# Textual Analysis - Greening finance for sustainable business

## Introduction

Our empirical project will be on the speech by Vice-President for the Euro and Social Dialogue, Financial Stability and Financial Services Valdis Dombrovskis, entitled *Greening finance for sustainable business* at Paris, on December 12, 2017.

The idea of this project is to set up an empirical method to study this discourse using different methods :

— Tokenization

— Substitution and cleaning

— Stop Word

— Lemmatization

— Word Cloud

To carry out this study, we will create a Python code, the useful parts of which will be indicated during the study or in the appendix.

## Objective

To carry out this study, we are going to use Python as mentioned above, as well as several libraries, such as *nltk*, *spacy*, *wordcloud* and *textblob*, which are libraries implementing Natural Language Processing solutions.

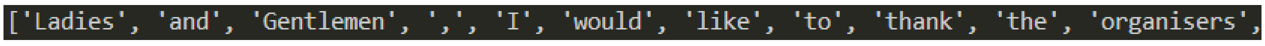
To get an initial idea of the content of the speech, we will create a word cloud. This is a visual representation of words where the size of each word is proportional to its frequency or importance in a data set. These word clouds are often used to summarise textual content in a concise and visually appealing way.

To create this word cloud, we first need to process the text.

## 1 Tokenization

First, we are going to tokenise the text. Tokenization refers to the process of breaking down the speech into individual units, or tokens, such as words, phrases, or sentences. These tokens are then used as the basic building blocks for further analysis, such as sentiment analysis, topic modeling, or natural language processing tasks. Tokenization is a crucial step in many text processing tasks, as it helps to organize and structure the data in a way that makes it easier for computers to understand and analyze.

To carry out this tokenization, we're going to use nltk's *word_tokenize* function, which directly separates words and punctuation elements. For example, for the start of the speech we get the following list start :



```
['Ladies', 'and', 'Gentlemen', ',', 'I', 'would', 'like', 'to', 'thank', 'the', 'organisers',
```

**Figure 1** – *Tokenization of the speech*

There are, of course, other methods of tokenisation or separation. In the remainder of this study, we will use a separation by sentence to study their polarities and subjectivities.

## 2 Substitution and cleaning

Once the tokenisation stage has been completed, we notice that the punctuation elements are not necessarily of interest to us, particularly the commas (the full stops used to separate sentences). What's more, we can put all the words in lower case in order to simplify the grouping of words and therefore calculate frequencies of occurrence, our objective in creating a word cloud. This processing corresponds to substitution and cleaning.

The text originally contained 5992 characters, and after deleting commas, colons, semicolons and hyphens, it now contains 5925.

```
text = text.replace(',','')
text = text.replace(':','')
text = text.replace(';','')
text = text.replace('-','')
text = text.lower()
```

**Code 1** – *Code to replace ,/ :/;/-*

We do this before tokenisation so that it can benefit from our modifications.

## 3 Stop Word

Stop words are common words that are often filtered out during textual analysis because they are considered to be of little value in understanding the meaning of the text. These words include frequently used words such as "the," "and," "of," "is," "in," "to," etc.

Removing stop words can help in improving the efficiency of natural language processing tasks such as text classification, sentiment analysis, and topic modeling by reducing the noise in the data and focusing on more meaningful words. However, the list of stop words can vary depending on the specific context or language being analyzed.

The easiest way to remove these stop words is to find lists adapted to the language used and remove the words contained in this list. In our case, we use the list provided by *nltk.corpus*.

```
stop_words = set(stopwords.words('english'))
print(stop_words)
token_stop_words = [word for word in tokens if word not in stop_words]
print(token_stop_words)
```

**Code 2** – *Code to delete stop words from the tokenization*

It is then interesting to note that we go from 1019 elements (words, punctuation elements, etc.) to 585, which represents 57% of the elements.

## 4 Lemmatization

Lemmatization is the process of reducing words to their base or canonical form, known as the lemma, to normalize variations of the same word. Unlike stemming, which chops off prefixes or suffixes to find the root word, lemmatization considers the context and morphology of the word to ensure that the resulting lemma is a valid word. For example, the lemma of *went*, *gone*, *going* is *go* and the lemma of *better* is *good*.

Lemmatization is commonly used in natural language processing tasks to improve the accuracy of text analysis, such as text classification, sentiment analysis, and information retrieval, by treating different inflected forms of words as the same entity.

```
nlp = spacy.load('en_core_web_sm')
text_filtrated = " ".join(token_stop_words)
lemmatized_tokens = [token.lemma_ for token in nlp(text_filtrated)]
print(f"{lemmatized_tokens = }")
```

**Code 3** – *Lemmatization of the filtrated text*

## 5 Word Cloud

Finally, after these various processes, we can now create the word cloud. The library used for this is *wordcloud*. It can be used to create a word cloud directly from a text, which can then be displayed using a library such as *matplotlib*.

```
wordcloud = WordCloud(background_color = 'white', max_words = 50).generate(lemmatization_string)
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

**Code 4** – *Creation of the Word Cloud*

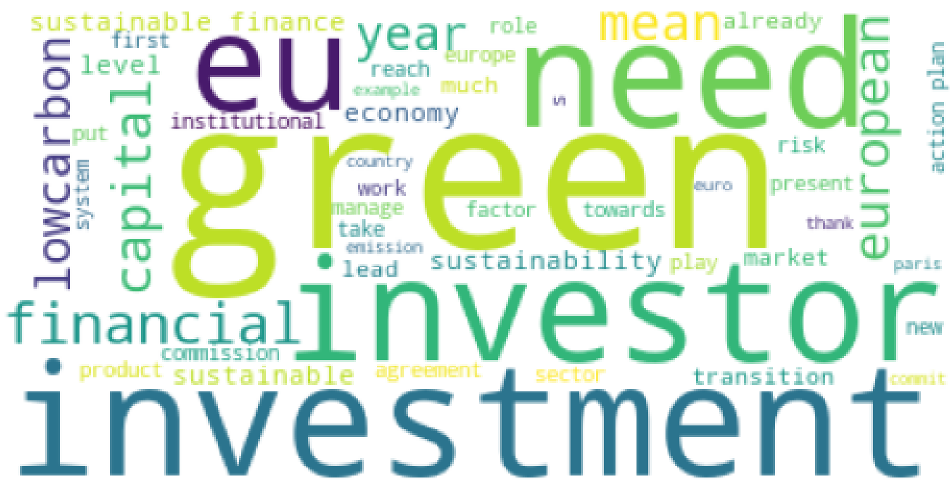


**Figure 2** – *Word Cloud*

This produces the desired Word Cloud. This can then be used to analyse the speech made by Valdis Dombrovskis. The key elements are *green, EU, investment/investor*. This is directly linked to the title of the speech, *Greening finance for sustainable business*, which is about sustainable investment in Europe.

From a processing point of view, we note that words such as *Investor* and *Investment* could have been grouped together, as could *EU* and *European* or *Sustainable* and *Sustainability*. It may be important to note that libraries specialising in textual analysis often implement the processing directly, so the steps may not be obligatory. For example, if we create a word cloud using the same *wordcloud* library without any prior processing, we obtain :

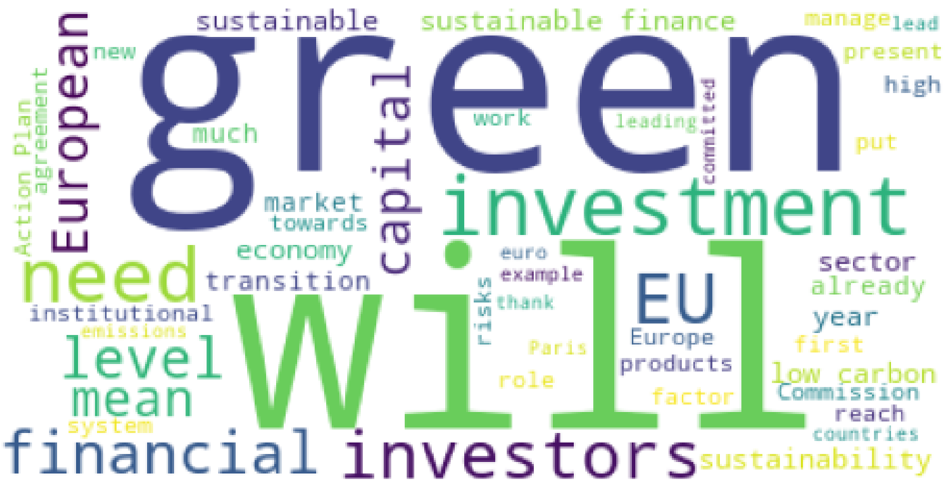


**Figure 3** – *Word Cloud without any treatment (except from the library)*

The result is very similar : the library performs the same processing, with a few differences, such as the stop word lists used are not the same.

## 6 Sentiment

In this section, we will be looking at the tone used and the feelings conveyed by the text. To do this, we'll use the *TextBlob* library, which enables us to analyse the feelings conveyed by a sentence or text. The sentiment property returns 2 values, the polarity and the subjectivity. The polarity score is a float within the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective.

To illustrate this function, let's take the sentence : *I love this amazing speech !*. We find a polarity of 0.625 and a subjectivity of 0.75. Similarly, with the sentence : *I hate this speech !*, we find a polarity of -1 and a subjectivity of 0.9.

```
text_test = "I love this amazing speech!"
blob_test = TextBlob(text_test)
print(blob_test.sentiment)
```

**Code 5** – *Usage of sentiment in the TextBlob library*

Now let's apply this function to our text. For the text as a whole, we find a polarity of 0.08 and a subjectivity of 0.33. The tone of the text is therefore fairly neutral, providing factual information and proposals for improving the sustainability of finance in Europe.

More specifically, we're going to look at the polarity and subjectivity of different sentences to see how these 2 feelings are expressed.

Here are some examples of sentences whose subjectivity is equal to 1 :

— *170 countries have ratified the Paris agreement by now, which sends a powerful signal that the low-carbon transition is here to stay.*

— *And I'm pleased that countries like Sweden, France and Luxembourg already have rules and labels to attract sustainable investment.*

— *This is a necessary condition for sustainable finance to reach scale.*

And here are some sentences with high polarity :

— *And we see that this is already happening, so that is good news.* (0.7)

— *And I'm pleased that countries like Sweden, France and Luxembourg already have rules and labels to attract sustainable investment.* (0.5)

— *But for our effort to succeed, we need more than a piece by piece, sector-by-sector approach.* (0.5)

## Annexe

Link to the speech : `https://ec.europa.eu/commission/presscorner/detail/fr/SPEECH_17_5235`

Github : `test.com`

NLTK : `https://www.nltk.org/`

TextBlob : `https://textblob.readthedocs.io/en/dev/`

spaCy : `https://spacy.io/`