Workshop ‘Computational Text Analysis’

Session 2: Bags-of-words

Mirko Wegemann

25 March 2025

Goals of this workshop

1. Automation of data collection
2. **Analysis of textual data**
   - unsupervised approaches (e.g., topic models)
   - supervised approaches (e.g. text classification)
3. Analysis of images-as-data

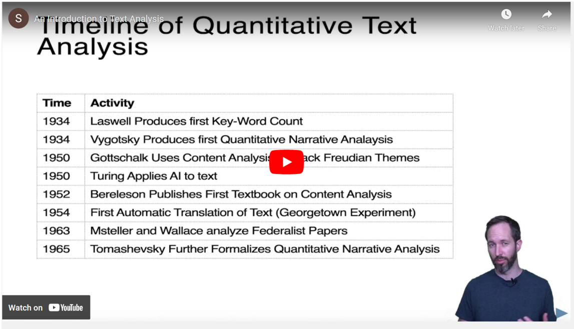
What is computational text analysis? I

"Computational text analysis (also called Quantitative Text Analysis, Automated Content Analysis, Text Mining, Text as Data etc.) draws on techniques developed in natural language processing and machine learning to analyse textual documents." (Chun-Ting Ho 2021)

What is computational text analysis? II

- It's a form of **content analysis** in which we **transform raw texts** that consists of characters **into numeric vectors** to identify relations and regularities within and between different textual inputs. (Benoit 2009)
- While it is often used in exploratory analyses, it should **not** be thought and used as an **atheoretical** tool (Bonikowski and Nelson 2022)

What is computational text analysis? III



Time	Activity
1934	Laswell Produces first Key-Word Count
1934	Vygotsky Produces first Quantitative Narrative Analaysis
1950	Gottschalk Uses Content Analysis [illegible]ack Freudian Themes
1950	Turing Applies AI to text
1952	Bereleson Publishes First Textbook on Content Analysis
1954	First Automatic Translation of Text (Georgetown Experiment)
1963	Msteller and Wallace analyze Federalist Papers
1965	Tomashevsky Further Formalizes Quantitative Narrative Analysis

More on the development of text analysis: SICCS Introduction to Text Analysis

What we can do with text analysis I

In the 1990s, the Government's task will be to provide an economic environment which encourages enterprise - the mainspring of prosperity. Our aims must be:

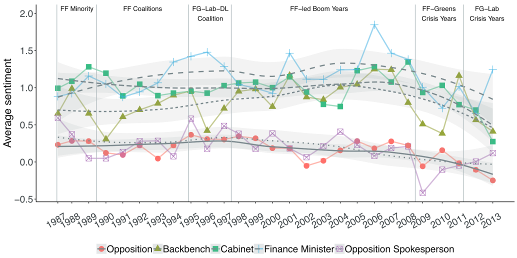
- To achieve price stability.
- To keep firm control over public spending.
- To continue to reduce taxes as fast as we prudently can.
- To make sure that market mechanisms and incentives are allowed to do their job.

Party programme of the British Conservative Party 1992

...extract political claims

What we can do with text analysis II

FIGURE 5
Sentiment Estimates in Irish Budget Debates, 1987–2013 [Colour figure can be viewed at wileyonlinelibrary.com]



...extract sentiment (Proksch et al. 2019)

What we can do with text analysis III

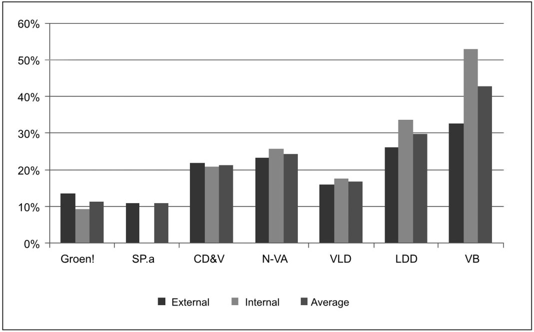
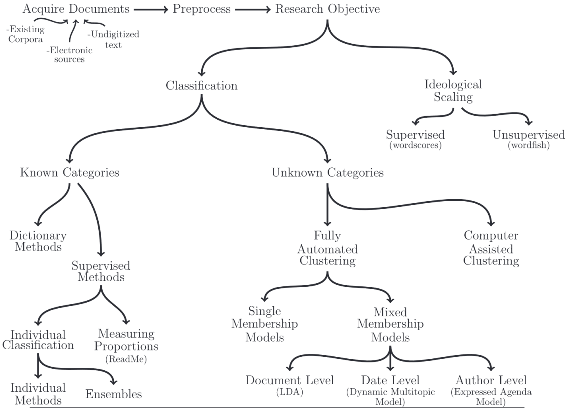


Figure 2. The degree of populist radical right values among Flemish parties.

...extract latent concepts (Pauwels 2011)

Different forms of text analysis I



A standard pipeline for bags-of-words approaches (Grimmer and Stewart 2013)

Different forms of text analysis II

Baden et al. (2022) distinguish between three different approaches in CTA

1. **rule-based text analysis**: classifying text by pre-defined rules (e.g., dictionaries, dependency-parsers etc.)
2. **supervised text analysis**: classifying text by observables; there is no clearly defined set of rules but pre-classified data from which we want to predict other instances (e.g., *Naive Bayes*, *GloVe*, *BERT*)
3. **unsupervised text analysis**: we do not know anything a priori but try to reduce complexity of a text inductively (e.g., *Latent Dirichlet Allocation*, *Wordfish*)

Caveats

> *"All Quantitative Models of Language Are Wrong—But Some Are Useful" (Grimmer and Stewart 2013)*

Models we use to make sense of text are always **incomplete** and **imperfect**. Like any other statistical models, they have measurement error (even Claude, Llama, GPT, DeepSeek or Mistral!)

Bags-of-words I

Imagine, we have several documents D which all consist of tokens (such as words) t. **Bags-of-words** (BoW) creates for each of D a vector of features t.

The EU condemns Russia for its war on Ukraine.

Bags-of-words II

The intuition behind bags-of-words approaches is that we can understand the **meaning** of a text from the **vocabulary** it uses. Comparisons between texts are based on the **frequency of terms** in a text.

Bags-of-words III

Document D1	*The child makes the dog happy* the: 2, dog: 1, makes: 1, child: 1, happy: 1
Document D2	*The dog makes the child happy* the: 2, child: 1, makes: 1, dog: 1, happy: 1

	child	dog	happy	makes	the	BoW Vector representations
D1	1	1	1	1	2	[1,1,1,1,2]
D2	1	1	1	1	2	[1,1,1,1,2]

Two documents with different meanings, yet same BoW representation (Source: AIML.com Research)

Bags-of-words IV

What's the problem with this structure?

Bags-of-words V

What's the problem with this structure?

- order of words is discarded
- disregarding grammatical structure
- context-blind (word with different meanings is treated the same)

Pipeline of bags-of-words

1. Data gathering
2. Data preparation
3. Data analysis
4. Validation

Data preparation

Except for descriptive purposes, we do not use our raw text but need to transform it.
Three transformations are common:

1. from data.frame to corpus object
2. from corpus to tokens
3. from tokens to document-frequency-matrix (also called document-term-matrix)

A corpus

A **corpus** object recognizes each row of an input vector as a document.

```
> corp <- corpus(df$sentence)
> head(corp,2)
Corpus consisting of 2 documents.
text1 :
"Peter Costello, Chris McDiven, my parliamentary
    colleagues a..."

text2 :
"This campaign, more than any other that I have been
    involved..."
```

A corpus II

- Usually, just the first transformation step.
- first glimpse into data with `summary(corpus_object)`
- also used for some statistics like the readability score `textstat_readability(corpus_object)`
- ...more on corpus objects

A tokens object

Splits the text into tokens which can be sentences, words, characters, and returns a matrix in the following form

```
> toks <- tokens(corp, what="word")
>
> head(toks[[1]], 20)
[1] "Peter"          "Costello"      ","
    "Chris"          "McDiven"       ","                 "my"
                "parliamentary"
[9] "colleagues"    "and"            "my"
    "fellow"         "Australians"   "."
```

A tokens object I

- tokens objects allow many pre-processing steps such as `remove_punct=T`, `remove_numbers=T`, `remove_symbols=T`, `remove_url=T`
- with `tokens_remove(stopwords())`, we can further remove stopwords of an object (like "and", "the", "a", "in")
- `kwic(token_object, "economy", window=n)` shows which terms surround a word
- there are different tokenizers, a more powerful alternative to quanteda is `library(spacyR)` which lemmatizes words and recognizes type of word (noun, verb, name, etc.)
- ...more on token objects

A document-frequency-matrix

A matrix with each row representing a document, and each column a text. Each cell shows how often a term has been used in a text

```
> m_dfm <- dfm(toks, tolower = T)
>
> m_dfm
Document-feature matrix of: 187,689 documents, 44,409 features (99.96% sparse) and 0 docvars.
features
docs    peter costello  ,    chris    mcdiven
text1     1        1    2        1          1
text2     0        0    2        0          0
[ reached max_ndoc ... 187,683 more documents, reached max_nfeat ... 44,399 more features ]
```

A document-frequency-matrix I

DFM are the object type which we will use for most of our analyses today

- with `docvars(dfm, "name") <- df$name`, we can add meta information (e.g. about communicator, time, topic)
- several functions to do text pre-processing operations
  - `dfm_keep`, `dfm_remove`, `dfm_subset`, `dfm_trim` and `dfm_replace` allows to drop or change features (terms) of the matrix
  - `dfm_lookup` allows to apply a dictionary to dfm
  - `dfm_group` groups dfm by a group defined in docvars()
  - `dfm_tfidf` weights each token by term-frequency inverse-document-frequency matrix
  - `dfm_sample` takes a random sample of documents (often used for training/test splits in classification tasks)

A document-frequency-matrix II

- `dfm_match` compares features of one dfm with another dfm and creates the same structure

- ...**more on dfm** objects

Let's do it in R

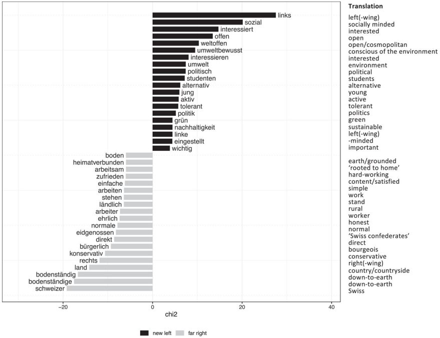
Descriptive statistics and visualization I

After we did all these pre-processing steps, we can visualize our data like that:

Bauer et al. (2017); in `quanteda`: `textplot_wordcloud(dfm)`

Descriptive statistics and visualization II

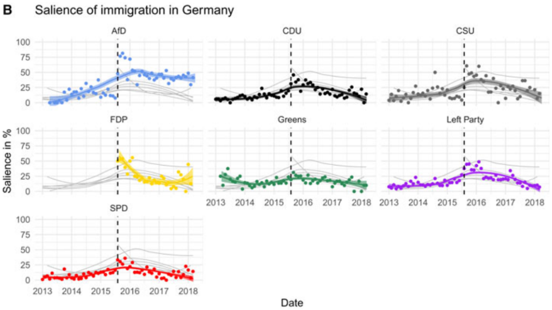
We can also calculate some statistics like keyness (based on χ^2), which terms are more often used by specific communicators?



Zollinger (2024); in `quanteda`: `textstat_keyness()` and `textplot_keyness()`

Descriptive statistics and visualization III

A very common and despite its simplicity valuable task is a dictionary analysis



Gessler and Hunger (2022); in `quanteda`: `dfm_lookup(dfm, dict)`

Let's do it in R

One more note on pre-processing

There are different views on how much you should pre-process (for an overview, please refer to Chai 2023).

Pros	**Cons**
Reduces complexity	Oversimplifies language
Improves interpretability	Leads to arbitrary results
Speed boost in model estimation	Time consuming in model preparation

Gold standard?

There is no pre-defined pipeline in pre-processing. If we account for the major pre-processing techniques (stopword removal, punctuation, ngrams, etc.), there are 128 potential models. Denny and Spirling (2017) warn researchers of selecting arbitrary steps.

1. Pre-processing requires a theory of your text [why are certain steps warranted whereas others may not?]
2. If you do pre-process, you should check the sensitivity of your results

An empirical test

Denny and Spirling (2017) provide a `R` package called `preText` that compares the effect of different pre-processing steps on your model. [you'll find it at the end of the script]

Unsupervised methods

- sometimes, we want to categorize text but do not know the categories a priori
- in this circumstances, we use unsupervised methods which are "a class of methods that learn underlying features of text without explicitly imposing categories of interest" (Grimmer and Stewart 2013, p. 281)
- we'll use them mainly for clustering

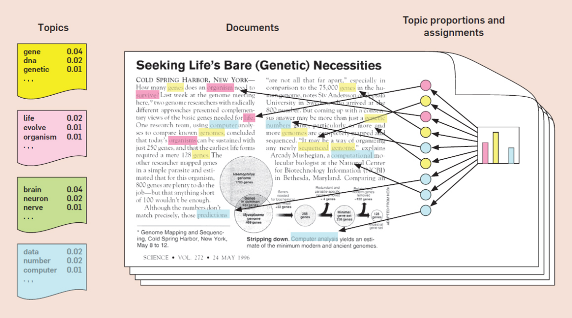
Topic models

- topic models are one form of **clustering**
- compared to single-membership models (clustered into distinct categories), topic models are **mixed-membership models** based on the assumption that each text uses vocabulary of different topics
- data requirement: large corpora of texts, pre-processed as described

General idea of topic models I

- documents consists of words whose combination represents a **hidden (latent) semantic meaning**
- idea: author of a text writes a text on topic *k* using those terms that are associated with it
- depending on the composition of words,each document has a **probability p of belonging to topic k**
- core techniques are **L**atent **S**emantic **A**nalysis (**LSA**) and **L**atent **D**irichlet **A**llocation (**LDA**)

General idea of topic models II



Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77–84.
https://doi.org/10.1145/2133806.2133826

General idea of topic models III

- **LDA** is a probabilistic model in which each term t has a certain probability of belonging to topic k, and each document d is composed of multiple k
  - mainly used for topic modelling
  - generative model in which topics are first randomly assigned to documents; each word has a probability of belonging to topic k
  - iterative process in which log likelihood is reduced (the smaller the better)
  - number of k needs to be defined a priori (no strict guidelines, generally: the larger the corpus, the more topics it includes)

...more on LDA (Blei, Ng, and Jordan 2003)

Topic models in R I

There are different packages available to estimate a topic model, such as `lda` and `stm`.

Structured topic model

- in its basic application, **structured topic models** (stm) resemble a Latent Dirichlet Allocation
- we can however add meta information to the topic model to improve its predictive power
  1. topic **prevalence**: covariates that influence the frequency of a topic (like seasonal effects)
  2. topic **content**: covariates that influence how a topic is discussed (e.g., by different parties)
- stm also allows to specify $k = 0$ whereby k is determined automatically (using a different distribution, `init.type="Spectral"` instead of LDA); *be aware*: that does not mean that $\hat{k}$ is the true k

Topic models in R II

Steps in R

1. **Preparation** (as done before: from data frame to corpus, to tokens, to data-frequency-matrix)
2. **Estimation** of topic model
3. **Interpretation**
4. **Validation**

Estimation and Interpretation

Main choice: K $\rightarrow$ number of topics

```
stm1 <- stm(m_dfm, K = 20, seed=421)
labelTopics(stm1)
Topic 1 Top Words:
Highest Prob: national, policy, change, climate,
    policies, government, strategy
FREX: positive, policy, equality, immigration,
    priority, common, population
Lift: highlights, positive, anti-nuclear, nps,
    2010-2020, formulation, internationalism
Score: policy, positive, climate, change,
    environmental, policies, national
Topic 2 Top Words:
Highest Prob: new, zealand, industry, water,
    environment, food, sustainable
```

Let's do it in R

Validation

Validation can be based on

1. **technical** parameters
2. **manual** coding

Technical validation

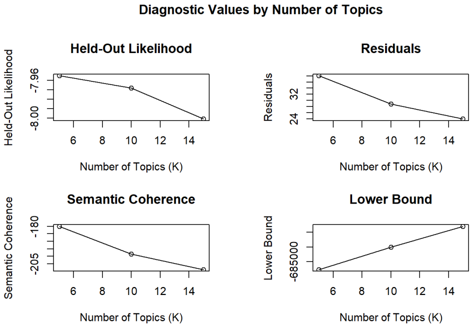
We can use stm's `searchK()` function to determine the best number of topics k based on likelihood, lower bound, residuals and semantic coherence.

```
# convert dfm to stm
m_stm_sub <- convert(m_dfm_sub,"stm")

# create vector with different numbers for k (here, 5,
    10 and 15)
K<- c(5,10,15)

# run validation
best_k <- searchK(m_stm_sub$documents, m_stm_sub$vocab,
    K)
```

Technical validation



Advancing topic models

- available using R package `topicmodels`
- correlated topic models: allow for correlation between latent topics; in R, run `CTM` (see for application and Blei and Lafferty (2005) for methodological background)
- seeded topic models: semi-supervised approach, topic model is fed with pre-specified keywords; in R, run `textmodel_seededlda` (see for application and Watanabe (2021) and Watanabe and Baturo (2024) for methodological background)

Scaling I

Sometimes, we have a clearly defined topic (e.g., the economy) and want to assess where actors stand on this topic

- early approach: **Wordfish** (Slapin and Proksch 2008)
  - calculates a weight for each word (in its initial application: how much does it distinguish parties)
  - provides a **position** of a party on a unidimensional scale
  - usually applied on a more aggregated level (like a document level)

Scaling II

- more recent approach: **Latent Semantic Scaling** (Watanabe 2021)
  - LSS “creates a polarity score of words depending on a certain number of seed words” (Watanabe 2021)
  - it’s semi-supervised, we use our domain-specific knowledge to actually determine the poles of our unidimensional space
  - applied on the sentence level

Wordfish in R

```
> m_wordfish <- textmodel_wordfish(m_dfm2)
> summary(m_wordfish)

Call:
textmodel_wordfish.dfm(x = m_dfm2)

Estimated Document Positions:
    theta       se
2001.1 -0.9434 0.02570
[...]
```

Latent Semantic Scaling in R

1. define polarisation terms
2. estimate LSS model

```
lr_dict <- dictionary(list(left = c("unemployment",
    "justice", "wage", "employee", "bargaining"),
right = c("budget", "merit", "deficit", "business",
    "growth")))
seed <- as.seedwords(lr_dict)
```

Latent Semantic Scaling in R

```r
lss_model <- textmodel_lss(m_dfm2, seeds = seed,
k = 300, auto_weight = T)
```

→ *k* is the number of singular values (dimensionality reduction); auto-weight adjusts the weights to tokens' similarity to seed words

Let's do it in R

Time for a break?

Supervised methods

For supervised tasks, we have information at hand to make inference about new cases.

Two types of supervised tasks

1. **Regression** is used for making predictions of continuous outcome variables
2. **Classification** is used for classifying data into n categories (categorical outcome variable)

More on supervised methods

Predicting text

Sometimes, we know the categories of interest a priori and want to predict whether a certain text is part of a category.
Solution: **supervised models**!

- in our case, we use MARPOR data that comes with topic classifications
- frequently used for sentiment analysis as well

Goal: Inferring relations between texts of a large corpus of text based on a small subset of annotated data

Basics concepts I

- **gold standard**: to train our models, we need annotated data, the gold standard of it being hand-coded data; the more data, the better the prediction (especially for complex latent concepts)
- **machine learning**: prediction tasks involve machine learning; it's an iterative process which assigns a weight to terms to reduce the error (for an introduction to machine learning)
- **training/test data**: to validate whether we are right or wrong, we need to leave out some data for testing purposes
- **inference**: the whole purpose of these tasks is to predict unlabelled examples

Basics concepts II

- **classification algorithm**: model used to classify our data (binary, multiclass, multilabel)
- **performance metrics**: accuracy, F1-score, true/false positive rates used to evaluate how good a model performs

Classification algorithms I

Naive Bayes

- family of classifiers that are based on Bayes' Theorem
- probability is assessed by the occurrence of an event
- e.g., how likely is it that someone speaks about migration if they use the features 'Refugees', 'are', 'welcome', 'here', '.'
- assumes feature independence (treats every feature independently) → in the example above, the term 'refugees' has nothing to do with 'welcome'
- each feature is assigned the same weight → 'here' is equally important as 'Refugees'

Classification algorithms II

	Outlook	Temperature	Humidity	Windy	Play Golf
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes

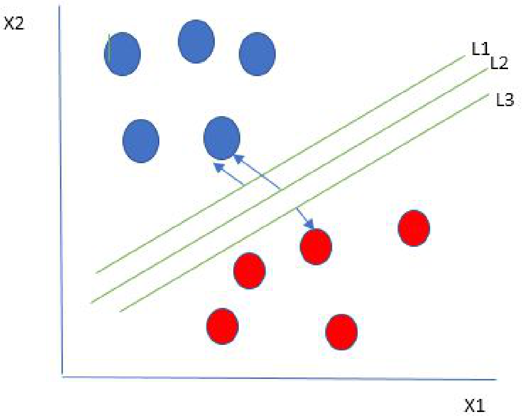
Graph and intuition

Classification algorithms III

Support vector machines

- general idea: draw a hyperplane (decision boundary) that separates our data best
- support vectors are those cases which are closest to the hyperplane
- we want to maximize the distance between empirical cases of our classes

Classification algorithms IV



Graph and intuition

Data

Our annotated data should consist of at least two columns

1. **feature**: one or more input vector(s) (like words in a text)
2. **label**: a variable indicating the category of the feature

Think of it in regression terms, the label is y and the feature x; we are trying to explain y by x_N

Classification in R

- today: `quanteda.textmodels`
- tomorrow: `keras3` for deep learning

Preparation

Assuming that we have a dfm with labels as docvars, we have to split our dfm into training and test data (ratio depends on you, but often 80/20 is chosen).
Since we sample documents, we need to remove those features (words) that are not present anymore from our dfm.

```
> train <- dfm_sample(m_dfm_sub,0.8*nrow(df_sub))
> test <- dfm_subset(m_dfm_sub,
+   !(docnames(m_dfm_sub) %in% docnames(train)))
> train <- train %>% dfm_trim(1)
> test <- dfm_match(test, featnames(train))
```

Model estimation

After this, we can already use an algorithm to train with our training data.

```
> nb_model<-textmodel_nb(train,docvars(train, "label"))
> nb_model

Call:
textmodel_nb.dfm(x = train, y = docvars(train, "label"))

Distribution: multinomial ; priors: 0.5 0.5 ; smoothing
    value: 1 ; 18796 training documents;  fitted
    features.
```

Model inference

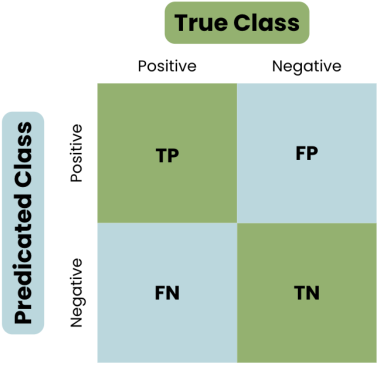
Given that we have set the weights of our model, we can use it for inference (here on our test data to evaluate its performance but potentially also on out-of-sample data)

```
> test_predictions<-predict(nb_model, newdata=test)
```

Let's do it in R

Evaluation I

As we have annotated data already, evaluation is easier than with unsupervised techniques. There are different metrics we can consider:



Evaluation II

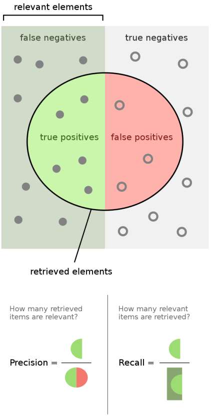
Accuracy

$$\frac{TP + TN}{N} \tag{1}$$

How many cases did we predict correctly?

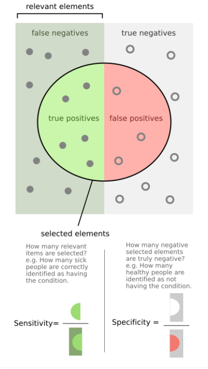
Evaluation III

Precision and Recall



Evaluation IV

Sensitivity and Specificity



Evaluation V

F1-Score:
Metrics may be less informative if we deal with an imbalanced dataset. For this purpose, we often use the F1-score

$$2 \times \frac{Precision * Recall}{Precision + Recall} \tag{2}$$

Evaluation V

Afterwards, we can create a matrix that consist of empirical and predicted value to check our performance (confusion matrix)

```
> (eval_mat <-
    table(docvars(test,"label"),test_predictions))
test_predictions
          Reference
Prediction    0    1
0  396  239
1   50 4014
```

Evaluation VI

`caret` also provides more sophisticated evaluation metrics

```
> confusionMatrix(test_predictions,
    as.factor(docvars(test,"label")))
[...]
Accuracy : 0.9385
[...]
Sensitivity : 0.88789
Specificity : 0.94380
Balanced Accuracy : 0.91585
```

Thank you for your attention!

References I

Baden, C., Pipal, C., Schoonvelde, M., & van der Velden, M. A. C. G. (2022). Three Gaps in Computational Text Analysis Methods for Social Sciences: A Research Agenda. *Communication Methods and Measures*, *16*(1), 1–18. https://doi.org/10.1080/19312458.2021.2015574

Bauer, P. C., Barberá, P., Ackermann, K., & Venetz, A. (2017). Is the Left-Right Scale a Valid Measure of Ideology? *Political Behavior*, *39*(3), 553–583. https://doi.org/10.1007/s11109-016-9368-2

Benoit, K. (2009). Introduction to quantitative text analysis.

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77–84. https://doi.org/10.1145/2133806.2133826

Blei, D. M., & Lafferty, J. D. (2005). Correlated topic models. *Proceedings of the 18th International Conference on Neural Information Processing Systems*, 147–154.

References II

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *The Journal of Machine Learning Research*, *3*(null), 993–1022.

Bonikowski, B., & Nelson, L. K. (2022). From Ends to Means: The Promise of Computational Text Analysis for Theoretically Driven Sociological Research. *Sociological Methods & Research*, *51*(4), 1469–1483. https://doi.org/10.1177/00491241221123088

Chai, C. P. (2023). Comparison of text preprocessing methods. *Natural Language Engineering*, *29*(3), 509–553. https://doi.org/10.1017/S1351324922000213

Denny, M., & Spirling, A. (2017). Text Preprocessing for Unsupervised Learning: Why It Matters, When It Misleads, and What to Do about It. https://doi.org/10.2139/ssrn.2849145

References III

Gessler, T., & Hunger, S. (2022). How the Refugee Crisis and Radical Right Parties Shape Party Competition on Immigration. *Political Science Research and Methods*, *10*, 524–544. https://doi.org/10.1017/psrm.2021.64

Grimmer, J., & Stewart, B. M. (2013). Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts. *Political Analysis*, *21*(3), 267–297. https://doi.org/10.1093/pan/mps028

Pauwels, T. (2011). Measuring Populism: A Quantitative Text Analysis of Party Literature in Belgium. *Journal of Elections, Public Opinion and Parties*, *21*(1), 97–119. https://doi.org/10.1080/17457289.2011.539483

References IV

Proksch, S.-O., Lowe, W., Wäckerle, J., & Soroka, S. (2019). Multilingual Sentiment Analysis: A New Approach to Measuring Conflict in Legislative Speeches. *Legislative Studies Quarterly*, *44*(1), 97–131. https://doi.org/10.1111/lsq.12218

Slapin, J. B., & Proksch, S.-O. (2008). A Scaling Model for Estimating Time-Series Party Positions from Texts. *American Journal of Political Science*, *52*(3), 705–722. https://doi.org/10.1111/j.1540-5907.2008.00338.x

Watanabe, K. (2021). Latent Semantic Scaling: A Semisupervised Text Analysis Technique for New Domains and Languages. *Communication Methods and Measures*, *15*(2), 81–102. https://doi.org/10.1080/19312458.2020.1832976

References V

Watanabe, K., & Baturo, A. (2024). Seeded Sequential LDA: A Semi-Supervised Algorithm for Topic-Specific Analysis of Sentences. *Social Science Computer Review*, *42*(1), 224–248. https://doi.org/10.1177/08944393231178605

Zollinger, D. (2024). Cleavage Identities in Voters' Own Words: Harnessing Open-Ended Survey Responses. *American Journal of Political Science*, *68*(1), 139–159. https://doi.org/10.1111/ajps.12743