Final Project for Data Mining

Baldi Valerio, 1940729
Dieni Saverio, 1946039

February 18, 2025

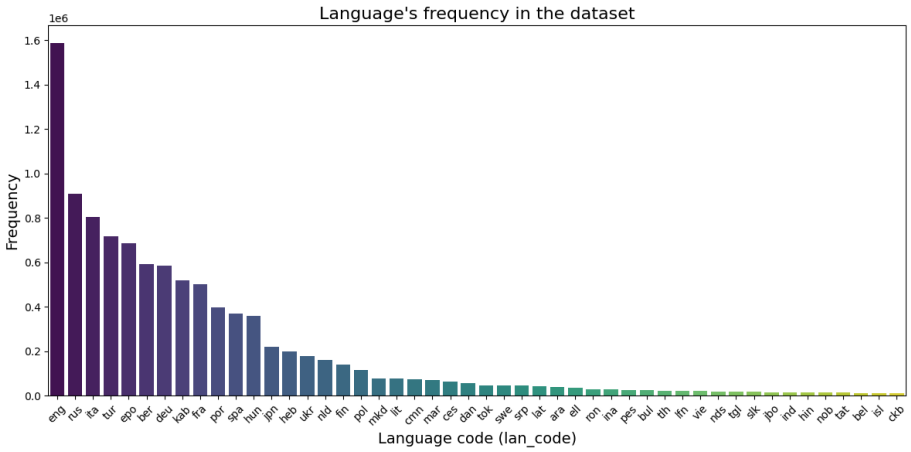
1 Introduction

For our project we thought about studying and exploiting data from different languages, in order to find methods able to detect the language of a given sentence. After a long research we found a dataset on Kaggle named "Big Language Detection Dataset", which contains around 10 millions sentences, labeled with the ISO 639-3 code of the sentence's language.
At the beginning we analyzed the distribution of the dataset using Tf-Idf and clustering techniques, then we trained two Logistic Regression models using Tf-Idf and Word2Vec embeddings and we compared them on their accuracy and other metrics.

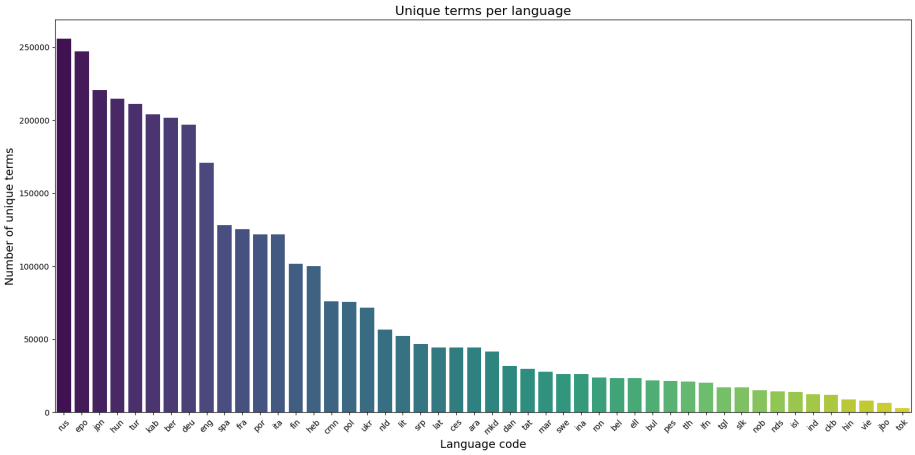
2 EDA

Since the dataset contains many sentences (+10 million) and languages (404) most of the languages don't have enough sentences to be relevant, we discarded all those of a language with less than the 0.1%, some had only 1 sentence, of the total number of sentences, doing so the number of sentences still remains around 10 million, but this reduces the number of languages from 404 to 48.

We noticed that the distribution of the sentences per language was unbalanced, the English language has the highest number of sentences (1.5 million) while the Central Kurdish has the lowest with 10 thousand sentences, as shown in the image below:

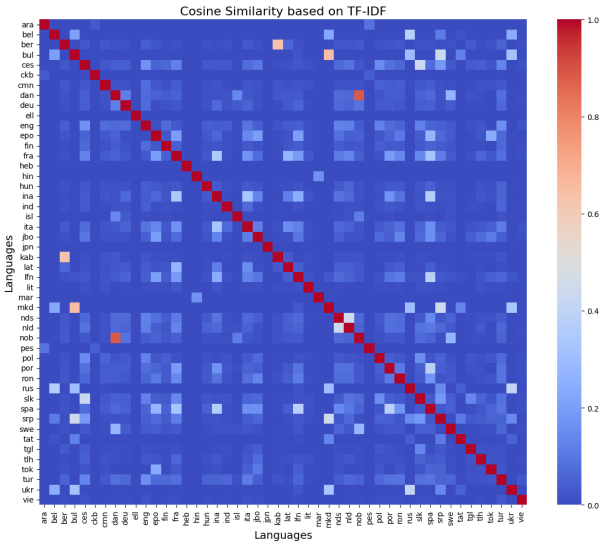


Also the number of unique terms per language wasn't balanced as for the sentences, but with a different distribution, with the Russian having 250 thousand of unique terms being the highest, the English 170 thousands and the Toki Pona 2 thousand (the lowest):

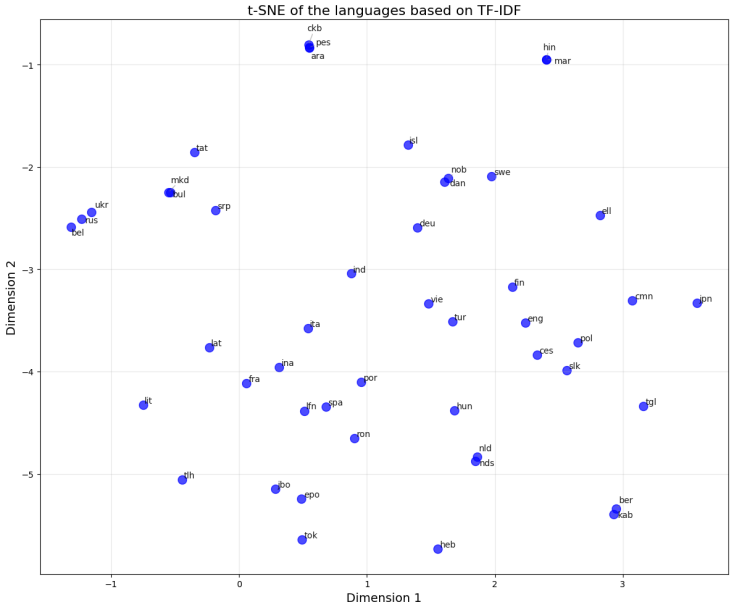


3 Tf-Idf analysis

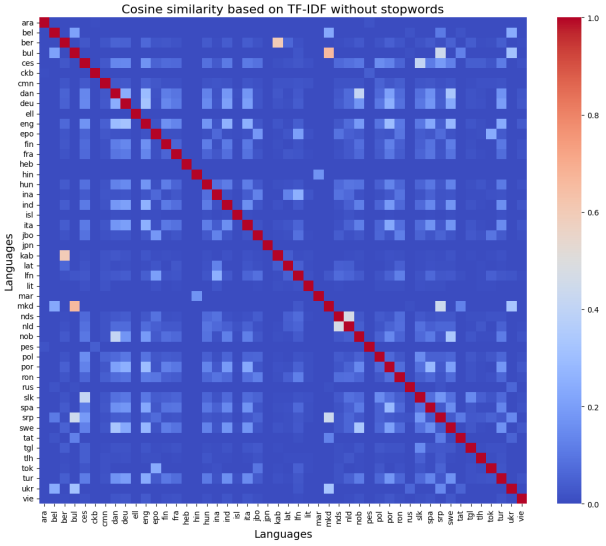
In this section we want to analyze the similarity between different languages. To achieve this goal we used Tf-Idf to build the language-term matrix and we considered as corpus the whole terms appearing in any sentence of the dataset. Once we had this matrix we calculated the cosine similarity between pairs of languages obtaining these results:



The cells with a white color, like Slovak and Czech, indicate a similarity value around 0.5, a dark blue cells that the similarity value is almost 0, like the majority of pairs, and a red cells that the similarity value is near 1, the most relevant case is Norwegian and Danish and all the languages paired with themselves.
Next we plotted the languages in a 2-D space using T-SNE dimensionality reduction, the resulting plot showed some interesting facts like the proximity of the Scandinavian languages, West-European languages, Cyrillic languages and Semitic languages.

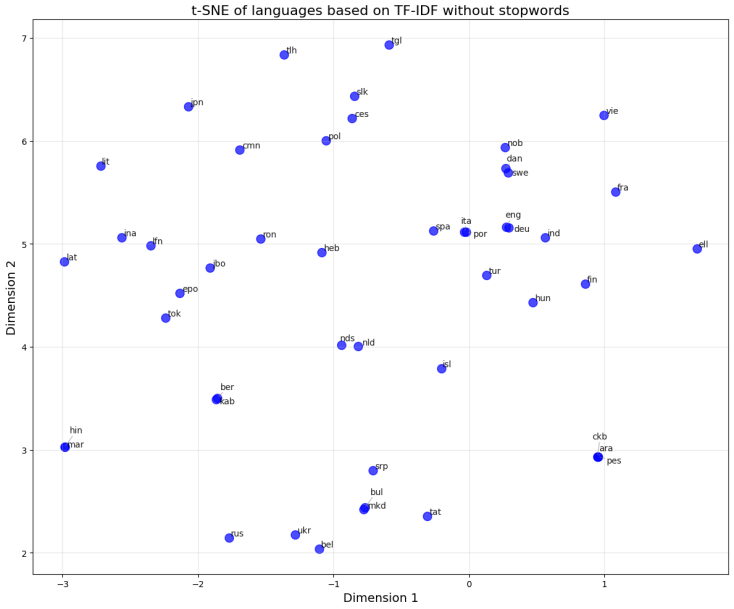


We tried to cluster the languages according to the previous results, but the resulting Silhouette score was inadequate, 0.077 with 34 clusters.
So we applied a stop-words removal to the dataset, we noticed that Norwegian and Danish are no longer similar as major difference:



Removing the stop-words leaded us to a slight decrease in the Silhouette score of the clustering, 0.059 using 30 clusters. We noticed a similar behavior of the projections, only the European languages seem to be positioned differently,

before Latin languages were far from English and Deutsch, but now these two are located in the same area of Italian, Portuguese, Spanish.



4 Language Recognition with clustering

Our goal was to recognize the language of a given sentence so we approached this problem under two different methodologies using supervised and unsupervised models. Our first attempt was to cluster the documents. We reduced the number of languages and sentences for the cluster discarding the languages with less than 1% of the total sentences to make the dataset lighter, the new dataset contains +9 million sentences, divided into 18 languages .
Then we assigned a language label to each cluster deriving from the language that appeared the most in the cluster. For a new given sentence we computed the vector, we looked for the nearest centroid to it to detect the language. At the beginning we clustered using straightforward the K-means++ algorithm, but the resulting clusters were mostly labeled with the English language, because it was the most frequent language by far, for our purpose this nullifies the effectiveness of our clusters. So our solution was to balance the dataset, by selecting the same number of sentences for each language, equal to the number of sentences of the language with the least. This improvement enriches the quality of the clusters, but still some of the languages were not identified by the clusters. So we increased the number of clusters, with the aim of producing clusters able to capture all the languages and to contain almost a single language, or at least a very predominant language.
After all these adjustments we measured an accuracy of 59.4%, that is a decent

result for a very naive strategy.

5 Language Recognition with Logistic Regression

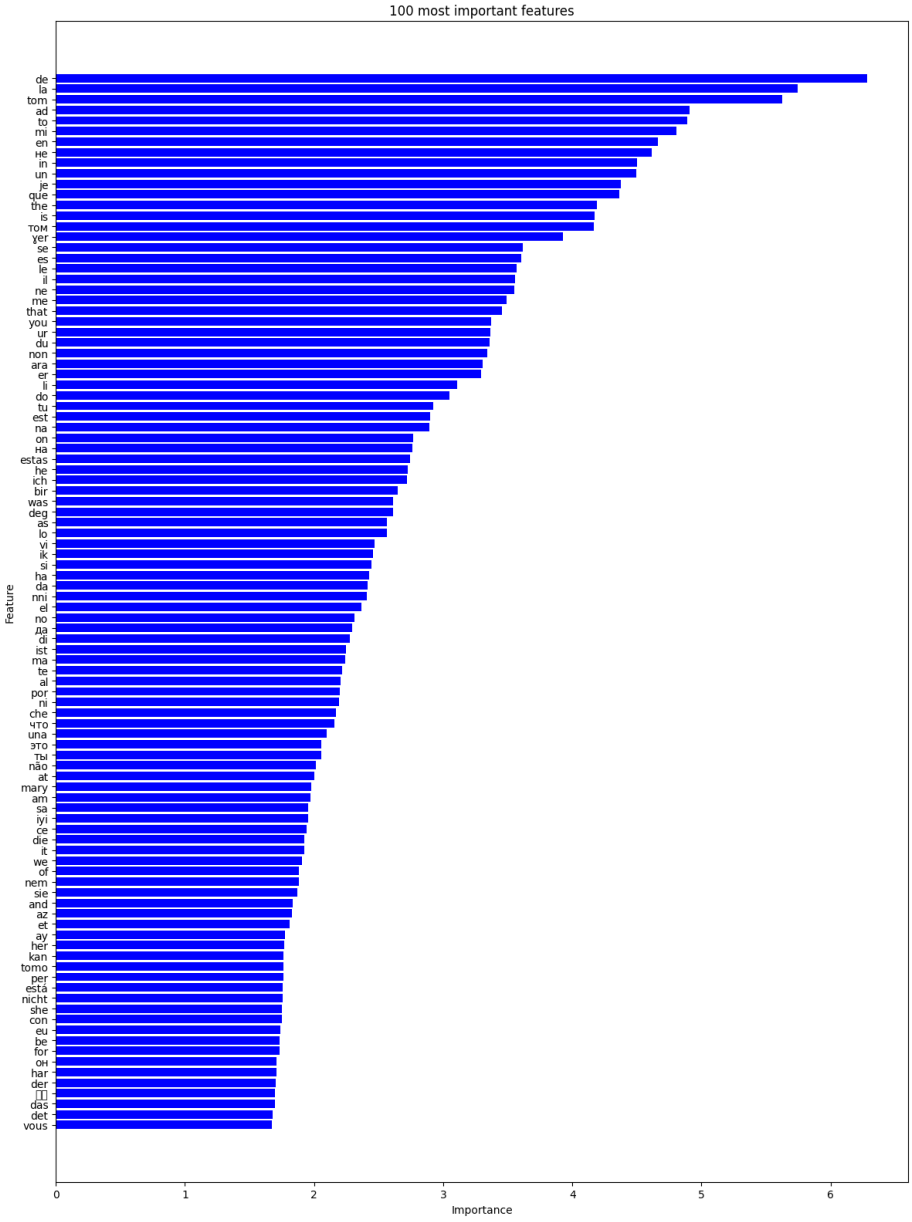
Our next approach was to use logistic regression to classify the language of a sentence. We transformed the dataset to vectors using TfidfVectorizer with 50000 features, we split it into training and testing sets and trained the logistic regression model.

Language code	Precision	Recall	F1-score	Support
ara	0.99	0.81	0.89	7599
bel	0.72	0.42	0.53	2500
ber	0.89	0.83	0.86	118909
bul	0.72	0.69	0.71	4922
ces	0.86	0.77	0.81	12793
ckb	1.00	0.67	0.80	2135
cmn	0.21	0.00	0.00	14394
dan	0.84	0.82	0.83	11160
deu	0.99	0.99	0.99	116759
ell	1.00	0.90	0.95	6935
eng	0.99	0.99	0.99	317329
epo	0.98	0.98	0.98	137848
fin	0.94	0.87	0.90	27805
fra	0.99	0.97	0.98	100167
heb	1.00	0.92	0.96	39875
hin	0.88	0.67	0.76	2865
hun	0.98	0.89	0.93	71898
ina	0.74	0.81	0.77	5493
ind	0.98	0.80	0.88	3204
isl	0.99	0.74	0.85	2489
ita	0.98	0.96	0.97	160487
jbo	0.80	0.80	0.80	3224
jpn	0.37	1.00	0.54	44269
kab	0.84	0.85	0.84	103507
lat	0.85	0.70	0.77	8403
lfn	0.84	0.73	0.78	4250
lit	0.95	0.89	0.92	15296
mar	0.97	0.86	0.91	13876
mkd	0.91	0.83	0.87	15366
nds	0.78	0.78	0.78	3549

Language code	Precision	Recall	F1-score	Support
nld	0.98	0.93	0.96	32010
nob	0.56	0.47	0.51	2838
pes	0.99	0.91	0.95	5180
pol	0.93	0.85	0.89	23308
por	0.98	0.96	0.97	79795
ron	0.89	0.77	0.83	5561
rus	0.97	0.96	0.97	181390
slk	0.70	0.46	0.56	3287
spa	0.97	0.94	0.96	73944
srp	0.85	0.64	0.73	9117
swe	0.92	0.84	0.88	9311
tat	0.94	0.62	0.75	2745
tgl	0.93	0.87	0.90	3708
tlh	0.91	0.55	0.68	4432
tok	0.98	0.98	0.98	9335
tur	0.96	0.95	0.96	142996
ukr	0.96	0.86	0.90	35823
vie	1.00	0.93	0.96	4216
accuracy			0.92	2008302
macro avg	0.88	0.80	0.83	2008302
weighted avg	0.94	0.92	0.93	2008302

The results are very promising so we were satisfied with them. It is noticeable that Oriental languages like Chinese and Japanese have very poor metrics, so we tried to understand this behavior.

The image below shows the 100 most important features considered by the model. It's clear that there are no Chinese nor Japanese terms, probably many terms from these languages were discarded by the TfidfVectorizer.



6 Stop-words removal for Language Classification

In this section we want to analyze the effect of removing stop-words from the sentences to classify their language. We built two light-classifiers containing

only the Italian, Spanish and English sentences using Logistic Regression. We processed the sentences as previously described for the first model, for the second we also removed the stop-words as only difference. These are the results:

Model with Stop-words

Language code	Precision	Recall	F1-score	Support
eng	1.00	0.99	1.00	316643
ita	0.96	0.99	0.98	161561
spa	0.99	0.93	0.96	74110
accuracy			0.99	552314
macro avg	0.98	0.97	0.98	552314
weighted avg	0.99	0.99	0.99	552314

Model without Stop-words

Language code	Precision	Recall	F1-score	Support
eng	0.99	0.91	0.95	316643
ita	0.71	0.98	0.82	161561
spa	0.96	0.49	0.65	74110
accuracy			0.87	552314
macro avg	0.89	0.79	0.81	552314
weighted avg	0.90	0.87	0.87	552314

Looking at the results it's evident that removing the stop-words reduces the quality of the classifier, indeed in the previous plot of the most important features many of them were stop-words so we deduced that they are useful to distinguish a language from another.

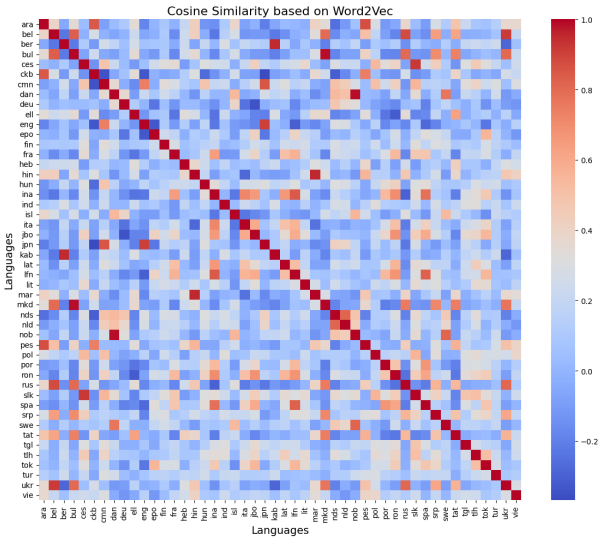
7 Word2Vec

As last attempt to build a classifier we tried to produce the embeddings using Word2Vec model. But first we present some basic use of Word2Vec.
To prepare the Word2Vec model we first preprocessed the sentences of the dataset removing special and numerical characters, transforming the sentences to lowercase, and at the end tokenizing them. Next we set the vector size to 100, window to 5 and minimum number of appearence to 15 as parameters of the model. We tried to do some experiments on the model like finding the nearest words to one, cat was linked to:

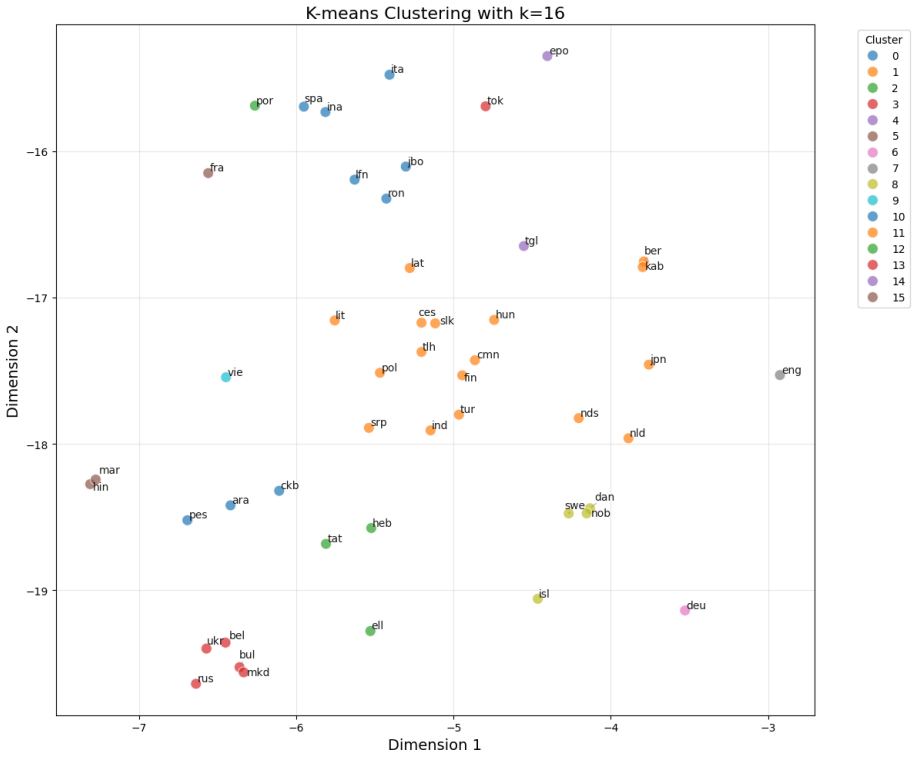
Word	Similarity
dog	0.9026539325714111
boy	0.8839496374130249
horse	0.8637125492095947
rabbit	0.8523182272911072
squirrel	0.849483072757721
bird	0.8340262174606323
snake	0.8270018696784973
puppy	0.8264356851577759
flower	0.8174018263816833
goat	0.8106449842453003

Next we tried to sum the vectors to understand the properties of the model, so when we tried to do King + Woman - Man the model successfully found Queen, but the most of the other trials fails like Kitten + Dog - Cat returned Flashlight and only as second result there was the correct answer Puppy. For many other examples the correct answer wasn't even in the first 3 positions, so our model failed in this task.

As before we considered the analysis of the languages, as done with Tf-Idf, and trying to use the Word2Vec model to understand the similarities between languages. We followed the same approach, but using the vectors produced with Word2Vec. The results are significantly different as shown in the similarity matrix:

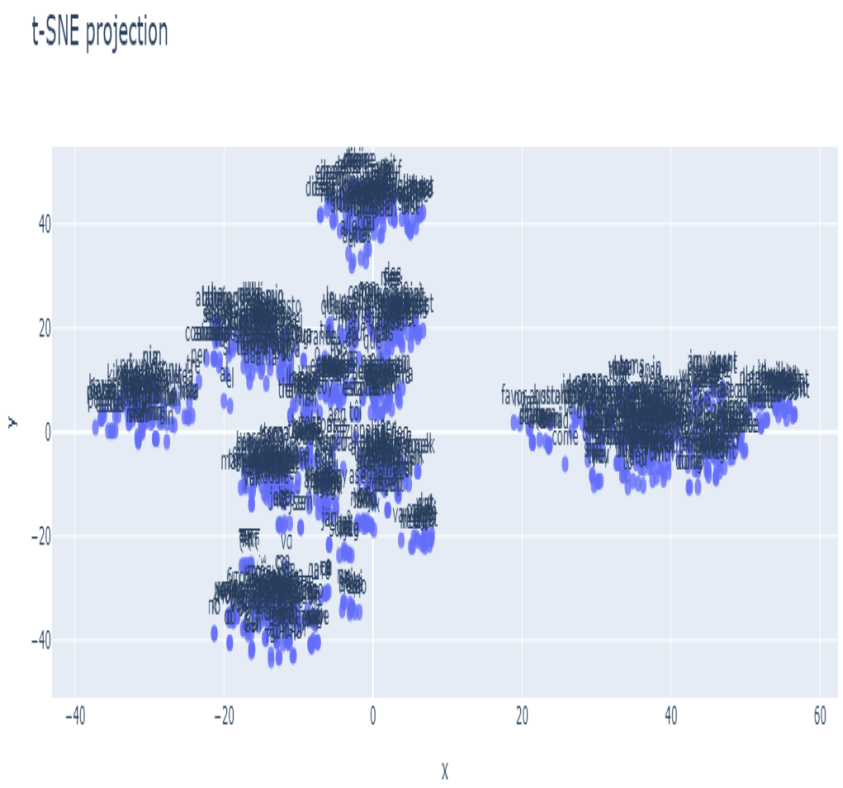


Languages now look to be more similar, blue cells seem to be sparser and some pairs have a high similarity as: Ukrainian and Belarus, Macedonian and Bulgar, Berber and Kabyle. Next we proceeded to cluster the languages, using K-Means++, according to these results and obtained the following diagram:



This clustering had a Silhouette score equal to 0.22 with 16 clusters, a big improvement compared to the previous results. We noticed that the languages are well divided as in the top-left there are Latin languages divided in different clusters, in the bottom-left East-European languages in the same cluster, also Middle-East languages are clustered together, same as Scandinavian, North-African and Indian languages. But there is a central zone which results to be very confused, with many diverse languages in the same area.

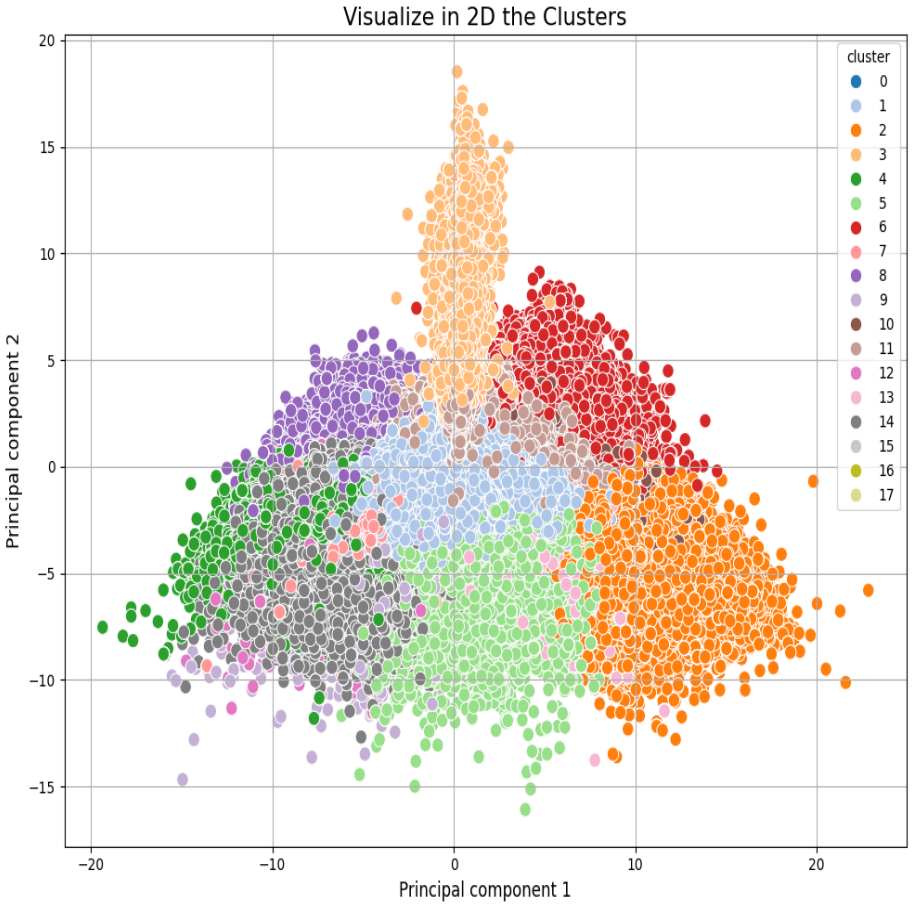
We also plotted the diagram of some words projected in 2D using t-SNE:



The plot shows that the terms are quite well divided in groups by language so it is reasonable to think that clustering the entire sentences can be quite effective to classify the language, more than before using Tf-Idf.

8 Language Classification with clustering using Word2Vec

We tried to implement this idea of clustering the sentences starting from the already balanced dataset in section 4, we followed the same approach of section 4 to cluster. In this case we used Word2Vec to calculate the vector of a sentence, we preprocessed the sentences as described in section 7, and the vector of the sentence is computed as the mean of the vectors of its tokens. The image below represents the clusters:



In this case the number of clusters necessary to cover almost all the languages, is lower (18) and the clusters were divided as follows:

Cluster	Language	Fraction
0	heb	1.000000
1	tur	0.996263
2	eng	0.942213
3	rus	0.520262
4	epo	0.795916
5	hun	0.394151
6	deu	0.985483
7	por	0.980784
8	epo	0.986274
9	fra	0.985055
10	nld	0.995065
11	jpn	0.385288
12	ita	0.987914
13	fin	0.889904
14	spa	0.949365
15	hun	0.999346
16	fin	0.997908
17	ber	0.510781

Not only the clusters cover 15 of the 18 languages but also most of them contain a great majority of a single language. Using this technique gave the following accuracy: 77% which is much bigger than the previous using Tf-Idf and requires much less time to compute it because the vectors have less dimensions, 100 instead of 50000 features.

9 Language Classification with Logistic Regression using Word2Vec

Lastly we also deployed a Logistic Regression model using our Word2Vec model. These are the results:

Language	Precision	Recall	F1-score	Support
ara	0.97	0.95	0.96	7730
bel	0.94	0.67	0.79	2531
ber	0.82	0.80	0.81	118244
bul	0.83	0.66	0.73	4922
ces	0.90	0.92	0.91	12776
ckb	1.00	0.91	0.95	2130
cmn	0.00	0.00	0.00	14488
dan	0.83	0.93	0.88	10989
deu	1.00	1.00	1.00	116935
ell	1.00	0.98	0.99	6961
eng	1.00	1.00	1.00	317324
epo	0.99	1.00	0.99	137128
fin	0.99	0.97	0.98	27818
fra	0.99	0.99	0.99	100248
heb	1.00	0.99	1.00	39841
hin	0.89	0.81	0.85	2915
hun	0.99	0.98	0.99	71566
ina	0.92	0.88	0.90	5499
ind	0.98	0.95	0.97	3143
isl	0.99	0.94	0.96	2492
ita	0.99	0.99	0.99	161021
jbo	0.96	0.95	0.96	3241
jpn	0.59	1.00	0.74	44021
kab	0.77	0.74	0.76	103666
lat	0.96	0.92	0.94	8299
lfn	0.94	0.85	0.89	4283
lit	0.98	0.97	0.98	15230
mar	0.96	0.98	0.97	13861
mkd	0.86	0.91	0.89	15590
nds	0.96	0.94	0.95	3598
nld	0.99	0.98	0.99	31838
nob	0.82	0.38	0.52	2865
pes	0.98	0.95	0.97	5132
pol	0.98	0.97	0.97	23316
por	0.99	0.99	0.99	79606
ron	0.97	0.93	0.95	5634
rus	0.98	1.00	0.99	181990

Language	Precision	Recall	F1-score	Support
slk	0.86	0.59	0.70	3376
spa	0.98	0.98	0.98	73968
srp	0.85	0.75	0.79	9040
swe	0.95	0.93	0.94	9401
tat	0.99	0.85	0.92	2753
tgl	0.98	0.96	0.97	3582
tlh	0.92	0.74	0.82	4458
tok	1.00	1.00	1.00	9441
tur	0.99	0.99	0.99	143579
ukr	0.97	0.94	0.96	35654
vie	1.00	0.99	1.00	4179
accuracy			0.95	2008302
macro avg	0.92	0.89	0.90	2008302
weighted avg	0.95	0.95	0.95	2008302

These results are very promising, slightly better than the other Logistic Regression model that used Tf-Idf, with an accuracy of 0.95 % vs 0.92%. Also in this case the Chinese Language was not correctly classified, but the Japanese has improved.

10 Conclusion

From these studies we understood the difficulties of working with large datasets and in particular with textual data. During our work we tried different approaches to analyze the dataset and some of them were left behind because of the enormous amount of space required. Between all our different approaches the most successful one was the Logistic Regression model using the Word2Vec vectors, but also the Logistic Regression model using Tf-Idf had great results. However Word2Vec achieve better results in an unsupervised environment since it reduces the dimensionality in a much more efficient way because it captures the semantics of a word instead of its syntax.