Data Science Report

Pierre Epron, Maxime Renard

May 16, 2023

1 Introduction

The objective of the project is to build a collection of Wikipedia articles about people to classify them. The choice of labels was free. We chose to try to classify people based on their astrological sign.

It is important to note that our intention was not to prove the validity of astrology or demonstrate that astral signs can accurately predict personal characteristics. Rather, the choice of this subject was primarily motivated by a discussion surrounding the potential of modern AI systems to uncover correlations that may be challenging to substantiate.

We were interested in exploring whether classification models, when trained on biographical information and corresponding astral signs, could produce any discernible classification results. Our main objective was to examine the extent to which classification systems could potentially detect spurious associations or chance correlations, rather than verifying the accuracy of astrological predictions.

We therefore expected to produce a totally random classifier and would have been surprised if it had been the other way around.

Before starting to detail how we constituted our corpus, a few words on astrology. The definitions of star signs in astrology can vary depending on the astrological tradition followed and the system used. One common approach is based on the tropical zodiac, which aligns the signs with the position of the Sun relative to the Earth's seasons. Another approach to defining star signs is based on the sidereal zodiac, which considers the actual positions of constellations in the sky. This system takes into account the precession of the Earth's axis and acknowledges the shifting of the zodiacal constellations over time. Consequently, the dates associated with each star sign may differ from the tropical zodiac.

We chose to follow the tropical zodiac definition for our dates. It should be noted that even in the tropical zodiac system, not all sources agree on the exact dates. We chose to use the dates we found most often used in French sources. Furthermore, astrologers may incorporate additional factors such as planetary placements, aspects, and houses into their interpretation of star signs. We chose to not take into account such factors in our study for simplicity's sake.

Github repository : https://github.com/PierreEpron/tal-m1-ds

2 Data Collection

The data collection part is divided into three steps: (1) We collect information about people on DBpedia using SPARQL. (2) We use data collected from DBpedia to retrieve each people's entire wikipedia page. (3) We parse the results with spacy and/or stanza. We preferred to do parsing and storing before the analysis or the classification because on a large number of files it can take a long time.

The final structure of our data looks like this:

- /{SAMPLE_NAME}/abstracts.jsonl
- /{SAMPLE_NAME}/pages/*.txt
- /{SAMPLE_NAME}/spacy.pkl
- /{SAMPLE_NAME}/stanza.pkl

SAMPLE_NAME represents the sample name. The deliverable contains only one sample "sm" but a larger sample "lg" can be found on our github. Note that "sm" is a subsample of "lg" and therefore people who are in "sm" are also in "lg". The "abstracts.jsonl" file contains the result of DBpedia queries. The "pages" folder contains all complete wikipedia pages associated with an abstract element. The name of the page files is made up like this: PageName_AstralSign. We just remove the spaces in the page name. We keep the part in brackets which generally serves as a disambiguation to avoid homonymy problems. Without this, we could have two namesakes with the same sign that overwrite each other. The "spacy.pkl" and "stanza.pkl" files contain the parsing results. The README in the deliverable details the use of the various scripts that used to collect data. We did not aim for a specific number of items to collect for the "lg" sample. We just let the collector script run when we had time and we achieve to collect 16 978 items.

In the following sections, we detail why we chose to use DBpedia and why we decided to parse the files before the analysis.

2.1 DBpedia

DBpedia is a project that aims to extract structured information from Wikipedia and make it available as a linked open data resource. It can be seen as a semantic representation of Wikipedia, where the information from Wikipedia is transformed into a structured format that can be easily queried and linked to other data sources on the web. DBpedia extracts data such as infoboxes, categories, and links from Wikipedia articles and represents them using RDF (Resource Description Framework), a standard for representing information on the web.

So DBpedia is a good solution to retrieve Wikipedia entries using the power of sparql and semantic information contained in a knowledge graph. In our case, the SPARQL query targets humans (?s a wikidata:Q5) who have a name (?s a dbp:name ?n), a date of birth (?s dbo:birthDate ?bd), a place of birth (?s dbo:birthPlace ?bp) and an abstract (?s dbo:abstract ?abs). We also collect, if it exists, the date of death (OPTIONAL {{ ?s dbo:deathDate ?dd. }}). This allows us to easily estimate the number of living people in our corpus, even if it is quite possible that a deceased person does not have a death date.

Some clarifications on the query: We chose to use the Wikidata property (wikidata:Q5) because after several tests, it seems the best suited to filter out non-fictional humans. We chose to recover the places of birth because we left ourselves the possibility of using it for the analysis or even for the classification (example: classified only with a sample of people born in France). We did not have time to use it. The "abstract" represents the first paragraph of the associated Wikipedia page. We collect them, in addition to retrieving the entire page, to facilitate analysis. This allows us to compare the content of the whole page and its abstract without further processing.

Finally, we use the DBpedia URI to construct the Wikipedia page url. This works in most cases but sometimes the URI will not fetch the page (about once every 4000-5000 entries). We assume that this is due to a synchronization error between DBpedia and Wikipedia (example: the name of a Wikipedia page changes and the update on DBpedia is not yet done). In any case, our script ignores entries for which we cannot find a Wikipedia page.

2.2 Parsing

Since one of the goals of the project is to compare the spacy and stanza pipelines, we implemented both. At first, we were parsing documents on the fly but, very soon, it turned out that the parsing was taking too long. We decided it was better to parse them only once and store the results, even if the file sizes are quite large.

We can see from Table 2.1 that Stanza takes more than 50 times Spacy's time to analyze the "sm" sample. It is important to take into account that the Stanza pipeline has been lightened. We have retained only the essential processors for our analysis (tokenize, pos, lemma, depparse). It is therefore obvious that we used Spacy for the analysis and classification part of the data. The Spacy pipeline still takes 25 minutes to parse all the documents in the "lg" sample. So we decided to do it only once and save the results as mentioned above. The downside of this method is that we had to remove the vectors during serialization to reduce file sizes. It's not a significant problem since we do not use them.

Parser	**Sample**	**Sample Size**	**Parsing Time (h:m:s)**		**Output Size (mo)**	
			Total Time	Time/item	Total Size	Size/item
Stanza	sm	1000	2:00:05	7.205	137	0,137
Spacy	sm	1000	01:34	0.094	100	0,1
Spacy	lg	16978	24:56	0,088	1720	1.72

TABLE 2.1: Performances of parser by samples. We did not run stanza on the lg sample.

The model we choose from Spacy is the "en_core_web_sm"[1]. It is the smallest pipeline available with Spacy and it has exactly the same performance as the other pipelines for the components we are interested in (Tokenization, Sentence Segmentaion and POS tagging). We believe that pipelines share the same components.

3 Data Analysis

The corpus contains 16 978 pages. The number of pages is mostly evenly distributed among the twelve categories, as can be seen in Table 3.1. Similarly, the number of tokens and sentences seem equally shared, as seen in Table 3.2.

We perform a statistical analysis to test whether there is a correlation between the number of tokens in a page and the category.

[1] en_core_web_sm from Spacy

Sign	Size
Aquarius	1463
Capricorn	1459
Gemini	1456
Pisces	1447
Aries	1445
Virgo	1435
Scorpio	1431
Cancer	1420
Libra	1409
Taurus	1406
Leo	1349
Sagittarius	1258

TABLE 3.1: Number of pages per category

Astral sign	Number of tokens per page			Number of sentences per page		
	Min	Mean	Max	Min	Mean	Max
Aquarius	7	311.332194	6474	2	30.514696	566
Aries	9	321.118339	8478	2	32.180623	858
Cancer	10	313.405634	5810	2	31.063380	607
Capricorn	7	283.286498	5586	2	27.949280	504
Gemini	9	300.865385	5552	2	29.675137	556
Leo	8	316.418829	6198	1	31.080059	628
Libra	4	295.819730	4936	1	29.228531	466
Pisces	9	310.328265	7279	1	30.776780	580
Sagittarius	10	295.218601	9210	1	29.565978	928
Scorpio	9	290.607268	7922	2	28.988120	506
Taurus	9	293.492888	6291	2	28.993599	603
Virgo	7	342.894077	12913	1	32.798606	999

TABLE 3.2: Number of tokens and sentences per pages in each category

We start by performing a series of normality tests. None of the categories' page length follows a normal distribution, which is corroborated by visual confirmation on Figure 3.1. Because the samples do not follow a normal distribution, we must use a non-parametric test. We perform a Kruskal-Wallis H-test: it returns a p-value above 0.05 (0.77). We keep the null hypothesis: there is not enough evidence to support the presence of significant differences between the medians number of tokens of the categories, which means that the variations in the data are not due to systematic differences caused by the independent variables.

Looking at the wordcloud in Figure 3.2 reveals that when performing the classification, the months should be removed as there are systematically among the most common words and are directly linked to the category.

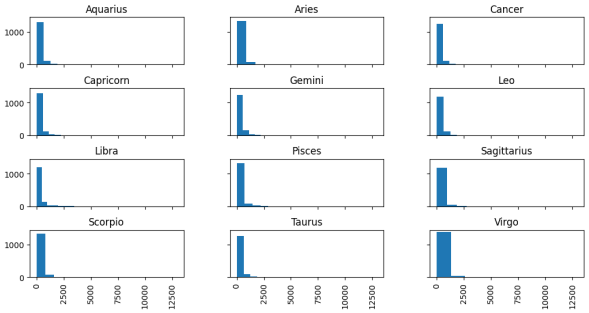

FIGURE 3.1: Histogram distribution of the number of tokens in a page, per astral sign

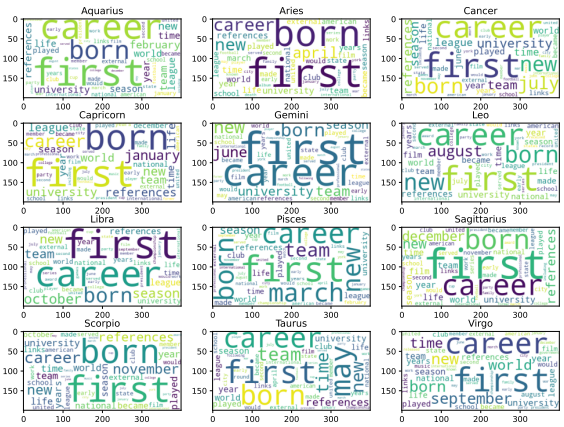

FIGURE 3.2: Wordcloud of the 50 most frequent words in a page, per astral sign

4 Classification

Since we have 12 different labels and the main goal is to do a binary classification, we set up 2 experiments:

- Try a binary classification on "Taurus" vs "Sagittarius". They are the Astrological signs of the authors.
- Try a classification on the whole corpus and on all the labels.

For the two experiments, we worked first on the abstracts and then on all the pages.

4.1 Methodology

As we assume that the results of the classification, whatever the configuration, will be random we decided to start by testing all our models with a basic preprocessing and default hyper-parameters. We kept the same preprocessing method as the one studied in class. We've removed stop words, including a selection of Part-Of-Speech tags. We did not use a lemma because we think it is less relevant on voluminous and very generic texts. All the processing is in the clean_tokens function of the /src/preprocess.py file.

4.1.1 Vectorizer

There are different options for vectorizing text. For this project, we explored two that are already implemented by sklearn: CountVectorizer and TfidfVectorizer.

CountVectorizer simply counts the occurrence of each word in the document and creates a vector representation based on those counts. On the other hand, Tfidf takes into account the frequency of a word in the document as well as its importance in the entire corpus. It assigns higher weights to words that are frequent in a document but rare in the corpus, as they are considered more informative. Tfidf also helps mitigate the impact of common words that appear in many documents, which might not carry significant meaning. In summary, CountVectorizer focuses on word counts, while Tfidf considers the importance of words in a document and the entire corpus.

Another way to vectorize text that we have not explored yet is to use word embedding like word2vec. This consists of using the hidden layers of a neural network trained as a language model as word vectors. This method is generally more efficient than the others but it is also more computationally intensive and therefore time-consuming. This could be a lead to explore later, but given the results that we will present below, there is little hope that it will work and that is expected.

We decided to do the default training with TfidfVectorizer because we did not want to remove the longest documents from our corpus. Tfidf seems more suitable for faithfully representing documents of different sizes

4.1.2 Models

In addition to the two imposed models (LogisticRegression and Perceptron), We decided to train an "ensemble model" which is generally more efficient on complex data. Ensemble approach helps reduce overfitting and improves the overall accuracy and robustness of a classifier. We chose to use a RandomForest which is relatively quick to train compared to a GradientBoosting for example. A random forest classifier is a machine learning algorithm that combines the predictions of multiple decision trees to make accurate predictions or classify data points. It works by creating an ensemble of decision trees, each trained on a different subset of the training data and using a random subset of features. During prediction, each decision tree in the random forest independently makes a prediction, and the final output is determined by a majority vote or averaging the individual predictions.

4.2 Results

All tables and figures below come from the Logistic Regression model. We did not include the performances from the other models in the report because they were similar. They can be found in the notebook /classification.ipynb.

4.2.1 Abstract Classification and keeping the months

Binary classification

When keeping the months, the binary classification performs well with all three models tested (Logistic Regression, Perceptron and Random Forest). As can be seen on Table A.2, accuracy and f-score for both classes are above or at 0.9.

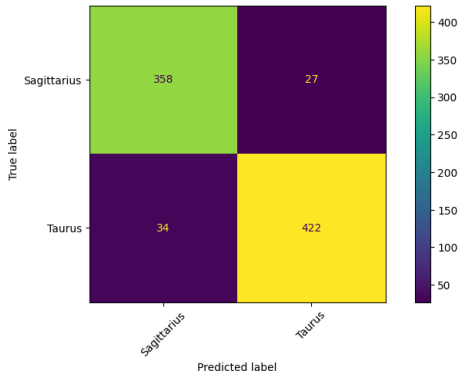

FIGURE 4.1: Confusion matrix for the Logistic Regression model on the binary classification of abstracts, with the months

N-ary classification

For the n-ary classification, the performance is worse but is still above random: accuracy and f-score around 0.5, see Table 4.1, for all models. The source of the higher error rate can be found by looking at the Confusion Matrix in Figure 4.2. It is due to the overlap between the months of signs. For example: Aries and Pisces share the month of March, which is why the model incorrectly identified a high number of Aries as Pisces.

	precision	**recall**	**f1-score**
Aquarius	0.51	0.46	0.49
Aries	0.50	0.44	0.47
Cancer	0.52	0.54	0.53
Capricorn	0.45	0.49	0.47
Gemini	0.53	0.50	0.51
Leo	0.49	0.56	0.53
Libra	0.52	0.54	0.53
Pisces	0.48	0.48	0.48
Sagittarius	0.47	0.51	0.49
Scorpio	0.54	0.54	0.54
Taurus	0.56	0.51	0.53
Virgo	0.52	0.54	0.53
Accuracy			0.51
Macro avg	0.51	0.51	0.51
Weighted avg	0.51	0.51	0.51

TABLE 4.1: Results for the Logistic Regression model on the n-ary classification of abstracts, with the months

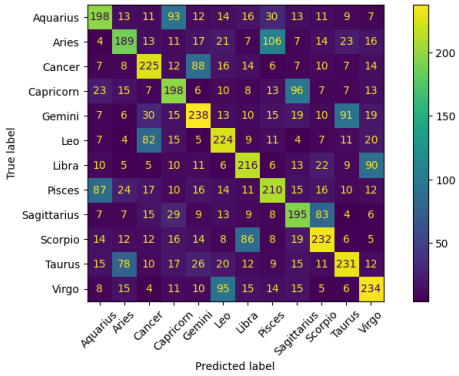

FIGURE 4.2: Confusion matrix for the Logistic Regression model on the n-ary classification of abstracts, with the months

4.2.2 Abstract Classification and removing the months

Binary classification

As was hypothesized earlier, the good performance of the classifiers was largely due to the presence of the months of birth. After removing them, we observe a performance close to random, as it can be observed in Table 4.2.

	precision	**recall**	**f1-score**
Sagittarius	0.47	0.45	0.46
Taurus	0.55	0.56	0.55
accuracy			0.51
macro avg	0.51	0.51	0.51
weighted avg	0.51	0.51	0.51

TABLE 4.2: Results for the Logistic Regression model on the binary classification of abstracts, without the months

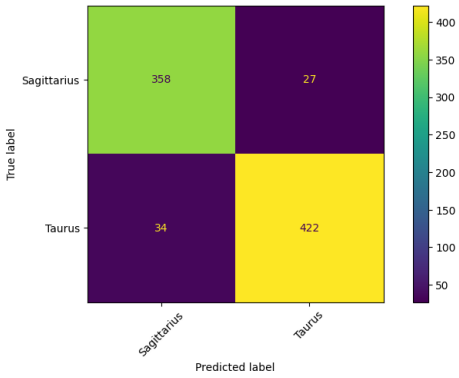

FIGURE 4.3: Confusion matrix for the Logistic Regression model on the binary classification of abstracts, without the months

N-ary classification

Similarly to the binary classification above, the performance observed, as seen in Table A.3 is close to random, with no particular pattern in the Confusion Matrix

(Figure 4.4).

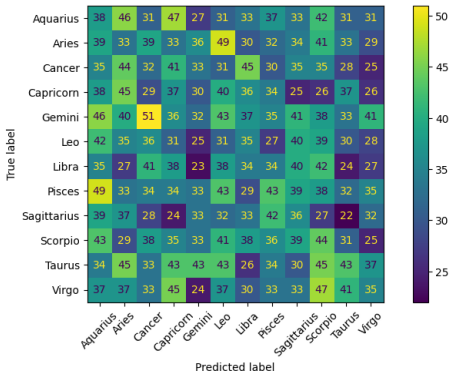

FIGURE 4.4: Confusion matrix for the Logistic Regression model on the n-ary classification of abstracts, without the months

4.2.3 Page Classification and keeping the months

We then carry out classification using the whole page rather than the abstract. We observe performance close to random for both binary and n-ary classifications even when keeping the months, which we hypothesize could be caused by the longer texts. You can find the results the notebook /classification.ipynb.

5 Spacy vs Stanza

For this part, all the functions are in the "stanza_spacy_compare.py" file. The "/spacy_vs_stanza.ipynb" notebook serves as a demonstration. The implementation is fully described in these two files. The results presented below are those obtained on the full pages of the "sm" sample.

5.1 Sentence Segmentation

Table 5.1 display the repartition of spacy, stanza and shared sentences by documents. We can observe that on average stanza produces more sentences than spacy and that on average a little more than half of the sentences are shared. It can also be observed that some documents have no phrases in common. After verification, this only concerns 48 (4.8%) documents. There is only one documents where spacy and stanza share the same number of sentence and this document has only one sentence.

5.2 Tokenization

The vocabulary of spacy contains is 50149 tokens by spacy when the vocabulary of stanza contains 49762 tokens. The shared vocabulary contains 47882 tokens which makes a difference of 2267 (4,52%) for spacy and 1880 (3,77%) for stanza. The difference between the vocabularies is therefore rather small. Tokens that differ between

	spacy sentences	stanza sentences	shared sentence
mean	27.694000	29.993000	16.068000
std	40.831747	42.627504	25.736389
min	1.000000	1.000000	0.000000
25%	8.000000	9.000000	3.000000
50%	16.000000	19.000000	9.000000
75%	32.000000	34.000000	20.000000
max	591.000000	592.000000	352.000000

TABLE 5.1: Repartition of sentence parsed by document for spacy, stanza and both.

vocabularies seem to be mostly tokenization errors or numbers coupled with a symbol (ex: 1–224).

All occurrences of shared tokens with or without sentence segmentation are in the notebook. We have two observations about this: First, the number of occurrences increases with sentence segmentation. This is totally expected in the sense that segmentation in practice gradually realigns tokenization errors. Second, most of the most common tokens are punctuation, articles or even prepositions. It logically follows the distribution of tokens in a text.

5.3 POS-tagging

Table 5.2 display the frequency of equivalent POS tagging for spacy and stanza. Most of the tags are recognized by both libraries at more than 90% or almost. We can see that there are major differences for interjections (INTJ), subordinating conjunctions (SCONJ) and others (X). Looking more closely at the results (in the notebook), we can see the different transfers. For example stanza tends to classify in PROPN (35%), PUNCT (28%) the X of spacy and spacy tends to classify in PROPN (59%) and NOUN (22%).

POS Tag	Spacy	Stanza	POS Tag	Spacy	Stanza
ADJ	91.48	72.55	PART	97.17	98.44
ADP	96.87	98.73	PRON	98.52	99.52
ADV	94.31	82.2	PROPN	89.14	97.22
AUX	99.07	99.85	PUNCT	98.73	99.51
CCONJ	99.78	99.37	SCONJ	50.13	53.48
DET	99.78	99.09	SYM	90.48	26.55
INTJ	23.53	30.77	VERB	97.06	96.44
NOUN	95.74	95.15	X	27.87	7.19
NUM	99.49	98.47			

TABLE 5.2: Frequency of equivalent POS tag for spacy and stanza.

6 Conclusion

The aim of this project was to classify the astrological signs of individuals based on their biographies sourced from Wikipedia. Our investigation revealed that the classification results were primarily random, with one notable exception. When we included the word indicating the month of birth in the text, some correlation emerged. However, it is important to acknowledge that this limited correlation does not validate the accuracy of astrological predictions. It may be argued that the textual information found on Wikipedia does not adequately represent an individual's true astrological traits, which is a valid point. The use of alternative and more complete data sources would be crucial for future research aimed at showing or not the possibility of the classification of astrological signs. From a more technical point of view, spacy has shown, for us, its superiority compared to stanza. The main reason lies in the processing time which is significantly longer.Finally, we will conclude by saying that we learned a lot about the complexity of combining tasks. Indeed as described in the instructions each task seems to be independently simple. But put together they are difficult to keep a coherence whether in the analysis or in the code.

A Appendix

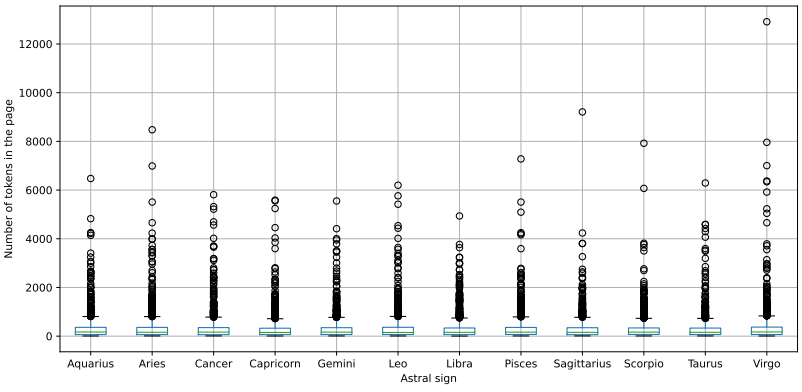

FIGURE A.1: Boxplot distribution of the number of tokens in a page, per astral sign

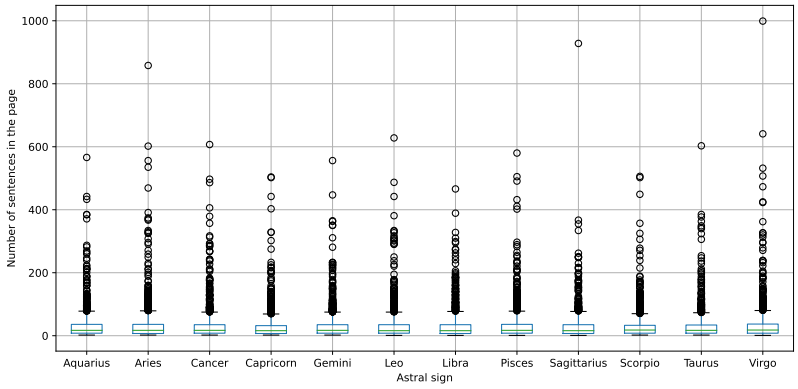

FIGURE A.2: Boxplot distribution of the number of sentences in a page, per astral sign

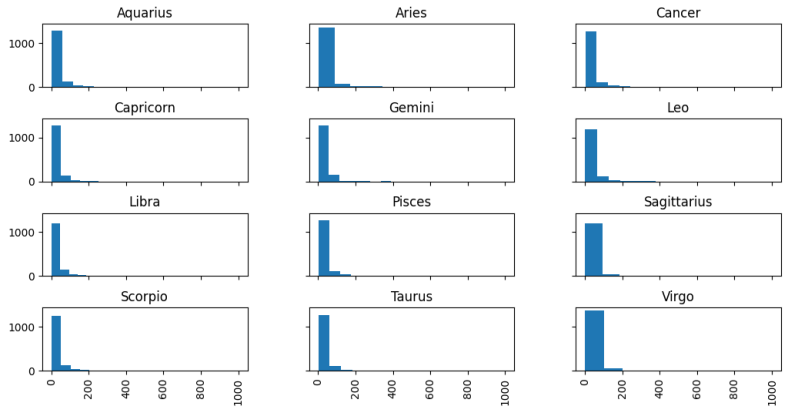

FIGURE A.3: Histogram distribution of the number of sentences in a page, per astral sign

Word
references
made
career
film
member
team
american
links
season
first
world
school
would
born
united
state
new
work
international
served
university
national
early
year
life
second
years
became
time
club
played
external
league

TABLE A.1: Most common words in the 12 categories

	precision	**recall**	**f1-score**
Sagittarius	0.91	0.93	0.92
Taurus	0.94	0.93	0.93
Accuracy			0.93
Macro avg	0.93	0.93	0.93
Weighted avg	0.93	0.93	0.93

TABLE A.2: Results for the Logistic Regression model on the binary classification of abstracts, with the months

	precision	recall	f1-score
Aquarius	0.08	0.09	0.08
Aries	0.07	0.08	0.08
Cancer	0.08	0.08	0.08
Capricorn	0.08	0.09	0.09
Gemini	0.09	0.07	0.08
Leo	0.07	0.08	0.07
Libra	0.08	0.08	0.08
Pisces	0.10	0.10	0.10
Sagittarius	0.08	0.09	0.09
Scorpio	0.09	0.10	0.10
Taurus	0.11	0.09	0.10
Virgo	0.09	0.08	0.09
accuracy			0.09
macro avg	0.09	0.09	0.09
weighted avg	0.09	0.09	0.09

TABLE A.3: Results for the Logistic Regression model on the n-ary classification of abstracts, without the months

	precision	recall	f1-score
Sagittarius	0.47	0.45	0.46
Taurus	0.55	0.56	0.55
accuracy			0.51
macro avg	0.51	0.51	0.51
weighted avg	0.51	0.51	0.51

TABLE A.4: Results for the Logistic Regression model on the binary classification of texts, with the months

	Aquarius	Aries	Cancer	Capricorn	Gemini	Leo
1	first	first	first	first	first	first
2	career	born	career	born	career	career
3	born	career	born	career	born	born
4	february	new	new	january	new	new
5	new	april	july	new	june	august
6	team	university	university	references	team	university
7	university	time	references	university	university	world
8	world	references	team	world	season	references
9	references	world	season	team	references	time
10	season	march	league	league	year	year
11	time	team	time	life	time	national
12	life	life	played	season	league	team
13	year	year	year	year	life	film
14	league	season	life	time	world	season
15	played	years	world	national	may	life
16	january	played	june	december	national	made
17	national	national	years	played	played	years
18	made	league	made	external	years	became
19	years	club	national	links	became	league
20	became	school	became	years	school	played
21	links	became	links	club	links	american
22	external	links	external	made	external	july
23	club	external	school	school	state	school
24	american	american	film	became	made	external
25	school	made	american	member	would	links
26	would	film	club	american	american	club
27	international	would	state	would	film	state
28	united	state	member	party	club	would
29	film	united	united	early	united	member
30	cup	early	second	state	member	second
31	second	work	early	work	early	united
32	state	second	york	united	second	early
33	member	international	would	film	international	served
34	early	member	work	international	party	international
35	war	war	served	president	football	college
36	work	college	international	second	college	work
37	president	served	player	city	york	city
38	party	may	may	may	president	president
39	football	january	city	professional	professional	series
40	march	football	march	served	served	york
41	professional	city	january	college	work	award
42	may	july	war	football	march	party
43	states	player	college	former	championship	may
44	served	series	games	music	following	record
45	college	family	john	march	match	march
46	city	album	november	cup	player	family
47	player	death	family	family	january	january
48	including	york	music	player	states	states
49	december	known	including	july	former	received
50	august	party	football	including	august	known

TABLE A.5: 50 most common words for the first 6 categories

	Libra	Pisces	Sagittarius	Scorpio	Taurus	Virgo
1	first	first	first	first	first	first
2	career	career	career	born	may	career
3	born	born	born	career	career	september
4	october	march	december	november	born	born
5	season	new	new	new	team	new
6	team	team	references	references	new	world
7	new	season	university	university	time	time
8	league	references	life	time	references	university
9	references	time	film	team	university	year
10	university	university	year	life	year	season
11	year	world	time	year	season	team
12	played	league	world	world	world	references
13	time	year	season	years	played	national
14	world	played	team	played	league	life
15	life	club	national	season	club	played
16	years	life	years	national	life	years
17	national	made	played	became	years	league
18	became	february	november	external	external	august
19	links	national	external	links	april	film
20	external	years	links	october	links	made
21	made	links	school	school	national	club
22	school	external	became	league	became	became
23	club	school	made	made	made	state
24	september	film	league	american	school	school
25	film	would	american	member	american	would
26	american	became	second	club	would	links
27	second	american	club	work	second	external
28	state	second	state	early	member	second
29	united	member	member	film	film	american
30	member	championship	series	united	united	member
31	international	international	early	served	round	united
32	served	state	would	would	state	music
33	would	early	work	state	city	president
34	player	united	city	international	international	party
35	may	may	served	second	early	early
36	early	city	united	president	january	january
37	games	served	international	may	following	served
38	series	cup	york	award	work	international
39	college	january	president	series	championship	may
40	party	work	championship	party	july	work
41	football	september	family	music	professional	june
42	professional	july	football	city	football	family
43	work	november	including	march	served	city
44	championship	games	college	york	player	championship
45	january	professional	award	college	march	december
46	july	april	party	january	cup	government
47	cup	match	cup	family	series	war
48	award	york	former	june	party	released
49	president	college	professional	cup	war	march
50	june	august	march	july	president	known

TABLE A.6: 50 most common words for the last 6 categories