Msc Natural Language Processing 2022-2023

UE 803 - Data Science

Project Report

Clémentine BLEUZE

Shehenaz HOSSAIN

Chun YANG

Contents

Introduction

This project is part of the UE 803 Data Science course by Claire GARDENT. It falls into two main parts: a comparison and classification task between two categories of texts, and a comparison of the behaviour of two linguistic Python libraries. The full subject will be available as a PDF on the GitHub repository of the project[1], alongside the code, present report, and other useful resources.

1 Part I: Binary classification of texts

In this first part of the project, we are interested in:

1) Collecting texts that belong to two distinct semantic categories
2) Analyzing, for each category of texts, their vocabulary, sentences, tokens, the Named Entities (NE) they contain, their Part-of-Speech (POS) distribution, etc.
3) Training two classifiers to label texts with their category, and evaluate their performance on a test set

1.1 Data Collection

1.1.1 Categories choice

Before starting to collect texts, we naturally had to choose two distinct semantic categories that we wanted to study. As asked for the task, we focused on Wikipedia biographies of famous people, choosing **USA Presidents**[2] and **Women Scientists of the** 21^{st} **century**[3]. Indeed, the Wikipedia pages for both of these categories are entry points to plenty of Wikipedia biographies, which means that they contain enough data for our task. In Figure 2[4], we see that all US Presidents' biographies are accessible via a chronological table, whereas dozens of Women Scientists' biographies are listed according to their nationality. Additionnally, we assumed that the difference in nature between the fields of USA Presidency and Contemporary Science makes these categories specific enough to control their own vocabulary, and thus makes them differentiable.

1.1.2 Web Scraping

The details of the text extraction process are presented and commented in the corresponding Jupyter Notebook. Using libraries *Requests* and *BeautifulSoup*, we first retrieved, for each category, the list of URLs of the target biographies. Then, we explored each of the biographies and extracted the raw text content. We had to decide what to extract precisely, as Wikipedia articles generally contain a lot of information of a diverse nature (text, URLs, pictures, etc.). In order to retrieve text only, with a focus on the meaningful content, we decided to extract only the content of paragraph tags (in HTML).

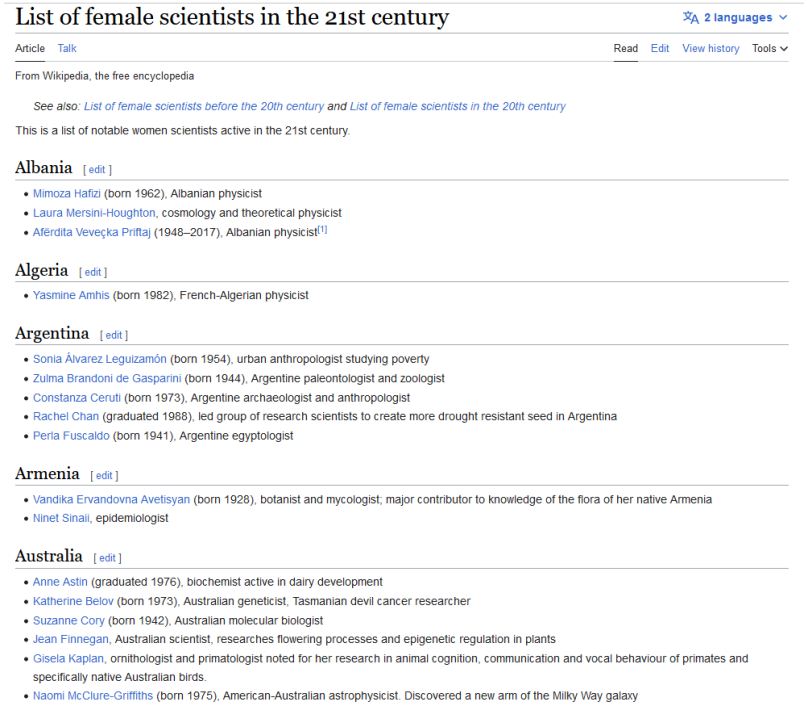
[1] `https://github.com/ClementineBleuze/M1-NLP-DataScience-Project`
[2] `https://en.wikipedia.org/wiki/List_of_presidents_of_the_United_States`
[3] `https://en.wikipedia.org/wiki/List_of_female_scientists_in_the_21st_century`
[4] Date of consulation for the screenshots: May 2, 2023.

44		**Barack Obama** (b. 1961) [73]	January 20, 2009 – January 20, 2017		Democratic	2008 / 2012	Joe Biden
45		**Donald Trump** (b. 1946) [74]	January 20, 2017 – January 20, 2021		Republican	2016	Mike Pence
46		**Joe Biden** (b. 1942) [6]	January 20, 2021 – *Incumbent*		Democratic	2020	Kamala Harris

See also

- Acting President of the United States
- Founding Fathers of the United States
- President of the Continental Congress

(a) Extract from the Wikipedia page of USA Presidents



List of female scientists in the 21st century

2 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

See also: List of female scientists before the 20th century and List of female scientists in the 20th century

This is a list of notable women scientists active in the 21st century.

Albania [edit]

- Mimoza Hafizi (born 1962), Albanian physicist
- Laura Mersini-Houghton, cosmology and theoretical physicist
- Afërdita Veveçka Priftaj (1948–2017), Albanian physicist[1]

Algeria [edit]

- Yasmine Amhis (born 1982), French-Algerian physicist

Argentina [edit]

- Sonia Álvarez Leguizamón (born 1954), urban anthropologist studying poverty
- Zulma Brandoni de Gasparini (born 1944), Argentine paleontologist and zoologist
- Constanza Ceruti (born 1973), Argentine archaeologist and anthropologist
- Rachel Chan (graduated 1988), led group of research scientists to create more drought resistant seed in Argentina
- Perla Fuscaldo (born 1941), Argentine egyptologist

Armenia [edit]

- Vandika Ervandovna Avetisyan (born 1928), botanist and mycologist; major contributor to knowledge of the flora of her native Armenia
- Ninet Sinaii, epidemiologist

Australia [edit]

- Anne Astin (graduated 1976), biochemist active in dairy development
- Katherine Belov (born 1973), Australian geneticist, Tasmanian devil cancer researcher
- Suzanne Cory (born 1942), Australian molecular biologist
- Jean Finnegan, Australian scientist, researches flowering processes and epigenetic regulation in plants
- Gisela Kaplan, ornithologist and primatologist noted for her research in animal cognition, communication and vocal behaviour of primates and specifically native Australian birds.
- Naomi McClure-Griffiths (born 1975), American-Australian astrophysicist. Discovered a new arm of the Milky Way galaxy

(b) Extract from the Wikipedia page of Women Scientists in the 21st century

Figure 2: Overview of the entry points for the two categories

1.1.3 Data Storing

For each scraped biography, its text content was stored in a textfile named after the article's title. Parallel to this, the content was also added to a global *Pandas DataFrame* with columns *title, content* and *category*. We can see an extract from this *DataFrame* in Figure 3.

	title	content	category
40	George H. W. Bush	\nGeorge Herbert Walker Bush[a] (June 12, 1924...	US_Presidents
22	Rutherford B. Hayes	\n\nRutherford Birchard Hayes (/ˈrʌðərfərd/; O...	US_Presidents
55	Alice K. Jacobs	Alice K. Jacobs is a professor at the Boston U...	Women_Scientists
72	Carme Torras	Carme Torras Genís (born 4 July 1956)[1] is a ...	Women_Scientists
0	George W. Bush	\nGeorge Walker Bush (born July 6, 1946) is an...	US_Presidents
...	...	...	...
20	Grover Cleveland	\nStephen Grover Cleveland (March 18, 1837 – J...	US_Presidents
60	Sara Gill	Sara Gill (Urdu: سارہ گِل) is a Pakistani phys...	Women_Scientists
71	Lydia Kavraki	Lydia E. Kavraki (Greek: Λύδια Καβράκη) is a G...	Women_Scientists
14	Martin Van Buren	\nMartin Van Buren (/væn ˈbjʊərən/ van BYURE-ə...	US_Presidents
51	Sandra Schmid	Sandra Louise Schmid (born March 7, 1958, in V...	Women_Scientists

Figure 3: Overview of the extracted data (after shuffling)

1.1.4 About the size of our two categories

When performing the scraping, we observed a large imbalance between the number of articles for our two categories: 46 articles for US Presidents, against 332 for Women Scientists. This is the reason why we also created a second *DataFrame* of balanced data by randomly selecting 46 articles of the category Women Scientists. However, we then noticed that, despite being numerous, Women Scientists articles are way shorter than US Presidents articles, which already makes up for the initial imbalance. This will be further confirmed by the comparison of the number of sentences per article and per category, as well as the vocabulary size in the following step of the project (see Section **??**).

1.2 Data Analysis

1.2.1 Preprocessing

We start by language preprocessing, where the goal is to clean the original content of the article and then break it down into words that a computer program can interpret. The preprocessing steps are as follows :

1) Remove unwanted formatting, e.g. quotation and excess white space from the text.

2) Sentence segmentation: splitting documents into sentences. This step is to prevent ambiguity when processing whole sentences. For example, in a word like *U.S* the symbol "." is not the end of a sentence, but it should be considered as a whole.

3) Word tokenization: split sentences into words.

4) Word normalization: converts words to normalized form, i.e lowercase every word in the text. For example U.S → u.s, since U.S and u.s belong to the same word.

5) Remove stop words : remove unnecessary words, which aims to reduce the size of vocabulary and map words with the same meaning to the same word. For example *the, a, of, for* etc. which are meaningless words for discovering different categories of text.

1.2.2 Vocabulary analysis

In NLP, vocabulary represents the set of all unique tokens that appear in a dataset. The size of vocabulary in each category are shown in 1, as well as the total number of token occurrences (count columns). We also present the count and size of what we could call a *Filtered Vocabulary*, in which punctuation and stopwords occurrences have been removed.

Category	**Vocab count**	**Vocab size**	**F_Vocab count**	**F_Vocab size**
USA Presidents	676 131	25 155	330 156	24 825
Women Scientists	197 360	16 756	98 823	16 443

Table 1: Vocabulary statistics

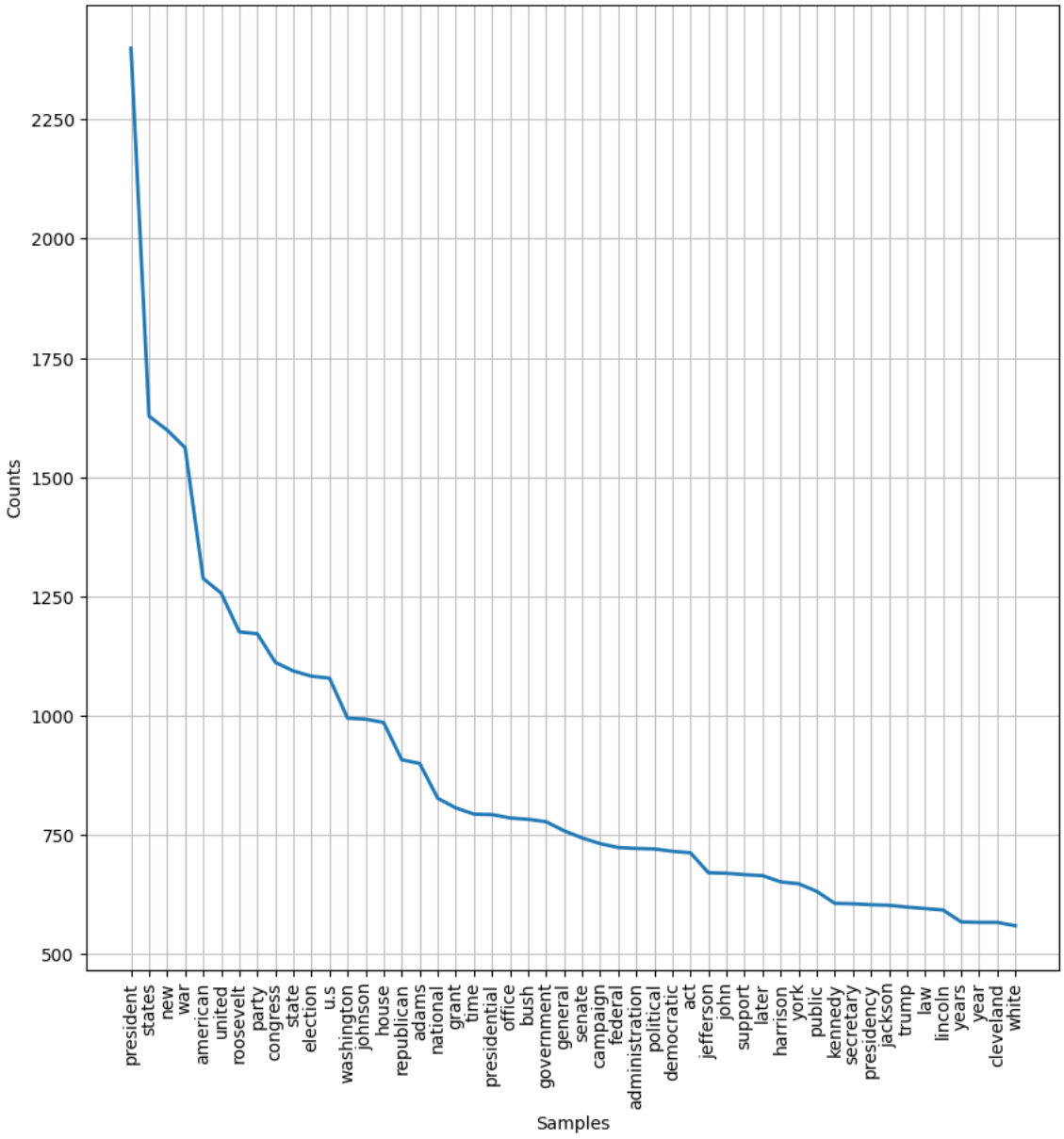
we also extracted the 50 most common words and display word clouds in the two categories of articles **USA Presidents** and **Women Scientists** respectively.

In Figure 4a, we see the 50 words with the highest frequency in **USA Presidents**: the word *president* has the highest frequency, more than 2250, and the 50^{th} word *white* appears more than 500 times. In Figure 5a, we see more clearly the frequent words in **USA Presidents**: *president, state, united, US, roosevelt, congress, war, johnson, election, washigton* etc.

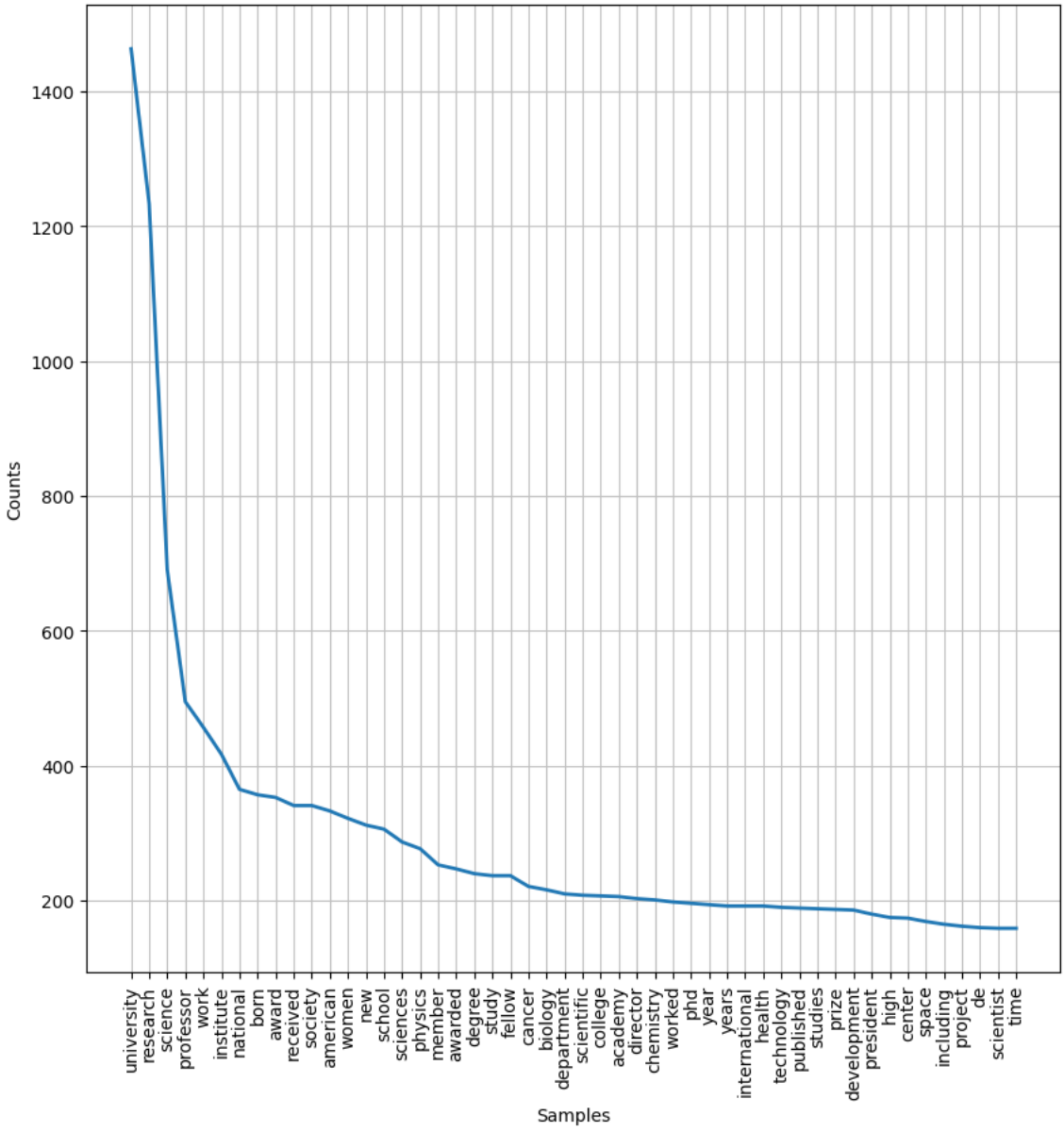
Now looking at **Women Scientists** (see Figure4b): the word *university* has the highest number of counts more than 1400, and the 50^{th} word *time* counted *less than 200* . In Figure 5b, the words with the highest frequency in **Women Scientists** are: *university, research, science, work, award, institute, received, professor, department, women* etc.

These results seem intuitively coherent with what we would expect as a specific vocabulary to both these categories. However, we can notice that the most frequent word in **USA Presidents** appears a lot more than the most frequent word in **Women Scientists**. In order to explain this phenomenon, we can formulate two hypotheses:

- Maybe this is due to the fact that the category **Women Scientists** has a smaller number of token occurrences than the category **USA Presidents** (this is verified in Table 1).
- Maybe *president* is more specific to the category **USA Presidents** than *university* is to **Women Scientists**, which also seems reasonable.

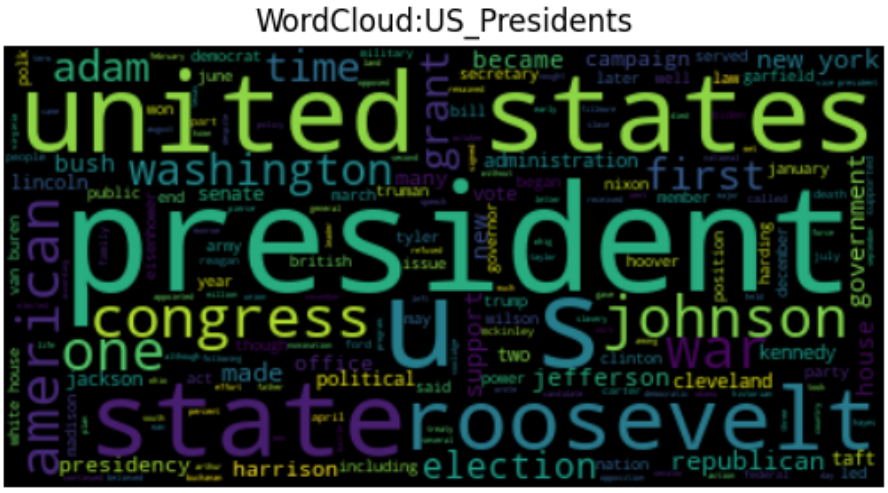


(a) USA Presidents



(b) Women Scientists

Figure 4: 50 most common words for both categories (with counts)



(a) WordCloud:US Presidents

(b) WordCloud:Women Scientists

Figure 5: 50 most common words for both categories (wordclouds)

1.2.3 Sentences and tokens analysis

For this part, we gave some statistics about the sentences and tokens for each category's articles. We focus on the following points:

1) How many sentences are there for each category ?

2) What are the minimum, maximum, and average number of sentences per article for each category ?

3) What are the minimum, maximum, and average number of tokens per sentence for each category ?

The number of sentence for **US Presidents** and **Women Scientists** are shown in Table 2. There is a huge difference in the number of sentences between these two categories, **USA Presidents** counting more than 3 times more sentences than **Women Scientists**.

Category	Number of sentences
USA Presidents	23 506
Women Scientists	6 951

Table 2: Sentence counts for both categories

Concerning sentences count per article for each category, the results are showed in Figure 6. We see that articles from the category **USA Presidents** have an average number of approximately 558 sentences, with the shortest article containing 328 sentences, and the longest one 900 sentences. We have the confirmation that articles from the category **Women Scientists** are way shorter: they contain on average 24 sentences, with a shortest article containing only 5 sentences, and the longest containing 214 sentences. A visualization of these results in the form of a histogram is also presented.

category	count	mean	std	min	25%	50%	75%	max
US_Presidents	46.0	558.500000	144.413103	328.0	451.5	531.0	646.0	900.0
Women_Scientists	332.0	24.521084	19.987976	5.0	14.0	20.0	28.0	214.0

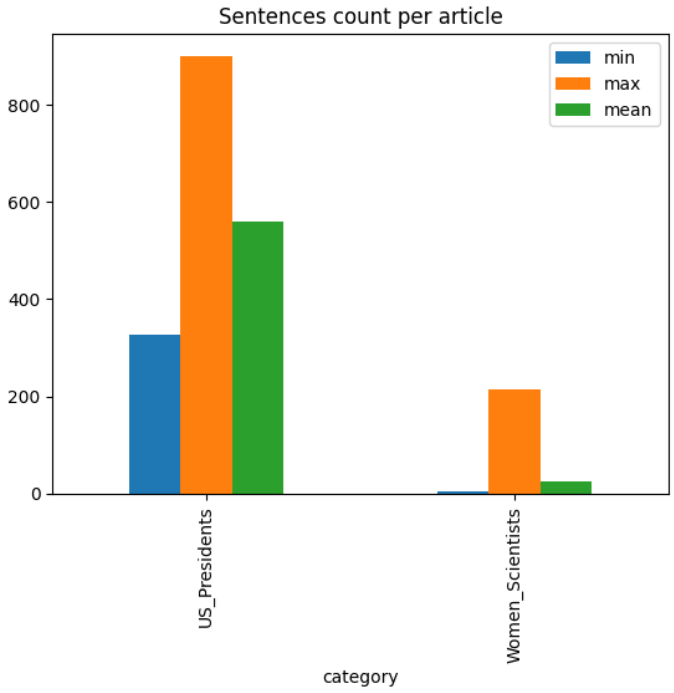


Figure 6: Statistics on sentences per article

Finally, Figure7 presents some statistics about the number of tokens per sentence for both categories. Sentences in **USA Presidents** are very balanced, with on average 26 tokens per sentence (min: 23, max: 29). The count of tokens per sentence varies more for the category **Women Scientist**, with an average of 23 tokens per sentence (min: 12, max:37).

category	count	mean	std	min	25%	50%	75%	max
US_Presidents	46.0	26.383907	1.297963	23.341493	25.394029	26.471619	27.206366	28.912351
Women_Scientists	332.0	23.863055	4.288405	12.406542	20.987500	23.651116	26.545151	37.750000

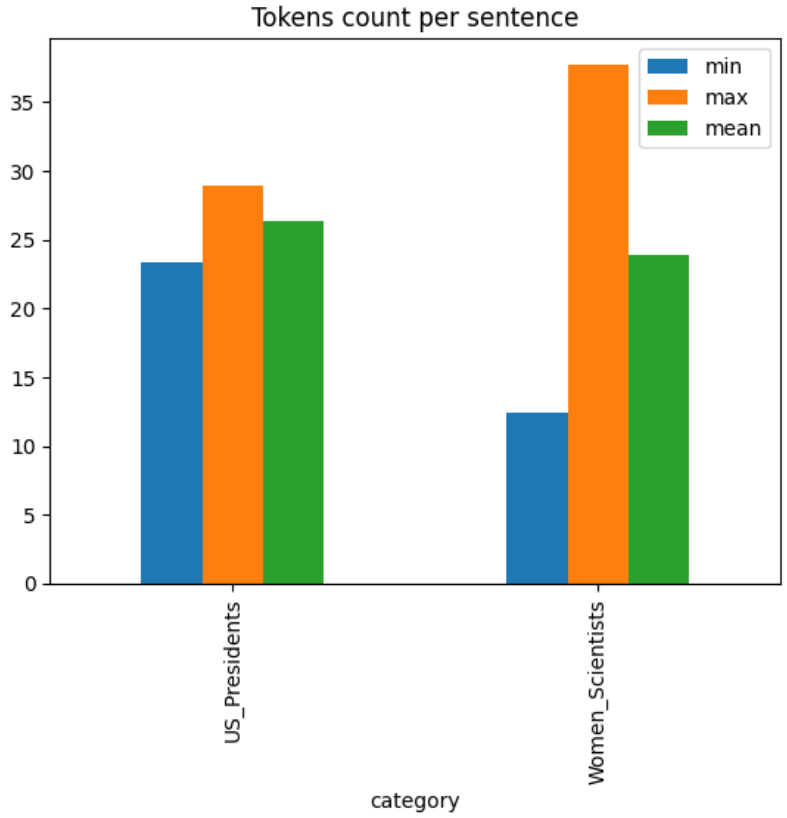


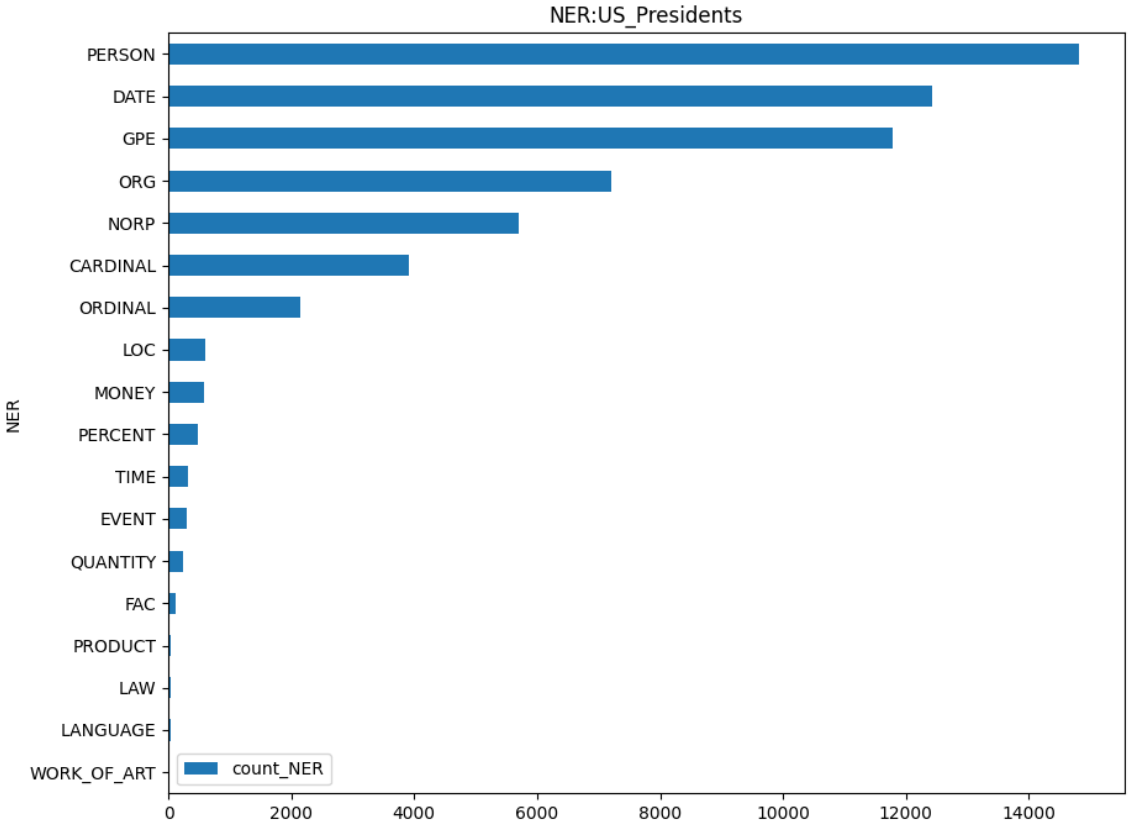
Figure 7: Statistics on tokens per sentence

1.2.4 Named Entity Recognition (NER)

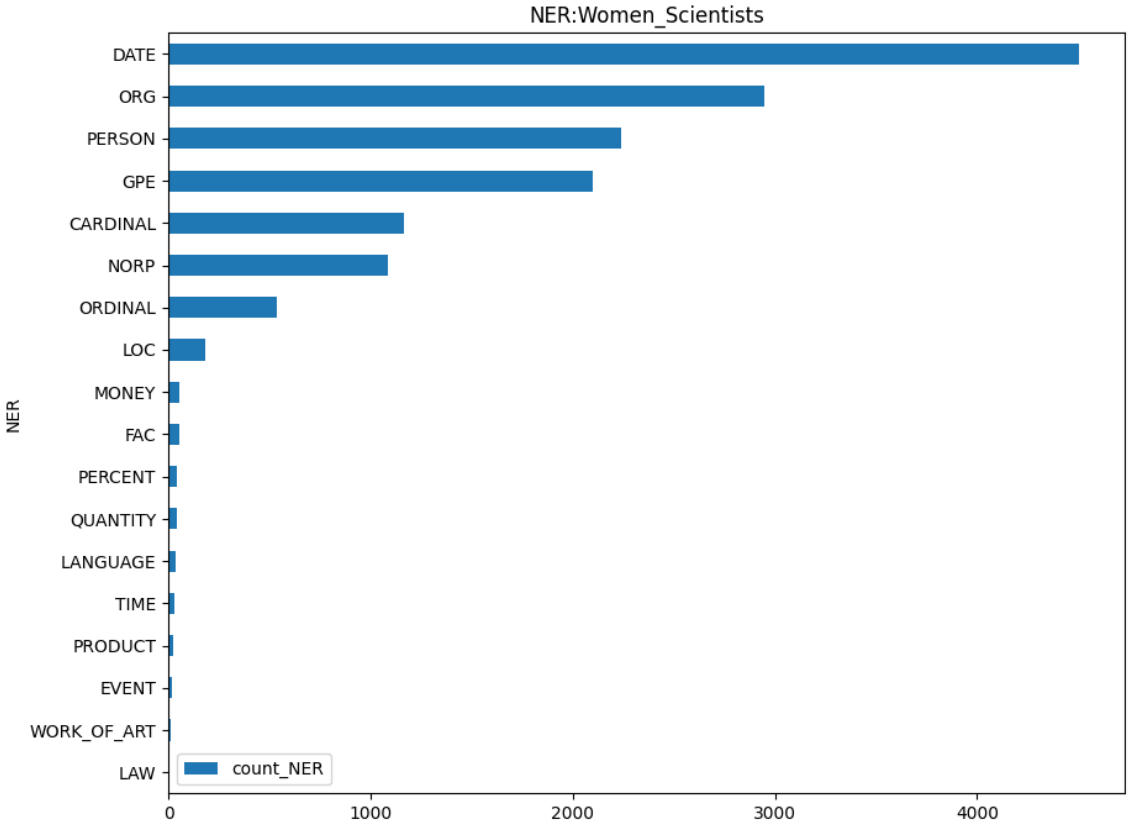
In this section, we focus on Named Entity Recognition (NER), which refers to identifying entities with specific meaning in text. It involves detecting and classifying important information in text, such as names of people, places, institutions, proper nouns, etc. The number of NER for two categories are showns in Table 3. The 20 most frequent NERs for two categories are shown in Figure 8. With the help of the library we can have an explanation for each NER, e.g., **TIME** which means *Times smaller than a day.*

Category	Number of NER
USA Presidents	60 790
Women Scientists	15 082

Table 3: NE Counts for both categories



(a) Most frequent NER for USA Presidents



(b) Most frequent NE for Women Scientists

Figure 8: 20 most frequent NER of two categories

From these two figures, we can observe that the most frequent NER types are very similar: PERSON, DATE, GPE, ORG, NORP, CARDINAL and ORDINAL make up for the majority of occurrences. It seems that **USA Presidents** articles are more focused on PERSON, and **Women Scientists** articles are more focused on DATE and ORG.

1.2.5 Part of Speech (PoS)

We also compute the PoS and count the number of word belonging to different PoS for the two categories of text. Some examples are shown in Figure 9.

	noun	count_noun	verb	count_verb	adjs	count_adj
0	president	1706	had	817.0	first	1056.0
1	war	1234	became	674.0	american	1038.0
2	election	1074	made	648.0	presidential	787.0
3	party	881	won	506.0	other	736.0
4	adams	862	including	480.0	new	724.0

(a) PoS US Presidents

	noun	count_noun	verb	count_verb	adjs	count_adj
0	research	1120	born	357.0	first	374.0
1	science	559	received	341.0	scientific	208.0
2	professor	448	awarded	247.0	new	190.0
3	work	417	worked	198.0	american	187.0
4	women	321	became	197.0	high	168.0

(b) PoS Women Scientists

Figure 9: PoS of two categories

1.3 Classification

Classification usually means the process related to the categorisation. Our current task is to classify texts between two categories namely: **US Presidents** and **Women Scientists**. This problem can be simplified as the task of binary classification.

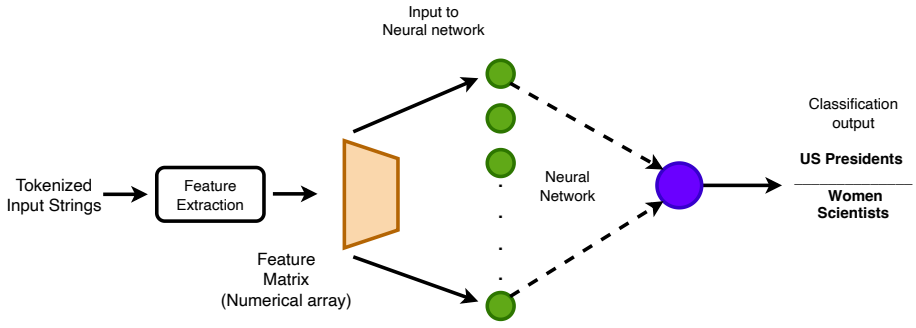


Figure 10: The overview of the machine learning based classification model

To perform this classification we test two machine learning models to aid in performing the classification task on an unknown text input. A quick overview of the operation can be seen in Figure 10. An input string is taken and thereafter cleaned and tokenized.

After tokenization is performed, feature extraction is carried out. The feature extraction process can be understood as way of translating the text input into a numerical vector. The numerical vector is then fed as input to the neural network. The neural network based upon the input carries out a certain set of activation functions, which are then aggregated in an appropriate manner to give the final classification output. These steps will be treated in detail in the upcoming sub sections. We start by describing feature extraction, followed by discussing the machine learning model architecture. Thereafter, we describe the performance metrics which will be used to compare the two machine learning models.

1.3.1 Feature extraction

Feature extraction can be simply described as a mathematical model to represent unstructured text to a numerical form, with each dimension of the of the numeric vector is a specific feature attribute. This task of feature extraction can be performed in several different ways, but to make the readers acquainted, we describe the two most popular and simplest methods for feature extraction for the ease of understanding.

- Bag of Words Model
- TF-IDF - Term Frequency Inverse Document Frequency

Bag of Words Model can be considered one of the most simplest vector space representational model for unstructured text. The bag of words model represents each text document as a numeric vector where each dimension is a specific word from the corpus and the value could be its frequency in the document, occurrence (denoted by 1 or 0) or even weighted values. The model's name is such because each document is represented literally as a 'bag' of its own words, disregarding word orders, sequences and grammar.

But the drawback of this feature extraction method is if the new sentences contain new words, then our vocabulary size would increase and thereby, the length of the vectors would increase. This would essentially mean that the vectors contains many 0s, thereby resulting in a sparse matrix, which essentially means we are not retaining information on the grammar of the sentences nor on the ordering of the words in the text.

TF-IDF or Term Frequency Inverse Document Frequency tries to combat the issues of bag of words model by using a scaling or normalizing factor in its computation. TF-IDF stands for Term Frequency-Inverse Document Frequency, which uses a combination of two metrics in its computation, namely: term frequency (tf) and inverse document frequency (idf).

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document} \tag{1}$$

$$idf_{t,d} = log\frac{number\ of\ documents}{number\ of\ documents\ with\ term\ 't'} \tag{2}$$

$$tf\_idf_{t,d} = tf_{t,d} * idf_{t,d} \tag{3}$$

In equation 1, the numerator n is the number of times the term 't' appears in a given document 'd'. Thus, each document and term would have its own tf value. This term gives

us the term frequency of of a given word in the document d. Equation 2 represents the inverse document frequency for the term t, which can be computed as the log transform of the total number of documents in the corpus C divided by the document frequency of the word w, which is basically the frequency of documents in the corpus where the word w occurs.

Combing the equations 1 and 2 the tf-idf score is calculated in equation 3, which is the TF-IDF score for word w in document d. To compare it with the previous method, Bag of Words just creates a set of vectors containing the count of word occurrences in the document (reviews), while the TF-IDF model contains information on the most important words and the less important ones as well. This also aids in achieving better performance out of the models. While both Bag-of-Words and TF-IDF have been popular in their own regard, but to perform more advanced tasks such as translating our given documents into another language, requires a lot more information on the documents. To addres this gap more advanced models such was Word2Vec, Continous Bag of Words (CBOW) are used.

1.3.2 Model architecture

We are using here a simple feed forward neural network using the Keras API.We are using the Perceptron algorithm with an activation function sigmoid to use it as a classifier. The model architecture consist of a single dense layer with one output neuron and sigmoid activation function. This means that the model takes in an input of a fixed length(here is the length of the vocab) and passes through a layer with a set number of neurons, in this case just one.The kernal initializer is set to 'glorot uniform',which initializes the weights of the layer according to the Glorot uniform initializer. The output of this layer is then passed through the sigmoid activation function, which maps the input to a value between 0 and 1.

The weight matrix of the dense layer, which is the only trainable parameter consist of size of the input vocabulary and the number of output neorons which is 1 and the resulting products are summed together to produce a scalar output for the neuron.

The activation function (sigmoid) applied to this scalar output then maps it to a probability value between 0 and 1, which can be interpreted as the model's prediction of the likelihood of the input belonging to the positive class.

The model is compiled with binary cross-entropy loss, which is commonly used in binary classification problems to measure the difference between predicted probabilities and the actual binary labels.The goal of this training is to minimize the loss function by updating the single weight parameter of the dense layer using the stochastic gradient descent (SGD) optimizer.

During training,the model is fed with training examples in batches, with each batch consisting of five samples.The model is trained for 50 epochs.

To monitor the model's performance during training, a validation split of 20% is used, which means that 20% of the training data is set for the validation set. The model's accuracy and loss on this validation set are computed at the end of each epoch, allowing us to track the model's progress and detect overfitting.

After training, the final performance of the model is evaluated using the test set,and the accuracy score is reported as the model's final performance metric.

1.3.3 Model Parameters

Single layer perceptron For the single layer perceptron model the following model parameters were used in Keras API

- Input Size: Length of Bag of Words Vector
- Output Size: 1
- Activation on input layer: Sigmoid with Glorot Uniform Kernel Initializer
- Loss function: Binary Crossentropy
- Optimizer: Stochastic Gradient Descent (SGD)

Multiple layer perceptron For the multiple layer model we have 5 layers utilising both dense and drop out layers.

- Layer 1 (Input Layer): Input size defined by length of vector with output size of 128 with activatio function of ReLU.
- Layer 2: It is a 50% drouput layer with size of 128
- Layer 3: Dense layer with 128 input and output size with activation function of ReLU
- Layer 4: It is a 50% drouput layer with size of 128
- Layer 5: Dense layer with input size of 128 and output size of 1 with output activation sigmoid.
- Optimizer: ADAM
- Loss function: Binary Crossentropy

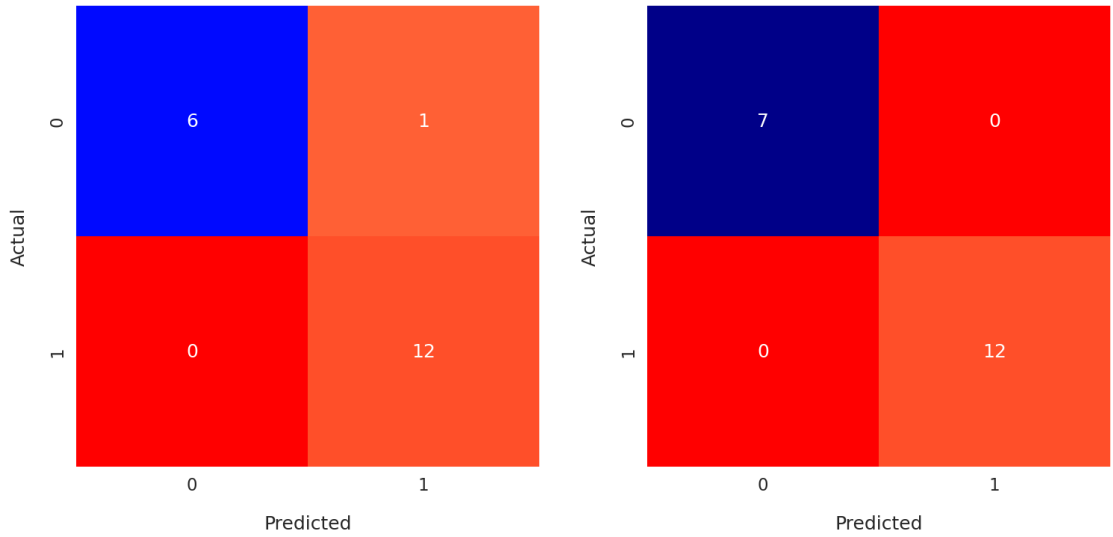
1.3.4 Classification Results

Here we include the classification results for two of our models. The first one being a single layer perceptron model and the second one being a multi-layer perceptron based model. Here we will detail about the model performance evaluation using the standard parameters such as accuracy, precision, recall as well as the confusion matrix for each of the two models.

We begin by describing the performance difference between the two models that were tested using the bag of words of model of vectorisation in Table **??**. The good performance may be attributed to similar nature of data and the task being limited to a binary classification task.

Table 4: Table describing the accuracy, precision, F1 score and recall for the two models

	Single Layer Model	Multiple Layer Model
Accuracy	0.9473684210526315	1.0
Precision	0.923076923076923	1.0
Recall	1.0	1.0
F1 Score	0.9600000000000001	1.0



(a) Single layer model

(b) Multiple layer model

Figure 11: Results depicting the confusion matrix for single and multiple layer model. Numerical 0 refers to the class of women scientists and 1 refers to the US Presidents

The above results are carried out on test data which was a subset from the consolidated data. The train and test split was 80 and 20 percent from the whole data. In case of machine learning model with only one layer there is one instance of women scientist which was wrong classified as US president. While in the case of the model with multiple layers using the same bag of words vectorisation method performs the classification task with 100% accuracy.

2 Part II: Comparison of *spaCy* and *NLTK*

In the second part of the project, we compare the performance of two well-known linguistic Python libraries: spaCy[5] and NLTK[6] on a textual dataset, in terms of sentence segmentation, tokenization and PoS Tagging.

2.1 Data Collection

A first step is to gather some textual dataset that we are going to use as a basis for the linguistic treatments. In order to do so, we create a function that, given an *url* and a size n, retrieves a textual dataset of n sentences extracted from the location *url*. In our case, we coded the function such that *url* can either be a Website URL or a local file path, which allows reproducability of the experiment on a large variety of datasets.

In order to measure the size of the text in terms of sentences, without using spaCy or NLTK yet, we adopted the naive approach that consists in splitting a text on the dot symbol. So the utterance *"Hey. I'm a NLP Student."* would be split into 2 sentences *"Hey."* and *"I'm a NLP Student.*. Of course, this strategy is questionable in terms of linguistic analysis, but we use it here only to get an order of size, and the sentences obtained are not stored for reuse, as this will be the role of spaCy and NLTK.

Our dataset consists in a 100-sentences-long text extracted from the Wikipedia article about Philosophy of Science[7]. A short extract is displayed below.

> Philosophy of science is a branch of philosophy concerned with the foundations, methods, and implications of science. The central questions of this study concern what qualifies as science, the reliability of scientific theories, and the ultimate purpose of science. This discipline overlaps with metaphysics, ontology, and epistemology, for example, when it explores the relationship between science and truth. Philosophy of science focuses on metaphysical, epistemic and semantic aspects of science. Ethical issues such as bioethics and scientific misconduct are often considered ethics or science studies rather than the philosophy of science. There is no consensus among philosophers about many of the central problems concerned with the philosophy of science, including whether science can reveal the truth about unobservable things and whether scientific reasoning can be justified at all. In addition to these general questions about science as a whole, philosophers of science consider problems that apply to particular sciences (such as biology or physics). Some philosophers of science also use contemporary results in science to reach conclusions about philosophy itself.

2.2 Sentence Segmentation

Calling spaCy's and NLTK's sentence tokenizers on the dataset, we find following results: NLTK identifies 98 sentences against 80 for spaCy. 45 sentences are identified by both libraries. NLTK's tendency to put a sentence boundary whenever a Wikipedia quotation is found (e.g [1]) can explain its higher sentence count. This is presented as a barplot in Figure 12.

[5] `https://spacy.io/`

[6] `https://www.nltk.org/`

[7] url: `https://en.wikipedia.org/wiki/Philosophy_of_science`, consulted on May 9, 2023.

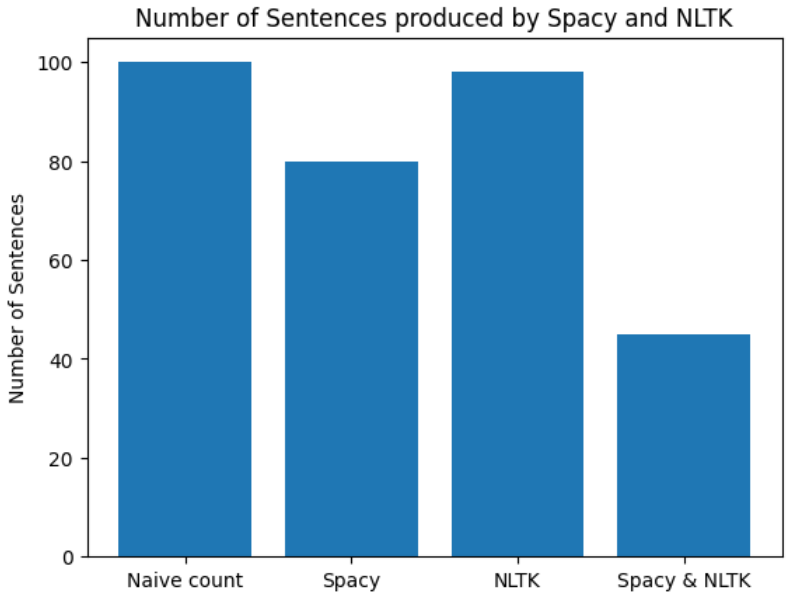


Figure 12: Comparison of spaCy and NLTK's sentence tokenizers

For example, this utterance is considered as one sentence by spaCy:

> 1) Thomas Kuhn's 1962 book The Structure of Scientific Revolutions was also formative, challenging the view of scientific progress as the steady, cumulative acquisition of knowledge based on a fixed method of systematic experimentation and instead of arguing that any progress is relative to a "paradigm", the set of questions, concepts, and practices that define a scientific discipline in a particular historical period.**[1]** Subsequently, the coherentist approach to science, in which a theory is validated if it makes sense of observations as part of a coherent whole, became prominent due to W. V. Quine and others.

... whereas NLTK splits it in 2 sentences:

> 1) Thomas Kuhn's 1962 book The Structure of Scientific Revolutions was also formative, challenging the view of scientific progress as the steady, cumulative acquisition of knowledge based on a fixed method of systematic experimentation and instead of arguing that any progress is relative to a "paradigm", the set of questions, concepts, and practices that define a scientific discipline in a particular historical period.
>
> 2) **[1]** Subsequently, the coherentist approach to science, in which a theory is validated if it makes sense of observations as part of a coherent whole, became prominent due to W. V. Quine and others.

There is also some difference in the way spaCy and NLTK handle space characters and empty lines surrounding text slots. Below, the characters "|" mark the beginning and end of the sentences. We see that despite having the same textual content, the 2 sentences differ in terms of surrounding characters. This can explain the fact that only arounf half of the

sentences are exactly the same for both libraries.

NLTK:

| Some philosophers of science also use contemporary results in science to reach conclusions about philosophy itself. |

spaCy:

| Some philosophers of science also use contemporary results in science to reach conclusions about philosophy itself. |

2.3 Tokenization

Regarding word segmentation, we analyze it by two tasks:

- computing unique tokens found **before sentence segmentation**: the vocabulary of the dataset.
- computing tokens found **after sentence segmentation**: token occurrences found in sentences of the dataset.

For each of these tasks, we will compared:

1) NLTK and spaCy's results
2) The intersection of their results
3) The results that are specific to each of these libraries

2.3.1 Token without sentence segmentation

In this part, without previous sentence segmentation, we performed tokenization using **NLTK** and **spaCy** on the whole dataset for comparing the vocabulary that both these libraries find. We compute :

- Tokens obtained by NLTK and its size
- Tokens specific in NLTK and its size
- Tokens obtained by spaCy and its size
- Tokens specific in spaCy and its size
- Intersection of tokens recognized by NLTK and its size.

The result is shown in Table5. See in Figure13 for its barplot.The scale of token occurrences between two libraires up to 0.997.

Libraries	Nb_token	Nb_token_only_in	Common_token
NLTK	861	16	845
spaCy	850	5	

Table 5: SharedTokensNoSentences

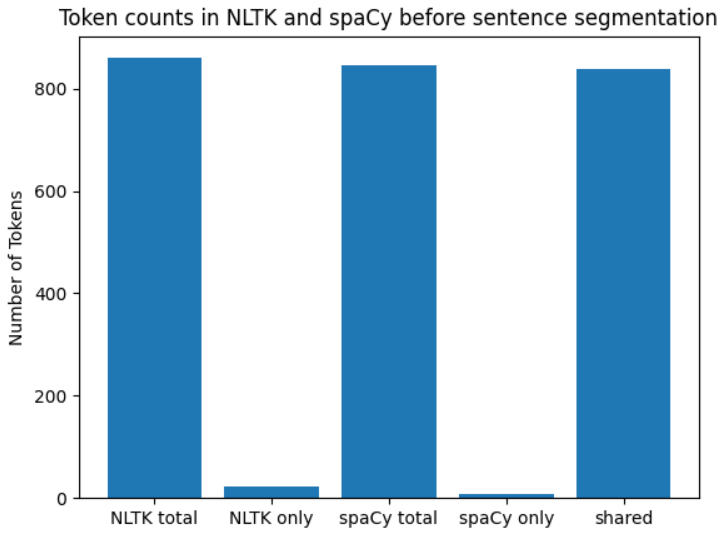


Figure 13: Comparison of spaCy and NLTK's tokens without sentence tokenization

2.3.2 Token with sentence segmentation

After completing word segmentation without sentence segmentation, we also performed tokenization on the shared sentences identified both by **NLTK** and **spaCy**. Here, we started at the sentence level and computed:

- Tokens obtained by NLTK and its size for each sentence
- Tokens specific in NLTK and its size for each sentence
- Tokens obtained by spaCy and its size for each sentence
- Tokens specific in spaCy and its size for each sentence
- Intersection of tokens recognized by NLTK and its size for each sentence.

And then we count the total number of shared token occurrences for all sentences. See in Table 6 for the result. Its corresponding barplot shown in Figure14.The scale of token occurrences between two libraires up to 0.995. The result is a bit better than tokenization without sentence segment, but the difference is very small.

Libraries	**Nb_token**	**Nb_token_only_in**	**Common_token**
NLTK	412	5	407
spaCy	414	7	

Table 6: SharedTokensInSentences

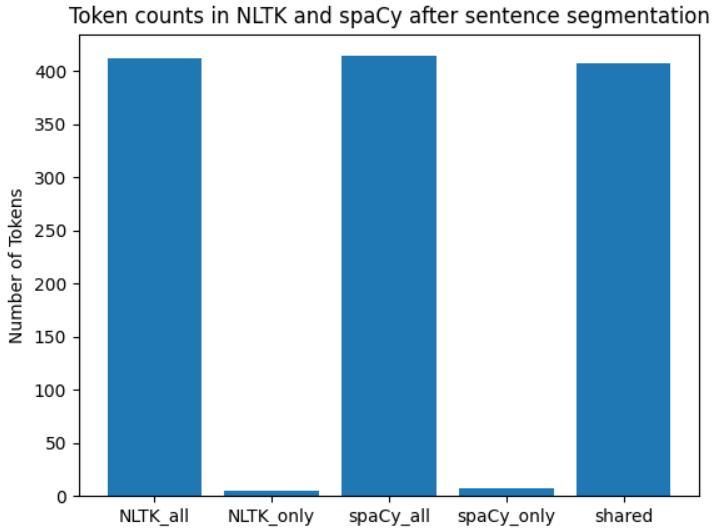


Figure 14: Comparison of spaCy and NLTK's tokens with sentence tokenization

From the two case of tokenization above, here we show list some examples for different result of tokenization between NLTK and spaCy in Table7

Libraries	**"cannot"**	**"1214–1294"**	**"intra-"**	**"(1620)-an"**	**"Ockham's"**
NLTK	"cannot"	"1214","-", "1294"	"intra", "-"	"1620", "-an"	"Ockham", " ' ", "s"
spaCy	"can", "not"	"1214-1294"	"intra-"	"(1620)-an"	"Ockham","'s"

Table 7: Different result of tokenization in NLTK and spaCy

- NLTK take "cannot" as one word, but spaCy splits it into two words: "can" and "not", even though there is no space between them.
- When a trait d'union between two words or numbers, NLTK breaks this whole into three words, e.g. "1214–1294" is divided into "1214", "-" and "1294, while spaCy treats it as one word.

2.4 PoS Tagging

In this part, we compared NLTK's and spaCy POS taggers. It is already important to note that, despite being called *universal*, NLTK's tagset slightly differs from UD's universal tagset[8]. Some changes are only in terms of naming (e.g the tag *"."* corresponds with UD's *"PUNCT"*) but some other constitute an information loss (e.g NLTK's *"CONJ"* is less precise that UD's *"CCONJ"* and *"SCONJ"*). This is the reason why we actually mapped spaCy's tag to NLTK's ones, in order to be able to compare them.

[8] `https://universaldependencies.org/u/pos/`

We find that, out of 407 TokensInSentence identified the same way by NLTK and spaCY, 360 of them are assigned the same tag by both libraries, which represents a ratio of agreement of approximately 88%. Now investigating *how* the tags differ, following matrix shows how NLTK's tags are classified by spaCy. For instance, all NLTK's *ADJ* are also classified as *ADJ* in spaCy, whereas only half of NLTK's *NUM* are also tagged as such in spaCy (the other half is tagged as *PRON*).

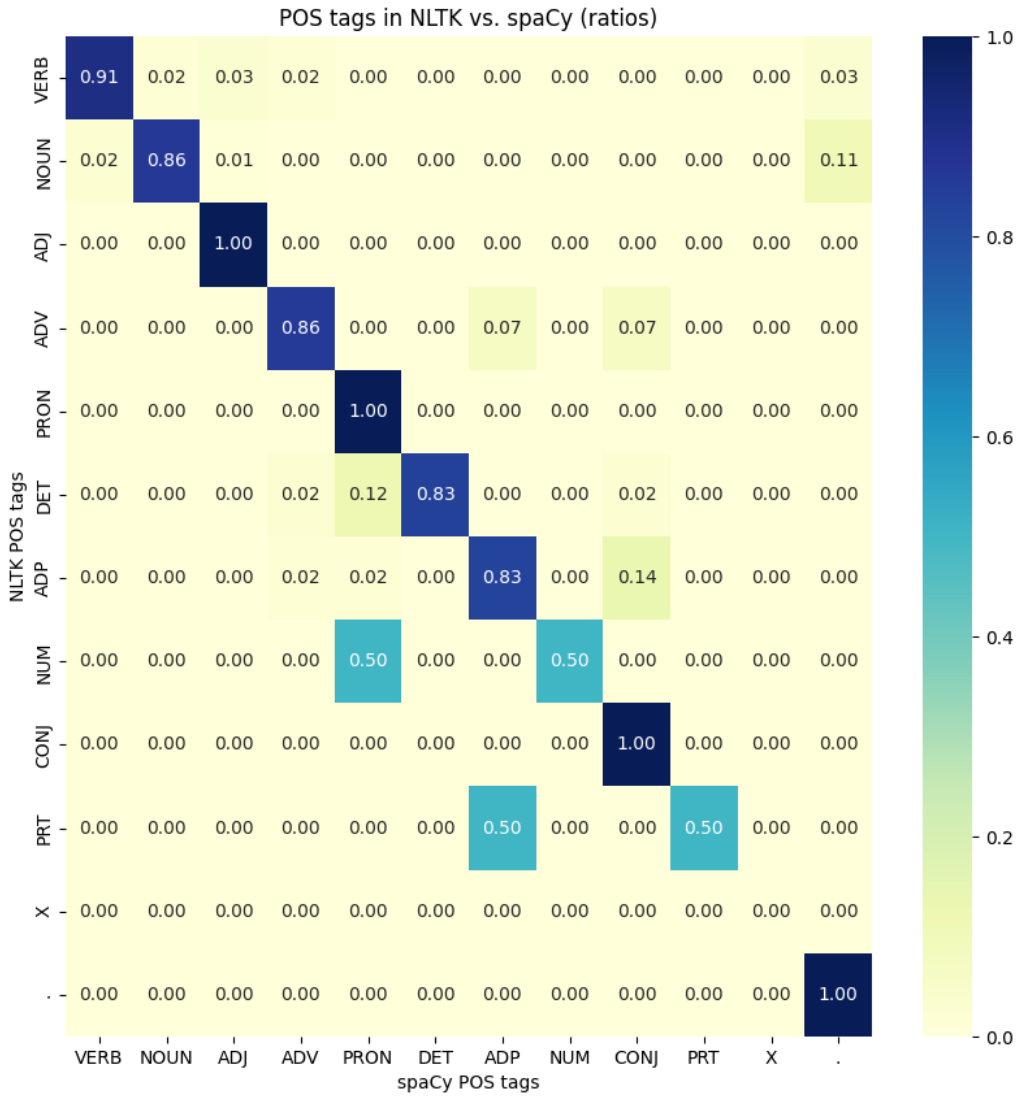


Figure 15: NLTK-spaCy POS comparison

The reverse study is presented below.

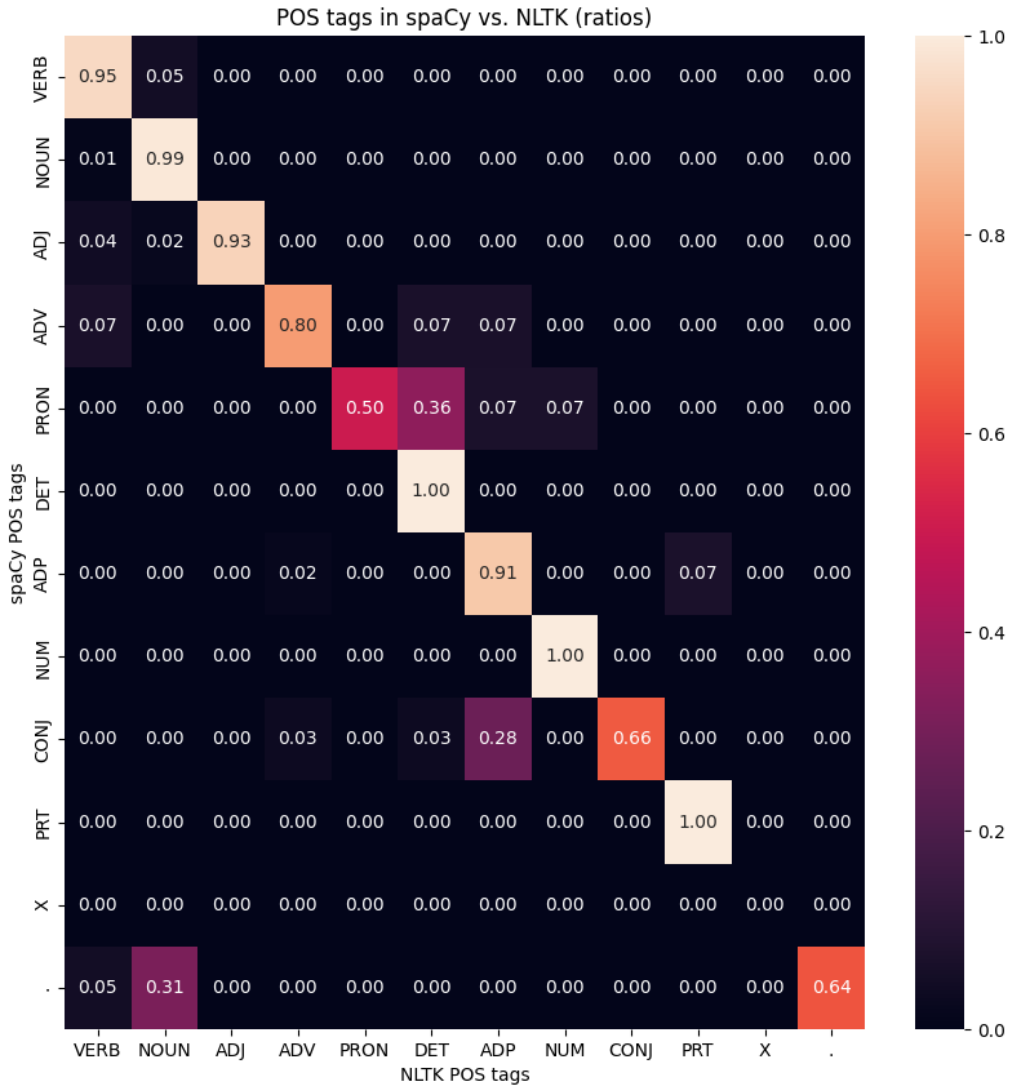


Figure 16: spaCy-NLTK POS comparison

In comparison with Figure 15, we see that all of spaCy's *NUM* are identified as such by NLTK. So, overall, spaCy used the tag *NUM* more often than NLTK. We see a good agreement between both libraries for tags like *VERB*, *NOUN* and *ADJ*. But there are strong disagreements with tags *NUM*, *PRT*, *CONJ* and *"."*.

3 Conclusion

As a conclusion, this project allowed us to become more familiar with Data Science tasks such as data collection, data analysis and data classification. We also had the opportunity to explore NLTK and spaCy's performance on a range of NLP tasks, which highlighted the importance of the choice of a linguistic library when conducting experiments on data.