Machine Learning for Social Sciences
Part 4: Text as Data

Felix Hagemeister

TUM School of Social Sciences and Technology
April 2022

Agenda

1. Quanteda Quanteda, Reading in Text, Preprocessing, Document Frequency Matrices
2. Zipf's Law, key-word analysis, cosine similarity, dictionary methods, topic models
3. Text Regression, MNIR
4. Naive Bayes

Motivation

Why should a social scientist study text data?

- Text can cover socially relevant information (e.g. Central Bank statements, product reviews, parliamentary speeches).
- Many traditional questions in social sciences can be answered better with text data (e.g. how does the media influence people's decisions?).
- Large digitized text data sets are now available.
- Many recent high-level publications use text data.

Text as Data

Wait! How can a text be represented as data?

- Solution: every word is a vector.
- **Bag of words** simplifying assumption: the order of words is irrelevant.
- For example:

  Document1: "Munich is nice, really nice."

  Document2: "Flowers are nice."

Pre-processing

Pre-processing of text: not every word is important

- **Punctuation:** , . ; : - “ ” ...
- **Stopwords:** are, is, really, actually, he, she, it...
- **Capitalization**
- **Numbers**
- **Special characters**

1.4 Document-Frequency-Matrix

Text can be represented in a **document-frequency-matrix (dfm)**:

	Munich	Flowers	nice
Document1	1	0	2
Document2	0	1	1

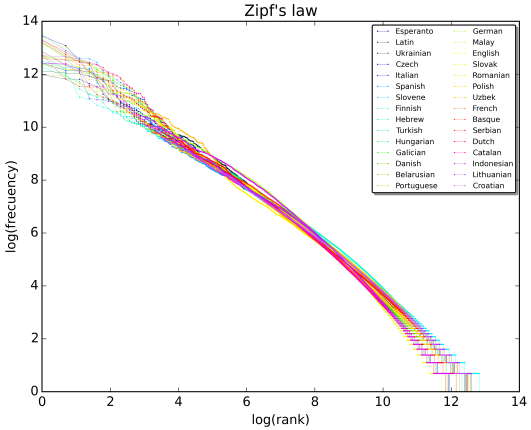
- As every word is a variable, this is *high-dimensional data*.
- One term is called a **token** or a **feature**.

Zipf's Law

How are the terms distributed in the data?

- We can look at the most frequent words
- **Zipf's Law:** Frequency decreases rapidly with rank.

Zipf's Law



Source: Wikimedia Commons

Zipf's Law

The underlying concept is called **sparsity**. A sparse matrix is a matrix with many zeros. In our document-feature-matrix, we usually encounter many zeros as

- Some few words are used (quite often) by all documents: e.g. der, die, das, er, sie, es
- Many words are only used in very few documents: e.g. "Asylbetrug" in the AfD's manifesto.

This leads to the fact that the share zero entries in the **dfm** is usually quite high. This share is called **sparsity**.

Tf-IDF

TF-IDF scaling.

- **TF(t)**: Term frequency =
  (Number of times term t appears in a document) / (Total number of terms in the document)
- **IDF(t)**: Inverse document frequency =
  log(Total number of documents / Number of documents with term t in it)
- **TF-IDF(t)**: TF(t) * IDF(t)

Note that TF can alternatively be defined as $1 + \log$ count.

TF-IDF Example

Consider a document with 100 words where the word “car” features 3 times. Let’s suppose our corpus consists of 10 million documents and “car” appears in one thousand of these.

- **TF(car)**: $3/100 = 0.03$
- **IDF(car)**: $\log(10{,}000{,}000 \ / \ 1{,}000) = 4$
- **TF-IDF(car)**: $0.03 * 4 = 0.12$

TF-IDF is a measure of importance of a certain word for both a given document and vis-a-vis the broader corpus.

Zipf's Law

We often don't care about those words which lead to highly sparse matrices. That is why

- we remove very frequent and less meaningfull words (**stopwords**)
- sometimes trim our **dfm** to contain only words which are at least present in some documents.

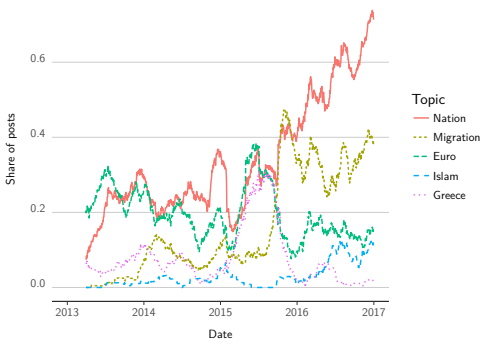
Exercise

Let's read in the 2013-2021 election manifestos of the main German political parties and do some preprocessing:

Here is the code.

Keywords

A simple analysis of text is an illustration of certain key-words. An example: Facebook posts from the German AfD party.



Source: Cantoni, Hagemeister, and Westcott (2019)

Keywords

As social scientists, we try to go beyond simple visualisation of data and try to uncover relationships. We use regression analysis to rule out alternative explanations. For example:

$$f(\text{stem} = s)_{ipt} = \gamma_p + \delta_t + \beta \cdot \mathbb{1}\{\text{party} = \text{AfD}\} \cdot \text{Post}_t + \varepsilon_{ipt}, \tag{1}$$

where the dependent variable $f(\text{stem} = s)$ is the frequency (mention per 100 words) of stem s in document i (party manifesto, speech), of party p at time t.

Source: Cantoni, Hagemeister, and Westcott (2019)

Keywords

Table 1: AfD's Language Change: Diff-in-diff Estimates

	(1) Greece	(2) Euro	(3) Islam	(4) Migration	(5) Nation
PANEL A: *Mentions per 100 words in manifestos*					
AfD × After March 2015	-0.011	-0.780***	0.052***	0.269***	-0.041
	(0.021)	(0.193)	(0.013)	(0.050)	(0.237)
PANEL B: *Mentions per 100 words in speeches*					
AfD × After March 2015	-0.183**	-0.546***	0.063*	-0.028	0.112
	(0.070)	(0.099)	(0.034)	(0.097)	(0.100)
PANEL C: *Mentioned in Twitter posts*					
AfD × After March 2015	-0.059***	-0.157***	0.020**	0.023**	-0.098***
	(0.009)	(0.011)	(0.009)	(0.012)	(0.018)
PANEL D: *Mentioned in Facebook posts*					
AfD × After March 2015	-0.017	-0.055***	0.042***	0.112***	0.209***
	(0.016)	(0.021)	(0.011)	(0.023)	(0.030)

Similarity

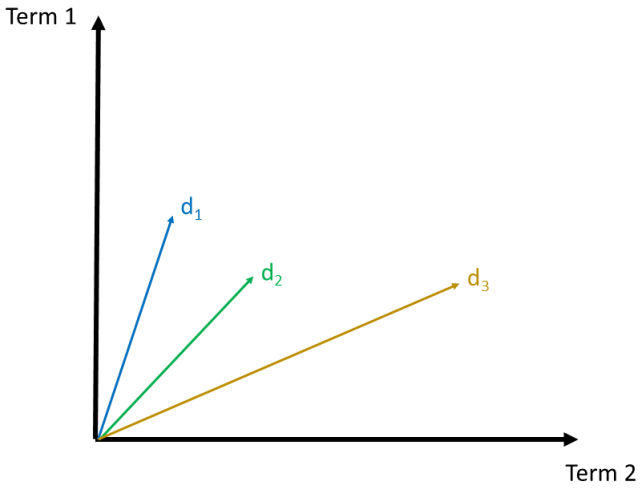
What do we need TF-IDF for? A very common application is **document similarity**.

Example: Consider three political party manifestos:

1. "The economy and jobs are important."
2. "Jobs and the environment are important."
3. "The environment and culture are important, especially the environment."

How similar are these manifestos to each other? Why?

Documents as vectors



2.3 Similarity

The most common approach to calculate document similarity is **cosine similarity**.

The cosine similarity between two documents A and B is defined as

$$\frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2 \sum_{i=1}^{n} b_i^2}}$$

Where a_i and b_i are (tf-idf weighted) counts of word i in documents A and B.

Dictionary Methods

A dictionary is a collection of words, usually related to a certain topic. Dictionary methods count the (relative) frequency of occurrence of these words in documents.

Example: Three parties and dictionary {Environment, Clima, Green, Temperature}.

1. "The economy and jobs are important."
2. "Jobs and the environment are important."
3. "The environment and culture are important, especially the environment."

–> Party 1: 0

–> Party 2: 1

–> Party 3: 2

Dictionary Methods

In addition, dictionaries usually assign weights to each word they contain:

Example:

Word	Weight
Environment	1
Green	0.7
Clima	0.6
Jobs	-0.2
Economy	-0.4

What are the scores in terms of this dictionary for our three parties here?

Dictionary Methods

Solution:

–> Party 1: -0.6

–> Party 2: 0.8

–> Party 3: 2

Dictionary Methods

Many dictionaries have been written for many different topics, including in different languages.

Here is a good source for Dictionaries in the German language:

- https://sites.google.com/site/iggsahome/downloads

Quanteda has some built-in dictionaries that we will use in the following exercise.

Exercise

Let's read in the 2013-2021 manifestos of German political parties, perform preprocessing steps, and create a DFM with this code and this text data.

Topic Models

With dictionary methods, we can make statements about documents regarding a specific aspect or "topic" (usually positive/negative sentiment).

- **Topic Models** allow us to divide documents into different topics.
- One needs to choose the numbers of topics.
- The documents are assigned to topics by the structure of the data ("machine learning").

Topic Models

The most common topic model algorithms are

- **K-means clustering** partitions the documents into k topics (exclusively).
- **LDA** (Latent Dirichlet Allocation) assigns to each document a score for each topic.

PCA (also called LSA) was common before the 2000s.

Text Regression

Under the assumption that the usage of certain terms signals some interesting outcome (e.g. political ideology), we can build an index of *slant* that sums across a speaker's term usage weighted by the direction of slant of each term.

How-to:

1. Build DFM.
2. Might transform DFM using tf or tf-idf weights, or use one-hot encoding.
3. PCR regress, or better LDA-topic model regress outcome on DFM
4. Alternatively, run a lasso regression
5. Use k-fold cross validation or IC to find best model
6. investigate the terms/phrases most indicative of the outcome

Exercise

Let's go through Matt Taddy's code using this file.