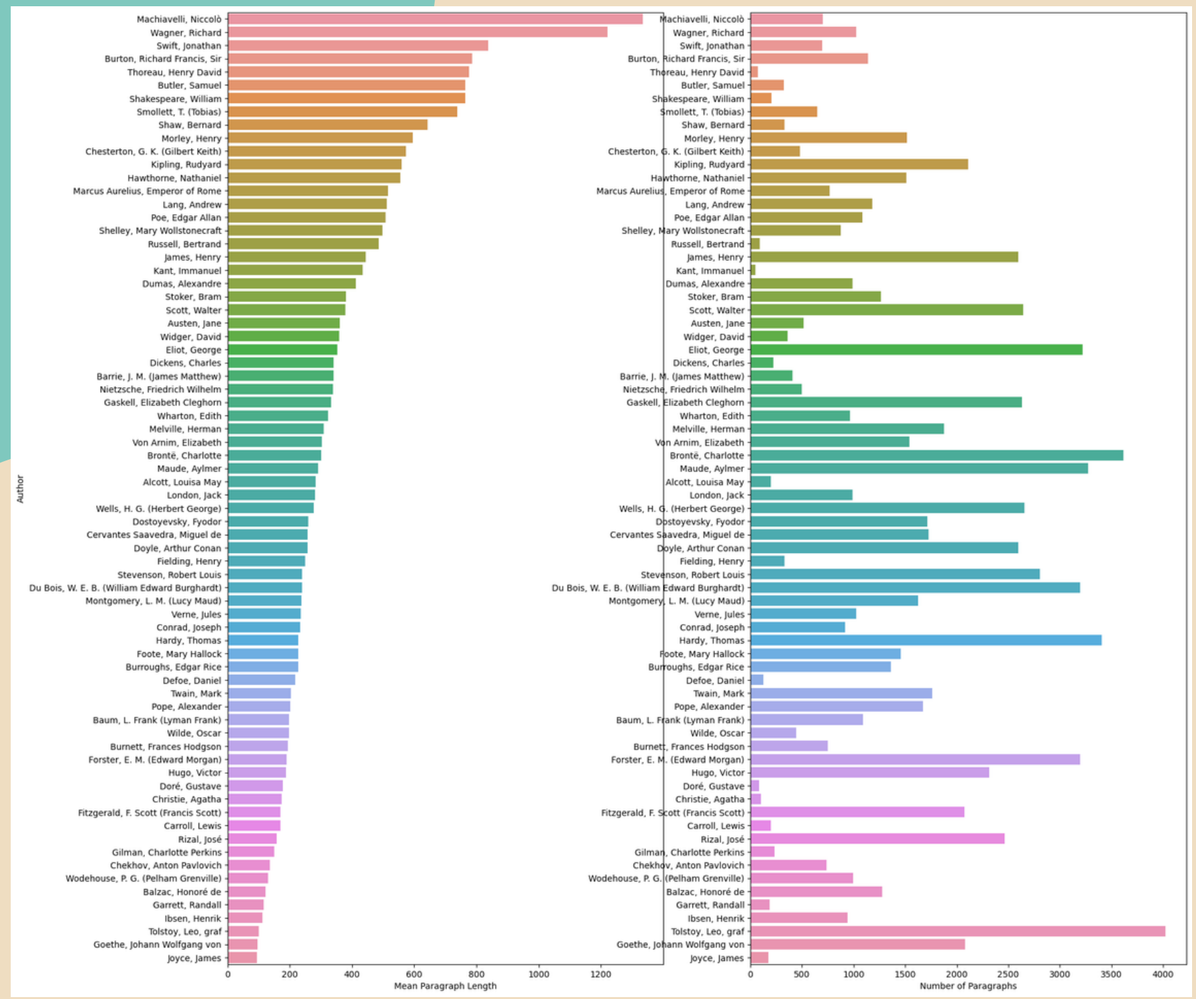
```
def download_text(id):
    try:
        text = gutenbergpy.textget.get_text_by_id(id)
        text = gutenbergpy.textget.strip_headers(text)
        text = text.decode('utf-8')
        return text
    except Exception as e:
        print("error downloading ID", id)
        raise e
```

Preprocesamiento

Preprocesamiento

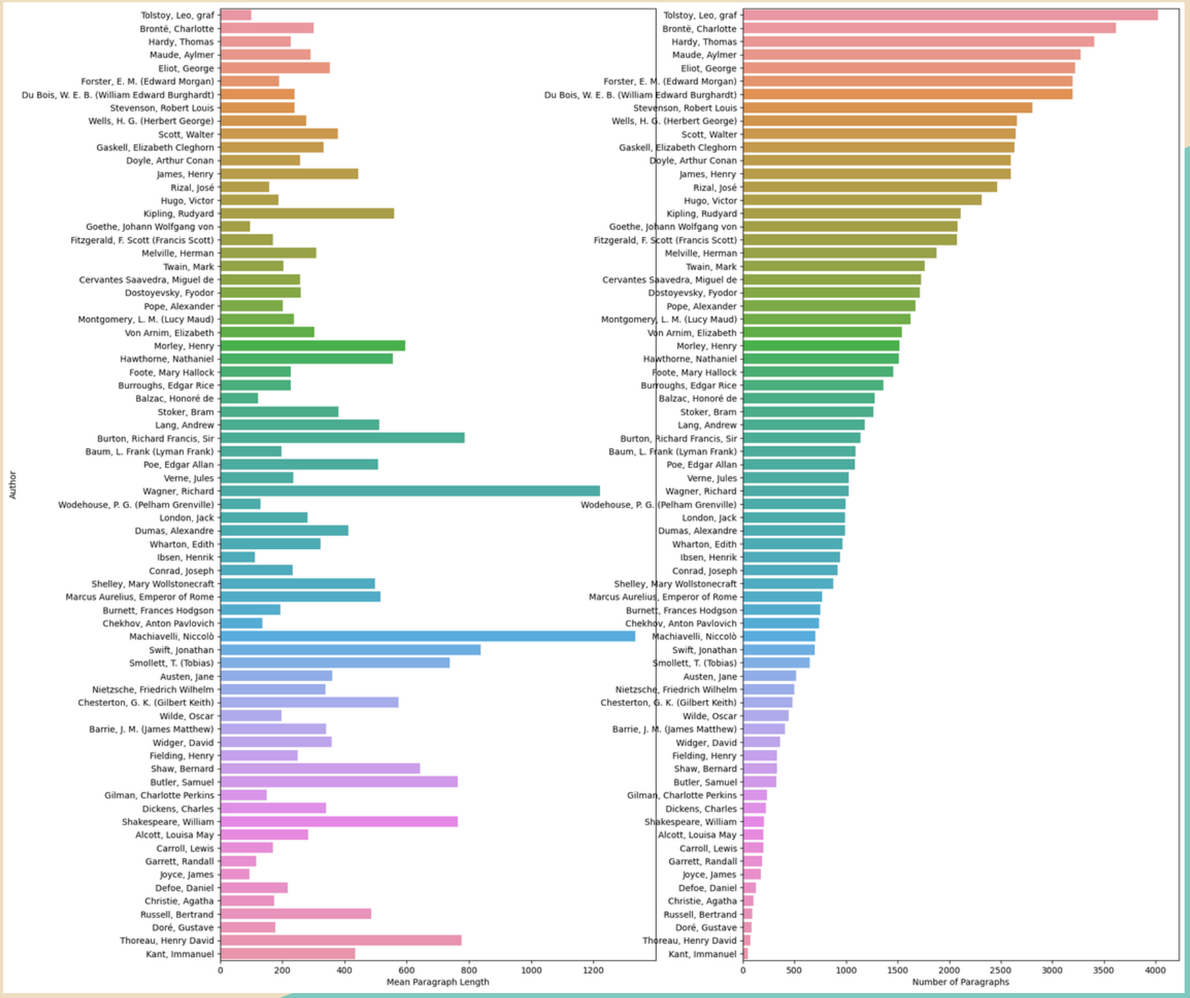
	Author	Mean Paragraph Length	Number of Paragraphs	Normalized Number of Paragraphs
42	Machiavelli, Niccolò	1333.285511	704	0.000749
66	Wagner, Richard	1221.575758	1023	0.000817
60	Swift, Jonathan	838.438672	693	0.001190
8	Burton, Richard Francis, Sir	785.711522	1137	0.001270
61	Thoreau, Henry David	776.541667	72	0.001284
9	Butler, Samuel	764.031153	321	0.001305
54	Shakespeare, William	763.127451	204	0.001307
57	Smollett, T. (Tobias)	738.868624	647	0.001350
55	Shaw, Bernard	641.879154	331	0.001553
47	Morley, Henry	595.815720	1514	0.001673
13	Chesterton, G. K. (Gilbert Keith)	572.872385	478	0.001740
39	Kipling, Rudyard	559.832781	2111	0.001780
33	Hawthorne, Nathaniel	554.821192	1510	0.001796
43	Marcus Aurelius, Emperor of Rome	516.503906	768	0.001929
40	Lang, Andrew	511.604061	1182	0.001947

Preprocesamiento



Mean Paragraph Length

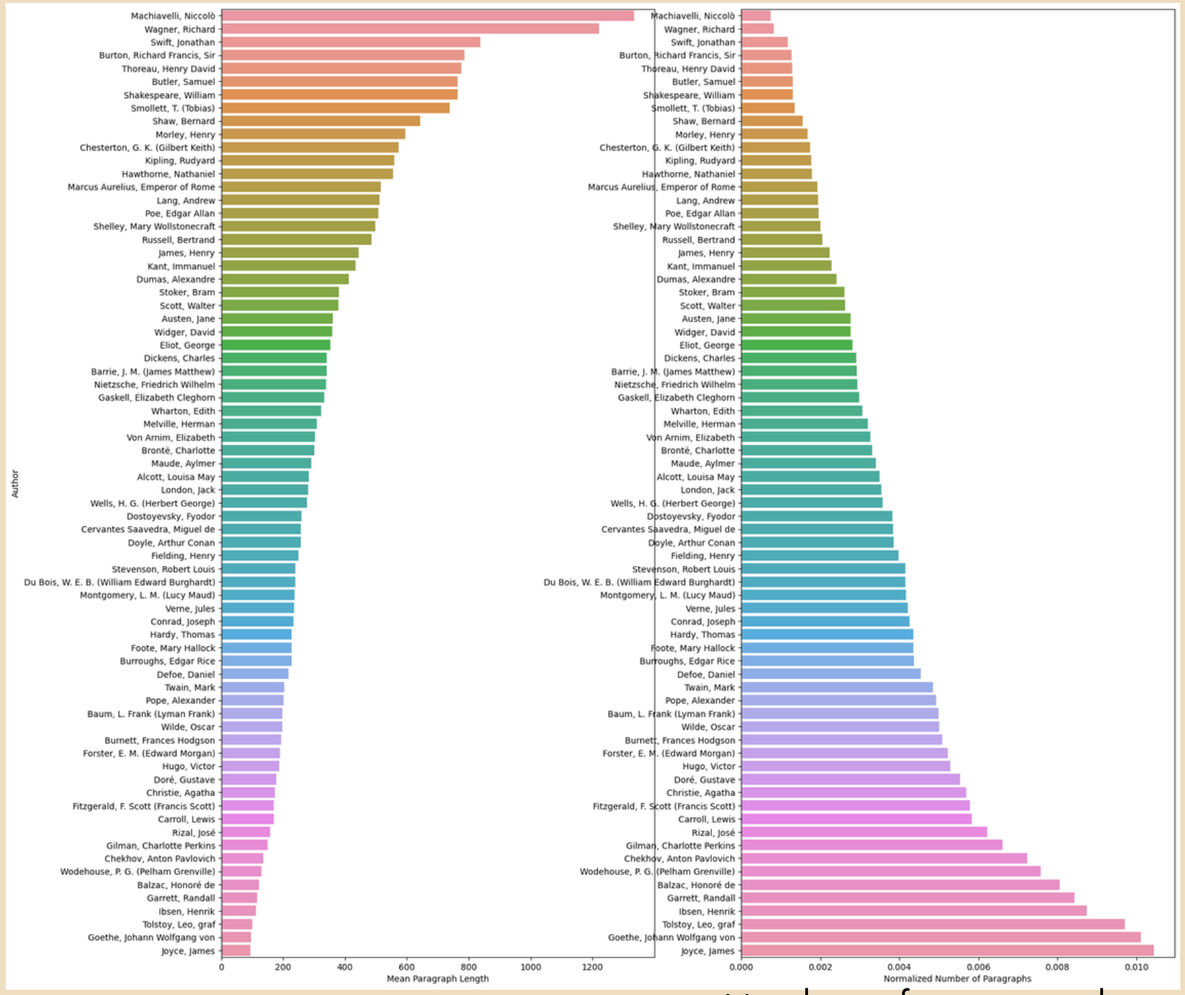
Number of paragraphs



Number of paragraphs

Mean Paragraph Length

Preprocesamiento



Mean Paragraph Length

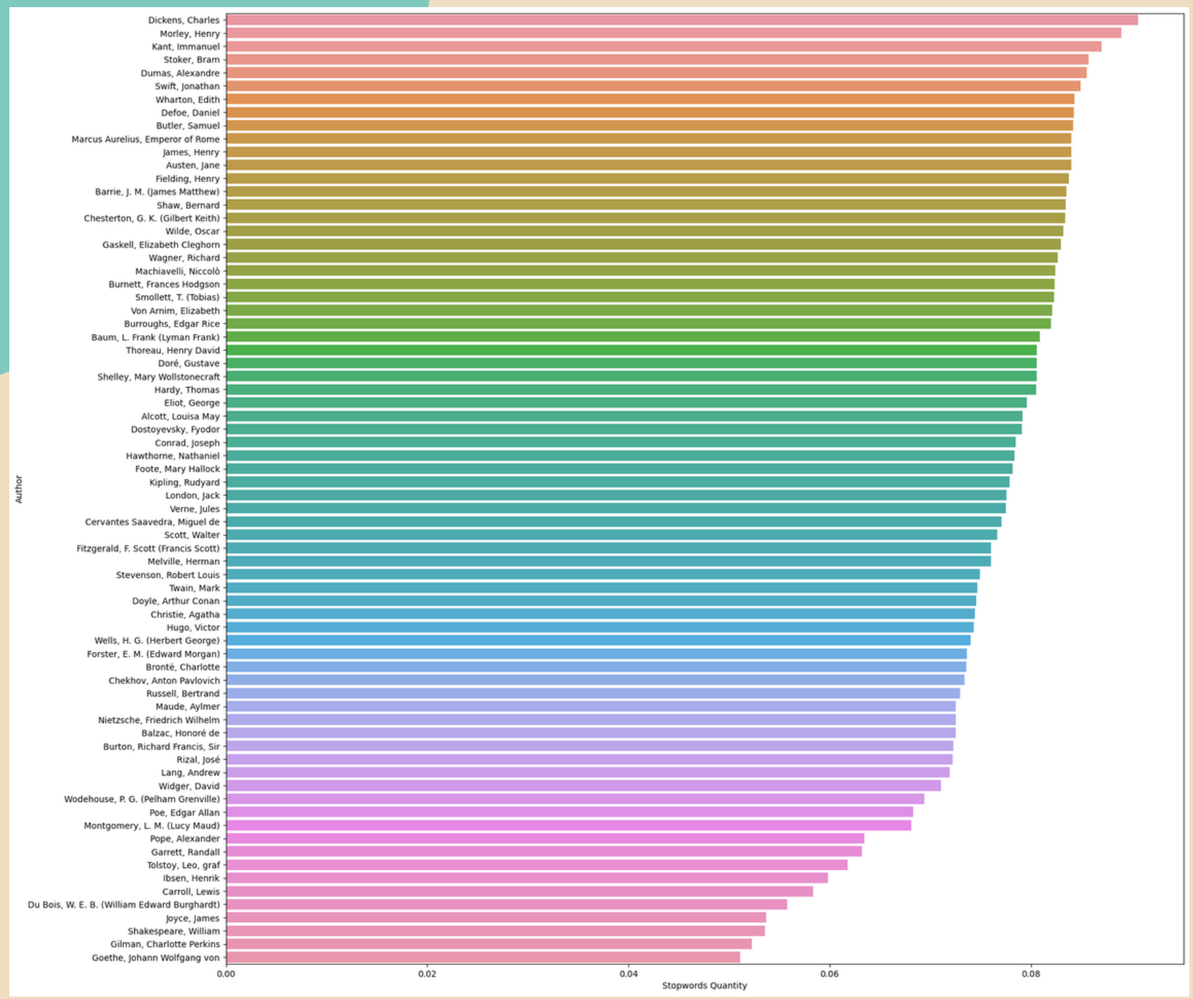
Number of paragraphs

Normalizamos el tamaño medio de párrafo según la longitud total del libro.

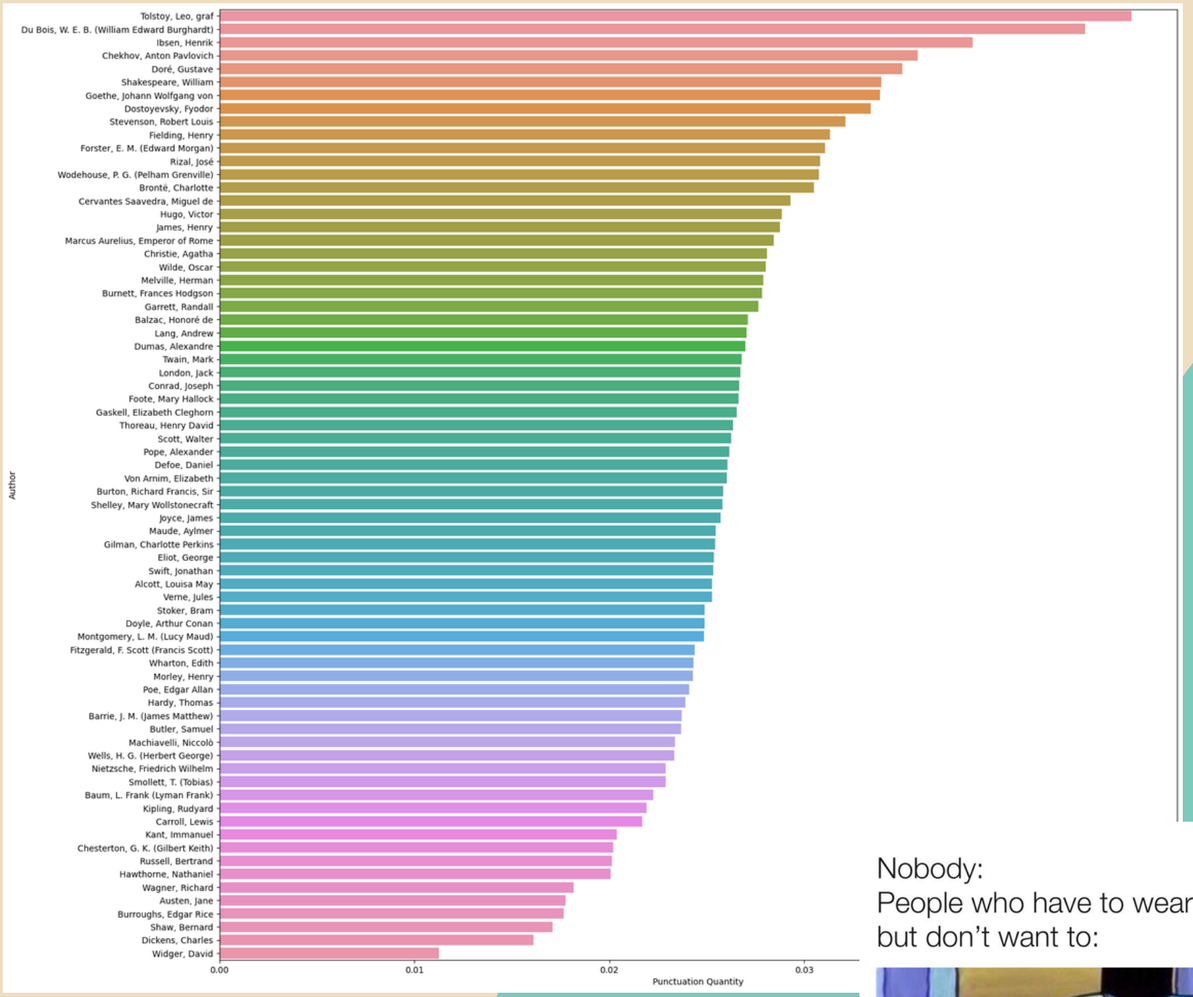
```
#for each author, get mean paragraph length
authors = []
mean_paragraph_lengths = []
num_paragraphs = []
normalized_num_paragraphs = []
for index, row in df_metadata.iterrows():
    authors.append(row['Authors'])
    paragraphs = get_book_paragraphs(row['Books'])
    num_paragraphs.append(len(paragraphs))
    mean_paragraph_lengths.append(np.mean([len(paragraph) for paragraph in paragraphs]))
    normalized_num_paragraphs.append(len(paragraphs)/len(row['Books']))
#sort by mean paragraph length
df_paragraphs = pd.DataFrame({'Author': authors, 'Mean Paragraph Length': mean_paragraph_
df_paragraphs
```

Preprocesamiento

Stopwords



Puntuación



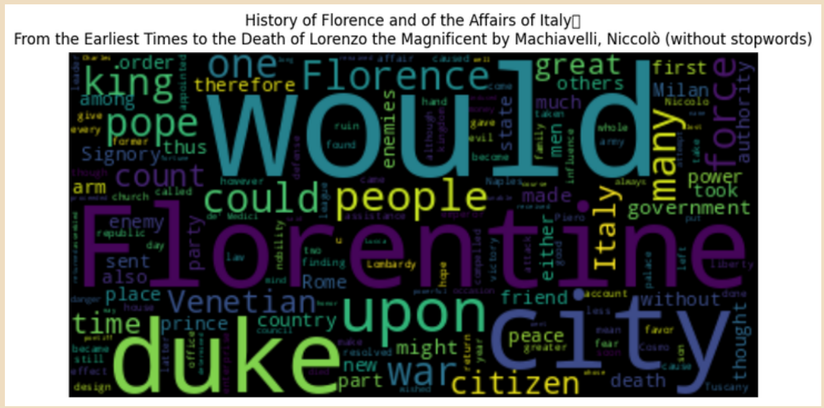
Nobody:
People who have to wear glasses but don't want to:

Preprocesamiento

Daniel Defoe

History of Florence and of the Affairs of Italy

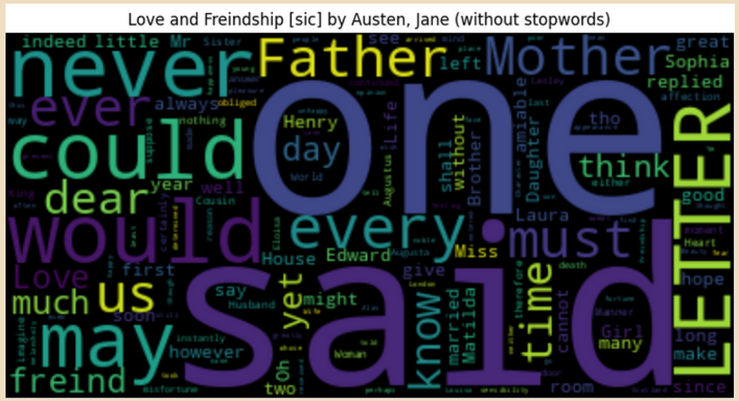
Eliminando stopwords:



Jane Austen

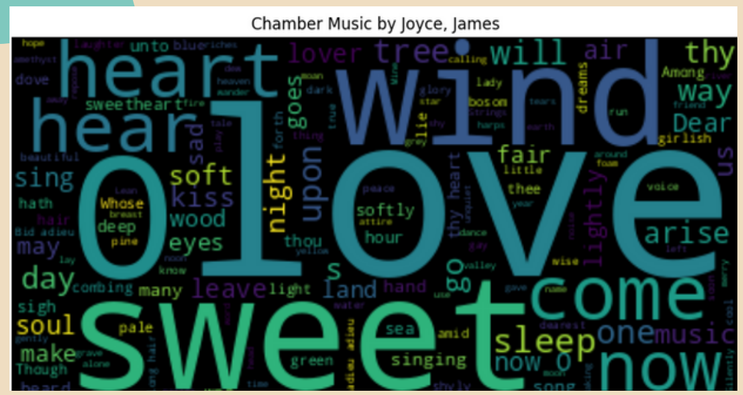
Love and Friendship

Eliminando stopwords:

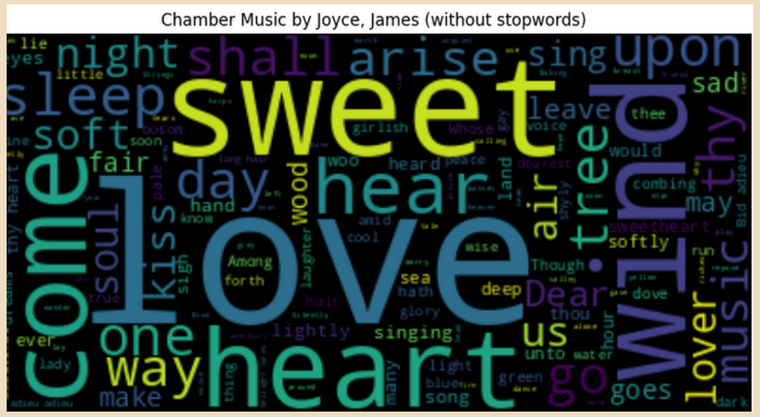


James Joyce

Chamber Music



Eliminando stopwords:



Preprocesamiento

Jane Austen

Love and Friendship

	TF-IDF
lesley	0.543093
freind	0.488784
lutterell	0.249823
eloisa	0.238961
freindship	0.195514
adeiu	0.195514
beleive	0.184652
scudamore	0.173790
musgrove	0.162928
janetta	0.141204
freinds	0.130342
philippa	0.130342
greville	0.130342
beautifull	0.086895
lesleys	0.076033
laurina	0.076033
wilhelminus	0.076033
disagreable	0.065171
uske	0.065171
dieing	0.054309
halton	0.054309
bridget	0.054309
beleived	0.054309
strephon	0.054309
enter	0.043447

Arthur Conan Doyle

The Return of Sherlock Holmes

	TF-IDF
lestrade	0.705289
milverton	0.279454
staunton	0.206264
soames	0.199610
woodley	0.172995
mcfarlane	0.166342
oldacre	0.166342
cubitt	0.159688
huxtable	0.126420
holdernesse	0.126420
norwood	0.126420
moriarty	0.113112
elsie	0.106459
moran	0.106459
hayes	0.106459
overton	0.099805
brackenstall	0.099805
cyclist	0.093151
charlington	0.093151
adair	0.093151
gilchrist	0.093151
beppo	0.079844
slaney	0.073190
saltire	0.066537
cyril	0.066537

Oscar Wilde

The Happy Prince, and Other Tales

	TF-IDF
impression_	0.588348
pyrotechnist	0.392232
pyrotechnic	0.294174
march	0.196116
daffodil	0.196116
pylotechnic	0.196116
february	0.196116
january	0.098058
tavistock	0.098058
gilly	0.098058
baalbec	0.098058
blacker_	0.098058
_sixth	0.098058
marjoram	0.098058
beryls	0.098058
ballantyne	0.098058
ibises	0.098058
_seventh	0.098058
_fifth	0.098058
nutt	0.098058
september	0.098058
_third	0.098058
ladysmock	0.098058
pygmies	0.098058
bulrush	0.098058

Próximos pasos

Próximos pasos

01 Modelo basado en características lingüísticas: longitud de las palabras, la frecuencia de palabras o el uso de ciertos patrones gramaticales.

Para conjuntos más pequeños. Limitaciones para relaciones semánticas más complejas

02 Modelo basado en BoW (Bag-of-Words): se crea un vocabulario a partir de todas las palabras presentes en el corpus y se representa cada texto como un vector de frecuencia de palabras.

Simple y rápido. No captura el orden de las palabras ni las relaciones semánticas entre ellas.

03 Modelo basado en TF-IDF (Term Frequency-Inverse Document Frequency). Se calcula la importancia de una palabra en un texto en relación con su frecuencia en todo el corpus.

TF-IDF puede ser mejor que BoW al dar importancia a las palabras menos comunes pero más distintivas.

Luego utilizar un clasificador como SVM, Naive Bayes , regresión logística o perceptron.

Bibliografía

Bibliografía

- Project Gutenberg Top 100 Authors Yesterday
- Gutenberg texts database

Gracias

NLP: ENTREGA 1

Agustín Jerusalinsky, Natali Lilienthal
y Camila Borinsky