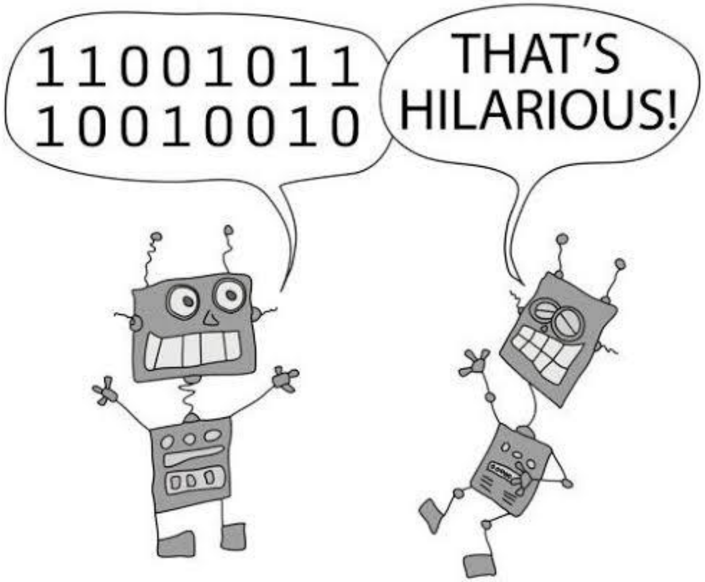
Text Data Vectorization – From Text to Numbers

SOFIA UNIVERSITY
ST. KLIMENT OHRIDSKI
EST. 1888

FACULTY OF ECONOMICS AND BUSINESS ADMINISTRATION

What you will learn

- What is **Text Vectorization**?
- Why do we need it?
- **Key techniques** for text data vectorization.
- How to **apply** them in **Python**?
- More advanced techniques for text data vectorization.



1.

From Text… to Numbers

Image credits: Can a Robot Make You Laugh? — Teaching an AI to Tell Jokes | by Lorenzo Ampil | Towards Data Science

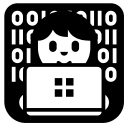
Text Vectorization (1)

- Formal definition of **text vectorization**:

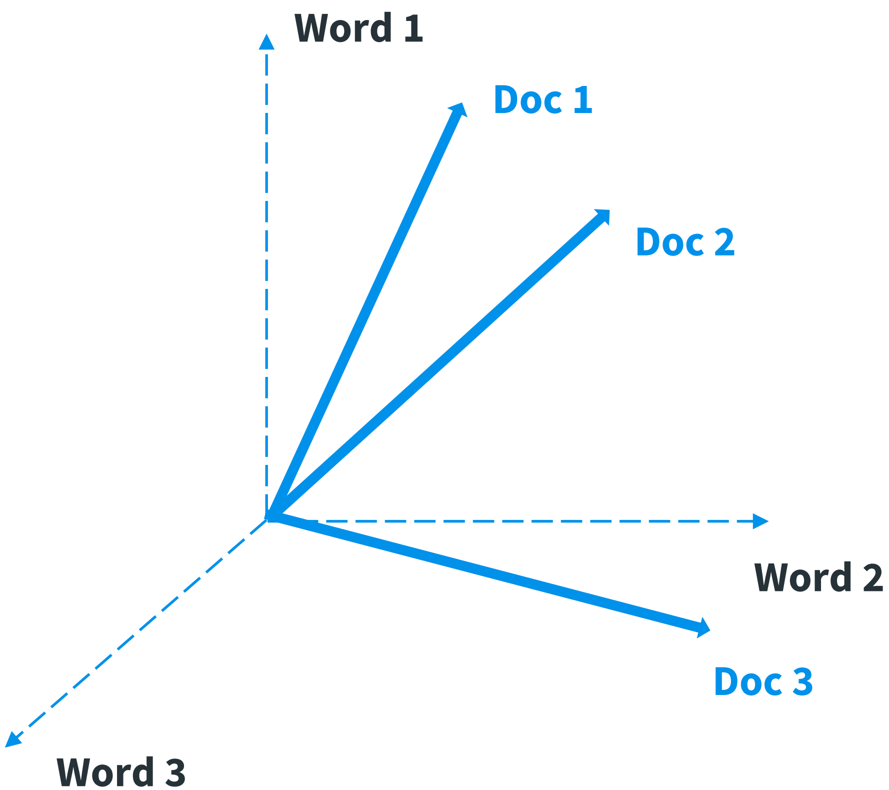
“

*The process of converting textual data into numerical form is called ‘**text vectorization**’. In this way, words, sentences or whole documents in a corpus are represented as vectors of numerical values.*

”



Text Vectorization (2)



Vocabulary = [Word 1, Word 2, Word 3]

Text Vectorization (3)

- Quantitative analysis **can’t** be applied without text vectorization!
- What **form of text vectorization** to apply depends on the problem at hand and the chosen algorithm for data analysis.
- Text vectorization might **highly impact** the results of the analysis!
- There are **basic and more complex** methods for text vectorization.

The Vector Space Model (1)

In the vector space model each document $\boldsymbol{d_i}$ is represented as a vector of weights $\boldsymbol{u_{ij}}$:

$$\boldsymbol{d_i} = (\boldsymbol{u_{i1}}, \boldsymbol{u_{i2}} \dots, \boldsymbol{u_{iv}}), \text{ where}$$

$\boldsymbol{D}$ – the sample of text data.

$\boldsymbol{M}$ – the number of documents in sample $\boldsymbol{D}$.

$\boldsymbol{d_i}$ – a given document in $\boldsymbol{D}$, $i = 1 \dots M$.

$\boldsymbol{w_j}$ – a given token part of the vocabulary, $j = 1 \dots \mathrm{V}$.

$\boldsymbol{V}$ – number of unique tokens in the sample. The unique tokens form the "**vocabulary**" of the model.

$\boldsymbol{u_{ij}}$ – the weight of the j-th token $\boldsymbol{w_j}$ in the document $\boldsymbol{d_i}$, where ($i = 1 \dots \mathrm{M}$) and ($j = 1 \dots \mathrm{V}$).

The Vector Space Model (2)

Input Text:
"the plot was good"

Vocabulary:
the
plot
was
very
good
boring

Vector Representation:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

The Bag-of-Words assumption

- The **bag-of-words assumption** conveys the idea of a text representation that disregards the order of words.
- The vector-space model makes the bag-of-words assumption - **order of words in text documents is not taken into account.**
- According to this assumption “Mary is quicker than John” **is the same as** “John is quicker than Mary” (Manning, 2008).
- Despite the “unrealistic” nature of this assumption, it proves to **work well in many practical text mining tasks**.
- Making this assumption we say: **“word order is not important but I expect that documents with similar representation share similar content”.**

Forms of Vectorization in the Vector Space Model

There are **three forms of text vectorization** in the Vector Space Model:

- Binary
- Count (absolute or relative)
- TF-IDF (term frequency – inverse document frequency)

Binary Vectorization

- Binary vectorization takes into account only the absence or presence of a given token. The **presence** is denoted with **1**, while the **absence** is denoted with **0**.

“the movie is ok”
“the beginning was boring…very boring”
“the plot was very good”

Our vocabulary

	“beginning”	“boring”	“good”	“is”	“movie”	“ok”	“plot”	“the”	“very”	“was”
Review 1	0	0	0	1	1	1	0	1	0	0
Review 2	1	1	0	0	0	0	0	1	1	1
Review 3	0	0	1	0	0	0	1	1	1	1

Count Vectorization (absolute)

- Count vectorization takes into account the **frequency of tokens** in the documents.
- Example of count vectorization with **absolute values:**

"the movie is ok"
"the beginning was boring…very boring"
"the plot was very good"

	"beginning"	"boring"	"good"	"is"	"movie"	"ok"	"plot"	"the"	"very"	"was"
Review 1	0	0	0	1	1	1	0	1	0	0
Review 2	1	2	0	0	0	0	0	1	1	1
Review 3	0	0	1	0	0	0	1	1	1	1

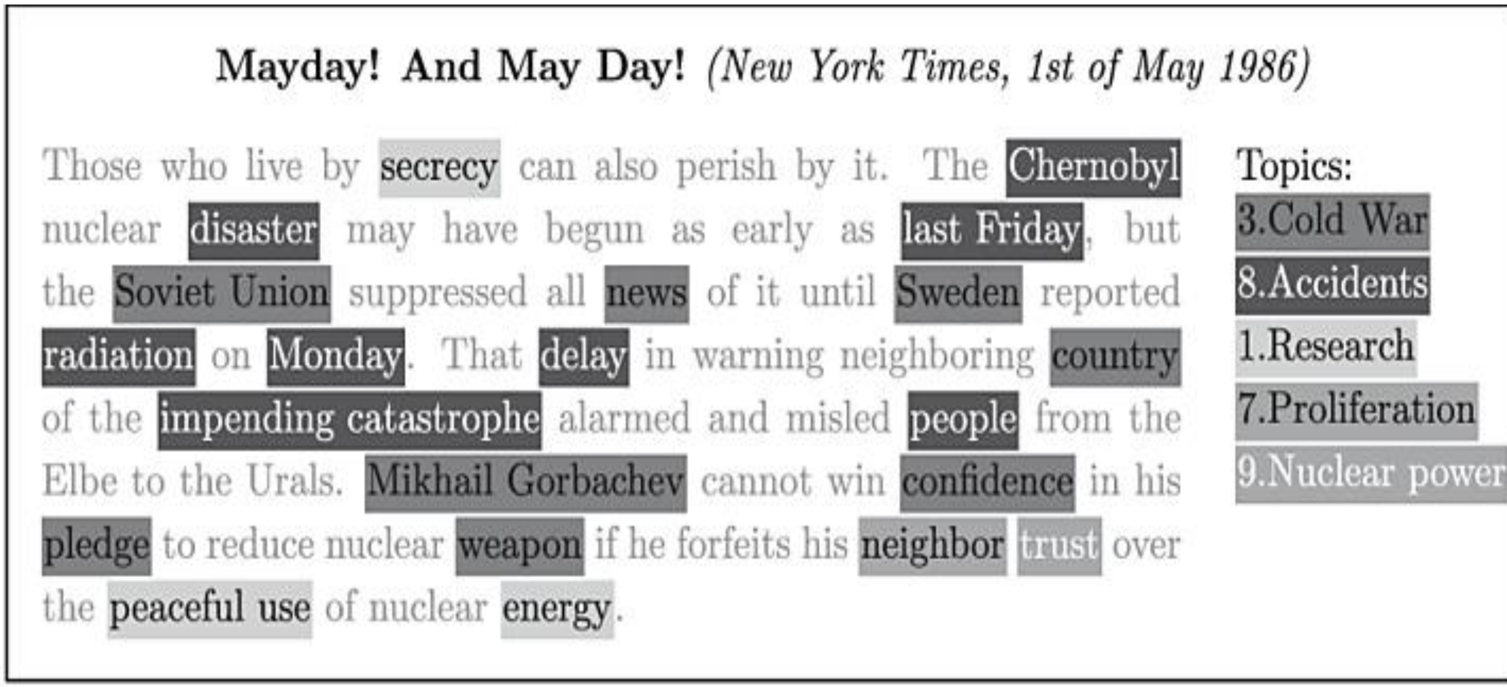
Count Vectorization (relative)

- In relative count vectorization, the **counts are normalized** by the number of words in the document.
- Example of relative count vectorization:

“the movie is ok”
“the beginning was boring…very boring”
“the plot was very good”

	“beginning”	“boring”	“good”	“is”	“movie”	“ok”	“plot”	“the”	“very”	“was”
Review 1	0	0	0	0.25	0.25	0.25	0	0.25	0	0
Review 2	0.16	0.33	0	0	0	0	0	0.16	0.16	0.16
Review 3	0	0	0.2	0	0	0	0.2	0.2	0.2	0.2

Keywords are important, right?


Mayday! And May Day! *(New York Times, 1st of May 1986)*

Those who live by secrecy can also perish by it. The Chernobyl nuclear disaster may have begun as early as last Friday, but the Soviet Union suppressed all news of it until Sweden reported radiation on Monday. That delay in warning neighboring country of the impending catastrophe alarmed and misled people from the Elbe to the Urals. Mikhail Gorbachev cannot win confidence in his pledge to reduce nuclear weapon if he forfeits his neighbor trust over the peaceful use of nuclear energy.

Topics:
3.Cold War
8.Accidents
1.Research
7.Proliferation
9.Nuclear power

…TF-IDF vectorization?

Image credits: https://www.researchgate.net/figure/Example-article-with-words-from-different-topics-highlighted-in-the-text_fig1_283671339

TF-IDF Vectorization

Term frequency

Inverse-document frequency

$$\boldsymbol{TF\text{–}IDF}(\boldsymbol{w_n}, \boldsymbol{d_i}) = \boldsymbol{tf_{w_n,d_i}} \times \boldsymbol{idf_{w_n}},\ where:$$

$\boldsymbol{D}$ – *the sample of text data.*
$\boldsymbol{M}$ – *the number of documents in sample* $\boldsymbol{D}$.
$\boldsymbol{d_i}$ – *a given document in* $\boldsymbol{D}$, $i = 1 \dots M$.
$\boldsymbol{w}_n$ – n^{th} *word in document* $\boldsymbol{d_i}$, $n \in \{1, \dots, N_d\}$.

$$df_{w_n} = |\{d_i \in D: w_n \in d_i\}|$$

$$idf_{w_n} = log\,\frac{M}{df_{w_n}}$$

$$TF\text{–}IDF(w_n, d_i) = tf_{w_n,d_i} \times\ log\,\frac{M}{df_{w_n}}$$

TF-IDF Vectorization in scikit-learn

- The implementation in scikit-learn uses the following formula for calculating TF-IDF:

tf_{w_n,d_i} - number of times the word occurs in the given document

df_{w_n} - number of documents in the corpus containing the word

$$idf_{w_n} = ln\frac{1+M}{1+\ df_{w_n}} + 1$$

$$\boldsymbol{TF\text{–}IDF(w_n, d_i) = \ tf_{w_n,d_i} \times idf_{w_n}}$$

NB! L2 normalization is applied on the vectors of TF-IDF values (sum of squares =1).

$$Original\ vector:\ x = [x_1 \quad x_2 \dots \quad x_V]$$

$$y_1 = \frac{x_1}{\sqrt{\sum_{i=1}^{V} {x_i}^2}}$$

$$L2\ normalized\ vector: y = [y_1 \quad y_2 \dots \quad y_V], y_i \in [0,1]$$

TF-IDF Calculation (1)

"the movie is ok"
"the beginning was boring…very boring"
"the plot was very good"

Calculate TF-IDF values for the first review:

"the" $\Longrightarrow TF = 1, DF = 3 \Longrightarrow idf = \ln\frac{1+3}{1+3} + 1 = 1 \Longrightarrow tf \times idf = 1 \times 1 = 1$

"movie" $\Longrightarrow TF = 1, DF = 1 \Longrightarrow idf = \ln\frac{1+3}{1+1} + 1 = 1.693 \Longrightarrow tf \times idf = 1 \times 1.693 = 1.693$

"is" $\Longrightarrow TF = 1, DF = 1 \Longrightarrow idf = \ln\frac{1+3}{1+1} + 1 = 1.693 \Longrightarrow tf \times idf = 1 \times 1.693 = 1.693$

"ok" $\Longrightarrow TF = 1, DF = 1 \Longrightarrow idf = \ln\frac{1+3}{1+1} + 1 = 1.693 \Longrightarrow tf \times idf = 1 \times 1.693 = 1.693$

First review before normalization:

$$[0 \quad 0 \quad 0 \quad 1.693 \quad 1.693 \quad 1.693 \quad 0 \quad 1 \quad 0 \quad 0]$$

TF-IDF Calculation (2)

"the movie is ok"
"the beginning was boring…very boring"
"the plot was very good"

By default in scikit-learn L2 normalization is applied on the TF-IDF weights (recommended approach):

First review:

"the" $\Longrightarrow$ TF-IDF= $1 \Longrightarrow \frac{1}{\sqrt{1^2+1.693^2+1.693^2+1.693^2}} = \frac{1}{3.098} \sim 0.3227$

"movie" $\Longrightarrow$ TF-IDF= $1.693 \Longrightarrow \frac{1.693}{\sqrt{1^2+1.693^2+1.693^2+1.693^2}} = \frac{1.693}{3.098} \sim 0.5464$

"is" $\Longrightarrow$ $\mathrm{TF{-}IDF} = 1.693 \Longrightarrow \frac{1.693}{\sqrt{1^2+1.693^2+1.693^2+1.693^2}} = \frac{1.693}{3.098} \sim 0.5464$

"ok" $\Longrightarrow$ $\mathrm{TF{-}IDF} = 1.693 \Longrightarrow \frac{1.693}{\sqrt{1^2+1.693^2+1.693^2+1.693^2}} = \frac{1.693}{3.098} \sim 0.5464$

First review after normalization:

$$[0 \quad 0 \quad 0 \quad \mathbf{0.5464} \quad \mathbf{0.5464} \quad \mathbf{0.5464} \quad 0 \quad \mathbf{0.3227} \quad 0 \quad 0]$$

TF-IDF Calculation (3)

"the movie is ok"
"the beginning was boring…very boring"
"the plot was very good"

Final matrix of TF-IDF values calculated in scikit-learn:

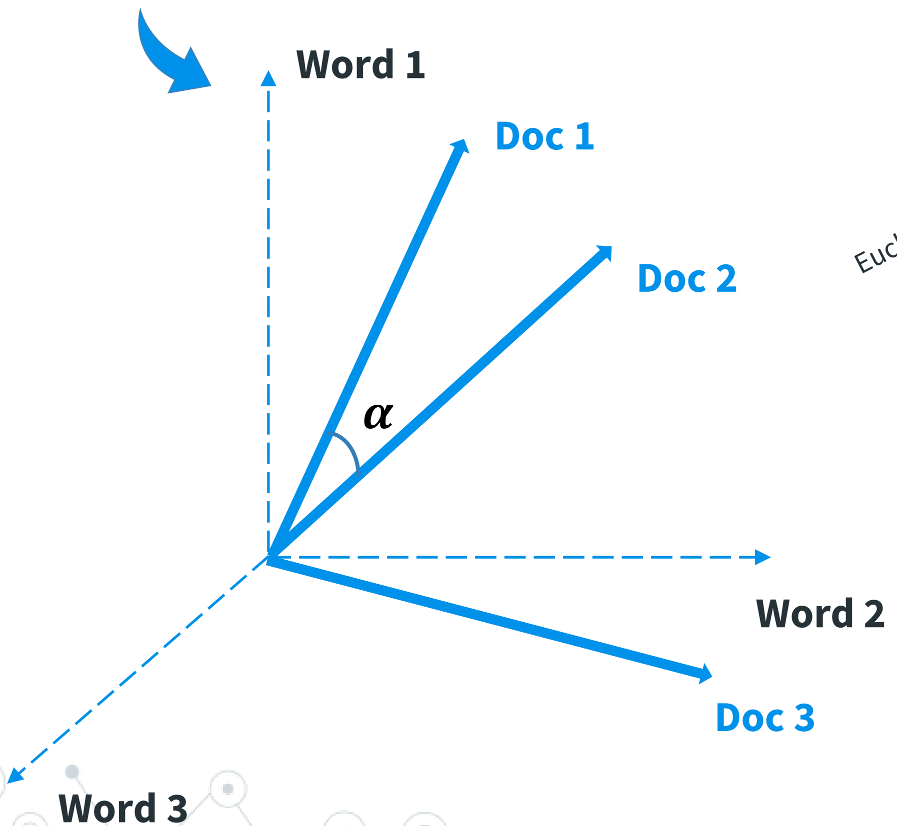
	"beginning"	"boring"	"good"	"is"	"movie"	"ok"	"plot"	"the"	"very"	"was"
Review 1	0	0	0	0.5464	0.5464	0.5464	0	0.3227	0	0
Review 2	0.3920	0.7841	0	0	0	0	0	0.2315	0.2981	0.2981
Review 3	0	0	0.5340	0	0	0	0.5340	0.3154	0.4061	0.4061

Practical Applications of the Vector Space Model

- **Text classification/clustering** (sentiment analysis, topic modeling etc.);
- Fundamental in **Information Retrieval** – serves as a basis for search engines development;
- **Recommender systems** – for movies, books, news articles etc.
- And many other.

Find Text Similarity with TF-IDF

Vocabulary = [Word 1, Word 2, Word 3]



Cosine similarity:

$$\mathrm{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|}$$

Euclidean vector length

$$|\vec{V}(d_1)| = \sqrt{\sum_{i=1}^{V} \vec{V}_i^2(d_1)}$$

$$\frac{\vec{V}(d_1)}{|\vec{V}(d_1)|} = \frac{\vec{V}(d_1)}{\sqrt{\sum_{i=1}^{V} \vec{V}_i^2(d_1)}} = \vec{v}(d_1)$$

$$\mathrm{sim}(d_1, d_2) = \vec{v}(d_1) \cdot \vec{v}(d_2)$$

$$\mathrm{sim}(d_1, d_2) = \cos \boldsymbol{\alpha}$$

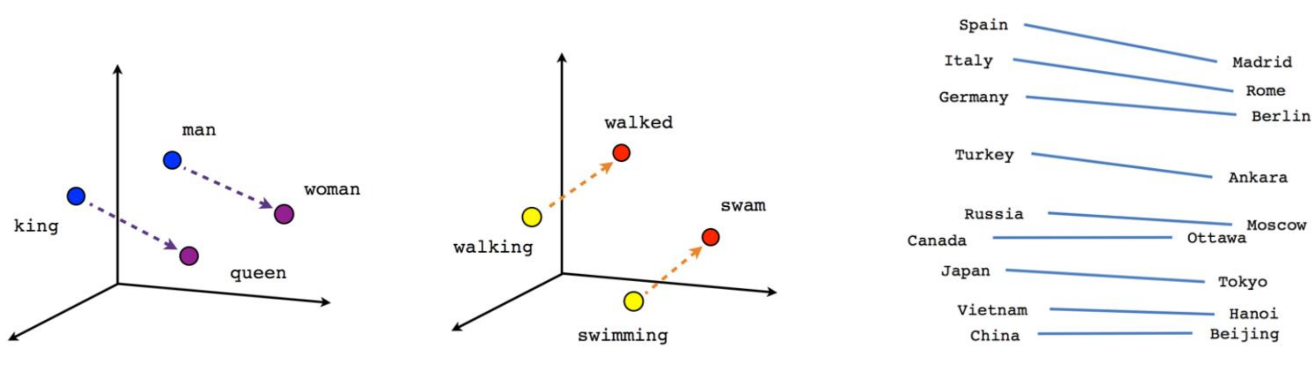
How to implement all these techniques in Python?

Use scikit-learn (https://scikit-learn.org/stable/index.html):

- Binary vectorization – CountVectorizer(binary= True)
- Count Vectorization - CountVectorizer()
- TF-IDF Vectorization – TfidfVectorizer()
- Cosine similarity - cosine_similarity()

More complex methods for text vectorization

- Word Embeddings – word2vec (2013), doc2vec, GloVe and other.
- An upgrade in 2018 - ELMo and BERT



Male-Female | Verb tense | Country-Capital

„You shall know a word by the company it keeps...” (Firth, J. R. 1957:11)

Image credits: [Mapping Word Embeddings with Word2vec | by Sam Liebman | Towards Data Science]

If you want to learn more… 🤓

- Introduction to Information Retrieval (stanford.edu) – p.154 - p.161 (pdf)
- Miner et al., "**Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications**" - Chapter 3

Thanks!

Any questions?

You can find me at:

g.hristova@feb.uni-sofia.bg