The Spanish Political Landscape

Sentiment Analysis

Pablo Martínez-Agulló
Andrés Caro-Acosta

May 2019

1. Introduction
2. Scraping
3. Data Processing
   - Data Preparation
   - EDA
4. Conclusions
   - Most used words
5. Next steps
6. Backup
   - Functions
   - Word Clouds before cleaning common words
   - Misc.

Introduction

Politics is always a hot topic anywhere and with the latest Spanish elections, people are still shaken up. The objective of this project is to understand the current sentiment of Spanish voters by analyzing newspaper articles from major publishers in order to gain some general insight before the upcoming municipal elections

Objectives

- Analyze and learn about the current mood of the voters
- Try to predict the results of the upcoming elections

Scraping

Tutorial Scrapping:
`https://github.com/resnad/web-scraping-tutorial-series/blob/master/web-scraping-part-1.ipynb`

Data Processing

Steps

- Data Cleaning
- Exploratory Analysis of Data
- Apply Sentiment Analysis

Tutorial

The code for this exercise can be foudn at:
`https://github.com/MartinezAgullo/NLP-for-Politics`

We got the inspiration from:
`https://github.com/adashofdata/nlp-in-python-tutorial`

Data Cleaning

The data scraped from the web must be cleaned, which means to remove the meaningless terms. This is important because feeding dirty data into a model will give us results that are meaningless.
The cleaning is divided in several steps (See "Backup" section for an example)

First round:

- Make text all lower case
- Remove punctuation
- Remove numerical values
- Remove common non-sensical text (/n, /t)

Second round:

- Remove stop words, which typically are the most common short function words (in English would be words such as the, is, at, which, and on)

After the tokenization (next slide), there is a third round in which we remove the words which have meaning but are very common for all parties and, therefore, do not help us to differentiate the party.

- libraries: **re** and **string**

Classification

The study framework is focused in the four parties that received the most votes during the past Spanish national elections: PSOE, PP, Podemos and Ciudadnos.
We build a function to do detect about which party is each article. This function is base in the search of key words (see backup for details)

	party	text
0	PSOE	PSOE sánchez intentará gobernar solitario pese...
1	PP	PP españa sorprende europa la socialdemocracia...
2	Cs	Cs rivera cierra puerta nuevo pacto abrazo sán...
3	Pdms	Pdms el centro derecha nunca gana elecciones c...

Figure: Information cleaned and classified

Now the tokenization is performed with:

- from **sklearn.feature_extraction.text** import **Vectorizer**

Exploratory Data Analysis

After the data cleaning step where we put our data into a few standard formats, the next step is to take a look at the data and see if what we're looking at makes sense. Before applying any fancy algorithms, it is always important to explore the data first.

- Most used words
- Word clouds:
  - from **wordcloud** import **WordCloud**

Sentiment Analysis

The main library for NLP is `nltk` and we are going to use `TextBlob` which is build on the top of `nltk`

- from **textblob** import **TextBlob**: It provides a simple API for diving into common NLP tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. For each word, TextBlob assigns a number between -1 and 1 to define the polarity (being -1 very negative, 0 neutral and 1 very postive) and a number between 0 and 1 to define its subjectivity.
- from **googletrans** import **Tanslator**: TextBlob works better in English, therefore, we translate our BoW to English

Conclusions

Most used words

- PSOE -	
puigdemont	64
catalunya	61
resultados	60
gente	59
vox	59
ser	57
erc	56

- PP -	
casado	76
españa	71
vox	70
dos	61
derecha	60
catalunya	59
resultados	53

- Unidas Podemos -	
españa	52
escaños	46
vox	45
derecha	43
dice	40
podemos	40
resultados	38

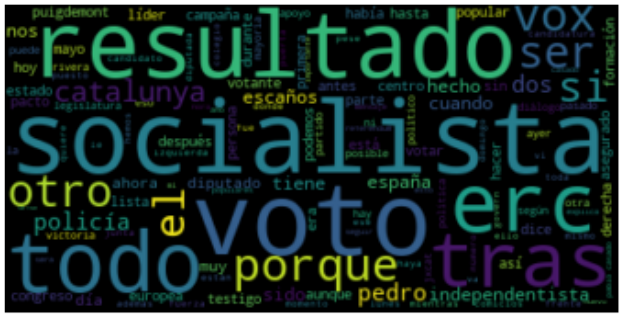
- Ciudadanos -	
escaños	46
podemos	46
vox	42
derecha	41
diputados	39
resultados	39
dice	39

Most used words

From the list of most used words for each party the first thing can be observed is that the classification of the articles has to be more discriminant because a lot of articles are being classified in the four categories. We notice this when we see that several words have the same or almost the same amount of repetitions for all parties or the counting of words is failing.
We can learn from the words listed in the previous slide the hot topics in Spanish politics:

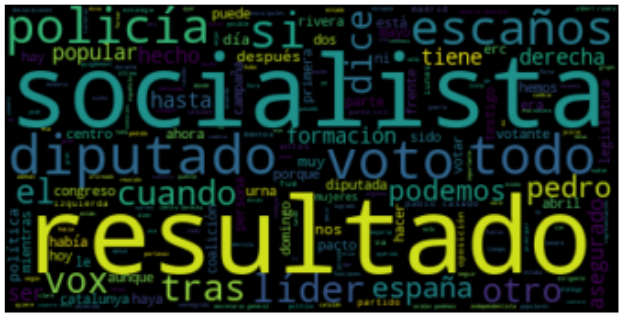
- The rise of alt-right party Vox
- The political conflict in Catalonia
- The elections

Word Clouds

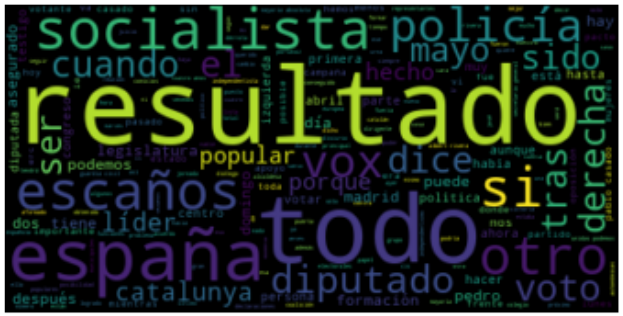


(a) PSOE

(b) PP



(c) Podemos



(d) Ciudadanos

Figure: Word Clouds after cleaning

Word Clouds

It catch our attention the fact that the words that appear in the wordclouds are not the same are not the same as the ones that are identified as most used words. This means that either the most used words list is not properly done or there is some bug.

Sentiment

When applying TextBlob to our Bag of Words we obtain the following output about for the subjectivity and polarity:

	party	text	polarity	subjectivity
0	PSOE	PSOE vice president functions socialist govern...	0.066758	0.404868
1	PP	PP the social democracy resists Iberian penins...	0.082358	0.414527
2	Cs	Cs the question repeated citizen leaders since...	0.086154	0.420359
3	Pdms	Pdms at this point the Spanish right should kn...	0.089888	0.430905

We see that TextBlob does not find any degree of polarity: less than 9% in all cases. The first conclusion would be that the source not polarized towards any particular position but I consider this a naive idea and my guessing is that the analysis has to be improved in order to gain some insight

Next steps

Next Steps

This was just and exercise and, hence, we did not devote a great amount of time to it. Nevertheless, there is a lot of growing potential to this idea.

If we want to improve this analysis the next steps that could be done are the following:

- Fix the issues mentioned previously like the lack of coherence between the wordclouds and the most used words list
- Do more web scraping to have a larger dataset and to compare the polarity of different sources
- Improve the classification
- Do lematization when preprocessing
- Get information from Twitter's API using hashtags
- Instead of applying TextBlob over the words separately, use n-grams
- Use more NLP techniques:
  - Word2Vect or TF-IDF instead of BoW
  - SpaCy, nltk, genism
  - ML based sentiment analysis

Backup

Function : Cleaner

```
import re
import string

def clean_text_round1(text):
    text = text.lower() #this makes text lowecase
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\w*\d\w*', '', text) #remove words that contain numbers
    text = re.sub('[‘’“”…]', '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\t', '', text)
    text = re.sub('\r', '', text)
    text = re.sub("[¡!@#$+%*:()'-]", ' ', text)
    text = re.sub('¿', '', text)
    return text
```

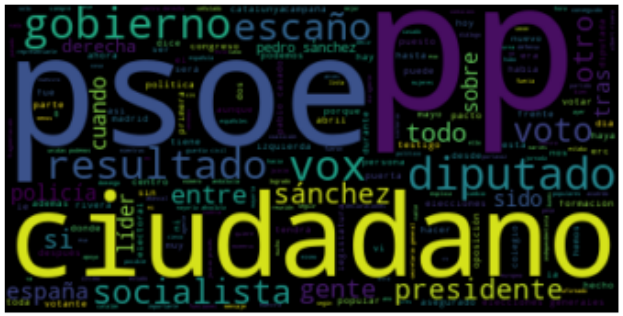
Function: Classifier

```
def clasifier(data_table):
  Global = {"PSOE" : [], "PP":[], "Cs":[], "Pdms":[]} # Reset dictionary
  lista_match_psoe = ['psoe','sánchez', 'socialistas', 'socialista', 'susana', 'puig', 'psc']
  lista_match_pp = ['pp', 'casado', 'populares', 'popular', 'rajoy', 'aznar']
  lista_match_pdms = ['podemos', 'unidas', 'iglesias', 'montero', 'carmena', 'morada']
  lista_match_cs = ['ciudadanos', 'rivera', 'arrimadas', 'naranja']

  lista_match_general = lista_match_psoe + lista_match_pp + lista_match_cs + lista_match_pdms

  for index, row in data_table.iterrows():
    #print(index, row)
    if any([match for match in lista_match_psoe if match in row['title']]):
      Global["PSOE"].append({"title": row['title'], "body": row['body']})
    if any([match for match in lista_match_pp if match in row['title']]):
      Global["PP"].append({"title": row['title'], "body": row['body']})
    if any([match for match in lista_match_cs if match in row['title']]):
      Global["Cs"].append({"title": row['title'], "body": row['body']})
    if any([match for match in lista_match_pdms if match in row['title']]):
      Global["Pdms"].append({"title": row['title'], "body": row['body']})
    if not any([match for match in lista_match_general if match in row['title']]):
      if any([match for match in lista_match_psoe if match in row['body']]):
        Global["PSOE"].append({"title": row['title'], "body": row['body']})
      if any([match for match in lista_match_pp if match in row['body']]):
        Global["PP"].append({"title": row['title'], "body": row['body']})
      if any([match for match in lista_match_cs if match in row['body']]):
        Global["Cs"].append({"title": row['title'], "body": row['body']})
      if any([match for match in lista_match_pdms if match in row['body']]):
        Global["Pdms"].append({"title": row['title'], "body": row['body']})
  print("Classified")
  return Global

Global = clasifier(data_table)
```

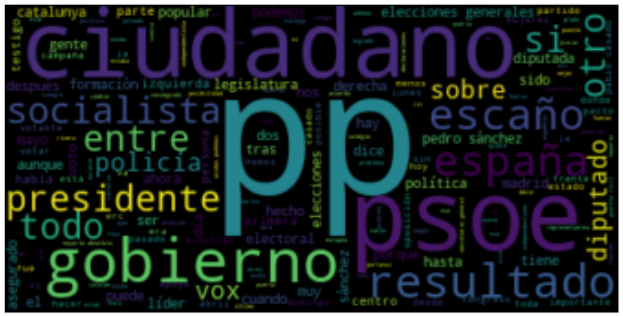
Word Clouds

(a) PSOE

(b) PP



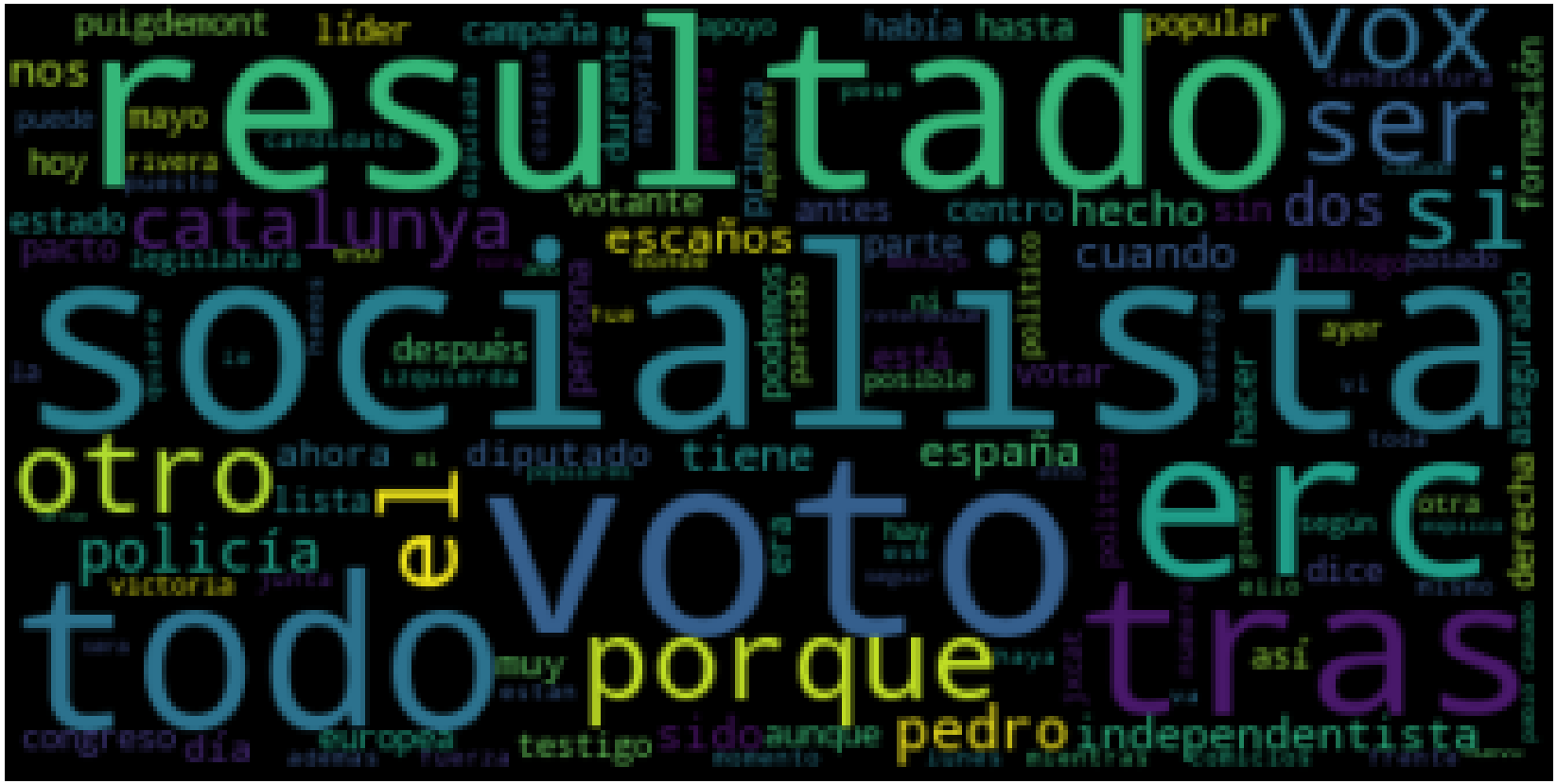
(c) Podemos



(d) Ciudadanos

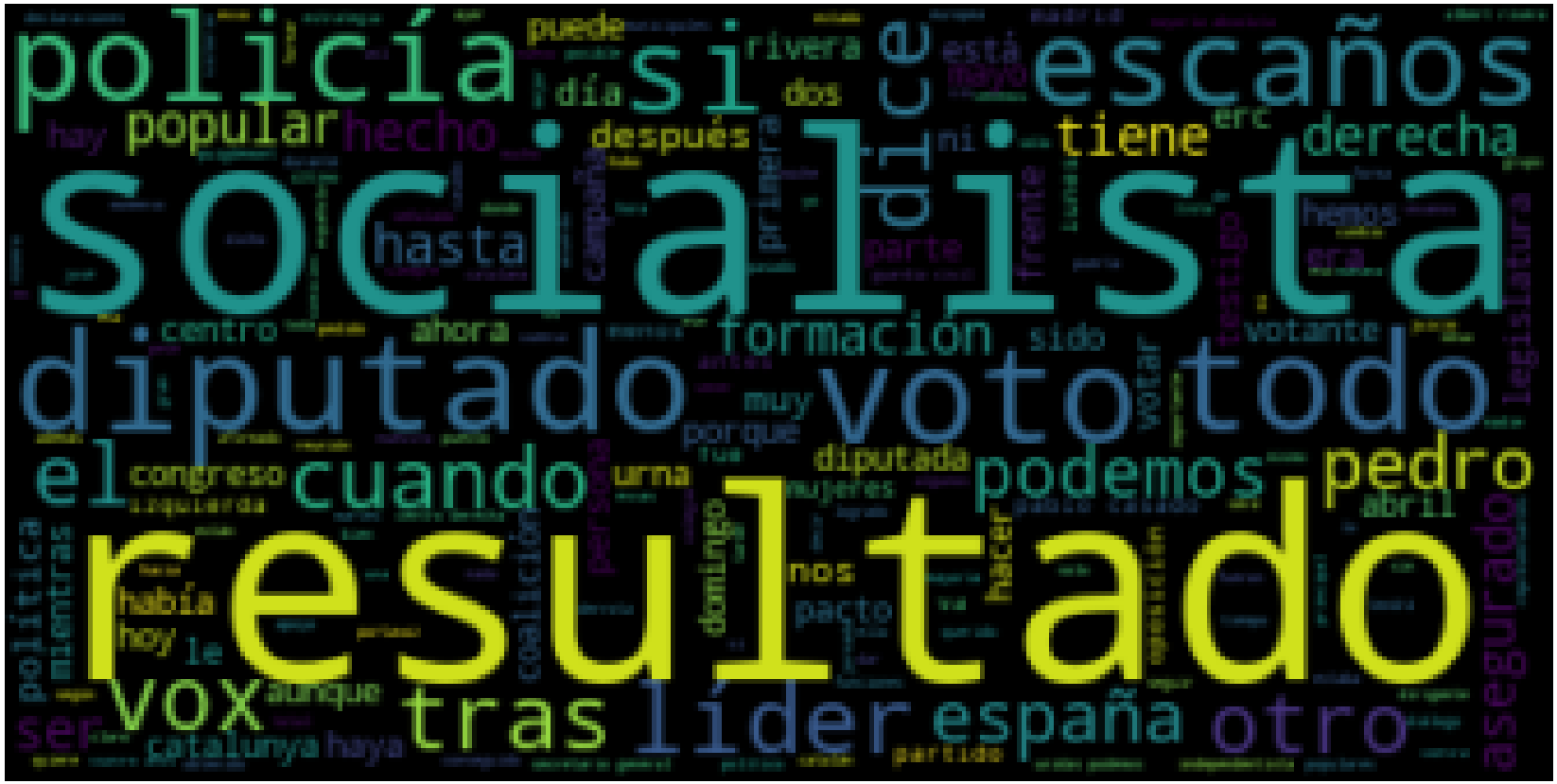
Figure: Word Clouds before cleaning round 3

Word Cloud - PSOE



Word Cloud - PP

Word Cloud - Podemos



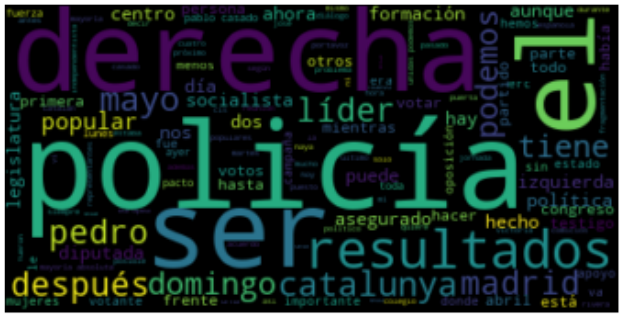
Word Cloud - Ciudadanos

Word Clouds - Deeper cleaning

(a) PSOE

(b) PP

(c) Podemos



(d) Ciudadanos

Figure: Word Clouds after deeper cleaning

Document Term Matrix

Index legend:

- 0 = PSOE
- 1 = PP
- 2 = Pdms
- 3 = C's

```
from sklearn.feature_extraction.text import CountVectorizer

#cv = CountVectorizer(stop_words='spanish') # This stop_words could be used as an alternative to clean_text_round2
cv = CountVectorizer()
data_cv = cv.fit_transform(data_clean.text)
data_dtm = pd.DataFrame(data_cv.toarray(), columns=cv.get_feature_names())
data_dtm.index = data_clean.index
data_dtm
```

	abajo	abanderada	abandonado	abandonar	abascal	abascalse	abc	abierta	abiertas	abierto	abiertos	abismo
0	2	1	3	1	6	1	1	2	2	3	2	3
1	1	0	3	1	7	1	1	1	0	1	1	3
2	2	0	2	1	5	1	0	2	2	2	2	0
3	2	0	3	1	5	1	0	2	2	2	2	2

Note: We can see that there are words without meaning (like "abascalse" in the 6th column) that may have appeared due to bugs when cleaning the text