Center for
Big Data Statistics

Natural Language Processing

Piet Daas, Statistics Netherlands &
Eindhoven University of Technology

Towards **smart statisti**

Natural Language Processing

- Natural language processing (NLP)

  a subfield of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data

  - Use computers to derive meaning from text!

Natural Language Processing (2)

- There are many NLP tasks:
  - Topic detection
  - *Classification of text*
  - Semantic analysis
  - Sentiment analysis/opinion mining
  - Name entity recognition
  - Automatic summarization
  - Machine translation
  - Speech recognition
  - Answering questions
  - Fake news detection
  - …
  - These all require the interpretation of language by computers

Natural Language Processing (3)

- Usually in NLP algorithms are applied to identify and extract information from text
  - Machine learning is heavily used nowadays
  - In the 'early days' more rule-based approaches were used

- Text is converted in a form computers can 'understand'
  - It is converted into numbers!

- But language is not always easy for computers to understand
  - "*John hit the dog with a stick*"

- Because its a broad area
  - *Let's focus on steps needed to Classify texts*

0. Obtain documents

- Document containing texts
  - Web pages
  - PDF files
  - Word documents
  - Social media messages
  - Emails
  - Product descriptions
  - …
- Size matters, different approach for:
  - Short texts, such as social media messages
    v.s.
  - Other (larger) documents
- For classification: essential to have annotated examples

Creating a text classifier

- Process overview

Webpages/
Documents

Extract
text

1. 'Text indexing'

Preprocessing (tokenization, lowercase, only characters, stop word removal, character length, stemming, ….)

More in: Taeho, J. (2019). Text mining: concepts, implementation, and big data challenge. Springer.

1. ‘Text indexing’: preprocessing

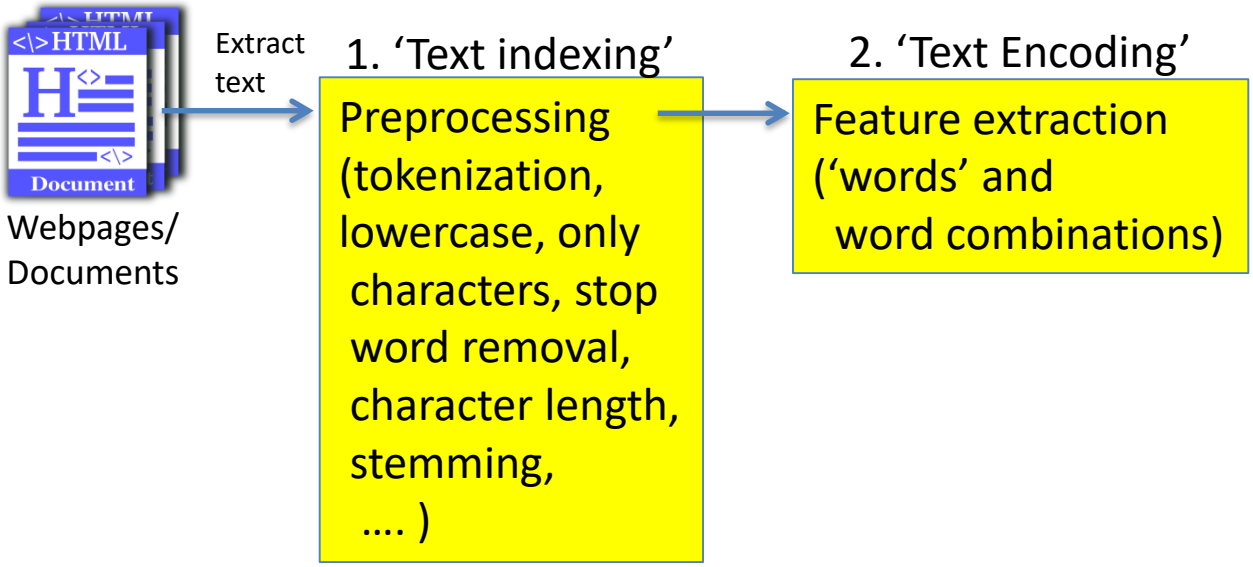
- After extraction, the text is usually to preprocessed

- Tokenization
- Convert to lowercase,
- Remove punctuation marks and numbers,
- Remove stop words (language dependent),
- Remove ‘small’ sized words (char <= 2)
- Do stemming / lemmatization (language dependent).

- This creates a more homogeneous text representation

Creating a text classifier

– Process overview



- Multistage model development
- Most common approach applied

More in: Taeho, J. (2019). Text mining: concepts, implementation, and big data challenge. Springer.

2. ‘Text encoding’

- Preprocessed text is subsequently encoded
  - This is the step were the ‘features’ are converted to a numeric representation

- *Character sequences*
  - Various lengths (often 2-5)
  - Count occurrences, Term frequency, TF-IDF
- *Single Words*
  - ‘Bag of words’ approach
  - Count occurrences, Term frequency, TF-IDF
- *Word combinations*
  - Usually bigrams, sometimes also trigrams
  - Count occurrences, Term frequency, TF-IDF

Term Frequency, TF-IDF explained

Doc1: ‘A: this is a sample’ Doc2: ‘Another example: this is another example, example’

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

Term Frequency:

tf(‘this’, doc1) = 1/5 = 0.2

tf(‘this’, doc2) = 1/7 ≈ 0.14

Term Frequency Inverse Document Frequency (TF-IDF):

tfidf(‘this’, d1) = tf(‘this’, d1)*idf(‘this’, D) = 0.2 * log(2/2) = 0

tfidf(‘this’, d2) = tf(‘this’, d2)*idf(‘this’, D) = 0.14 * log(2/2) = 0

Term Frequency, TF-IDF explained

Doc1: ‘A: this is a sample’ Doc2: ‘Another example: this is another example, example’

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

Term Frequency:

tf(‘example’, doc1) = 0/5 = 0

tf(‘example’, doc2) = 3/7 ≈ 0.429

Term Frequency Inverse Document Frequency:

tfidf(‘example’, d1) = tf(‘example’, d1)*idf(‘example’, D) = 0 * log(2/1) = 0

tfidf(‘example’, d2) = tf(‘example’, d2)*idf(‘example’, D) = 0.28 * log(2/1) ≈ 0.129

2. ‘Text encoding’

- Preprocessed text is subsequently encoded
  - This is the step were the ‘features’ are converted to a numeric representation

- *Character sequences*
  - Various lengths (often 2-5)
  - Count occurrences, Term frequency, TF-IDF
- *Single Words*
  - ‘Bag of words’ approach
  - Count occurrences, Term frequency, TF-IDF
- *Word combinations*
  - Usually bigrams, sometimes also trigrams
  - Count occurrences, Term frequency, TF-IDF
- *Word embeddings*
  - Words in their context, skip n-grams
  - Vector representation (usually a Neural network)

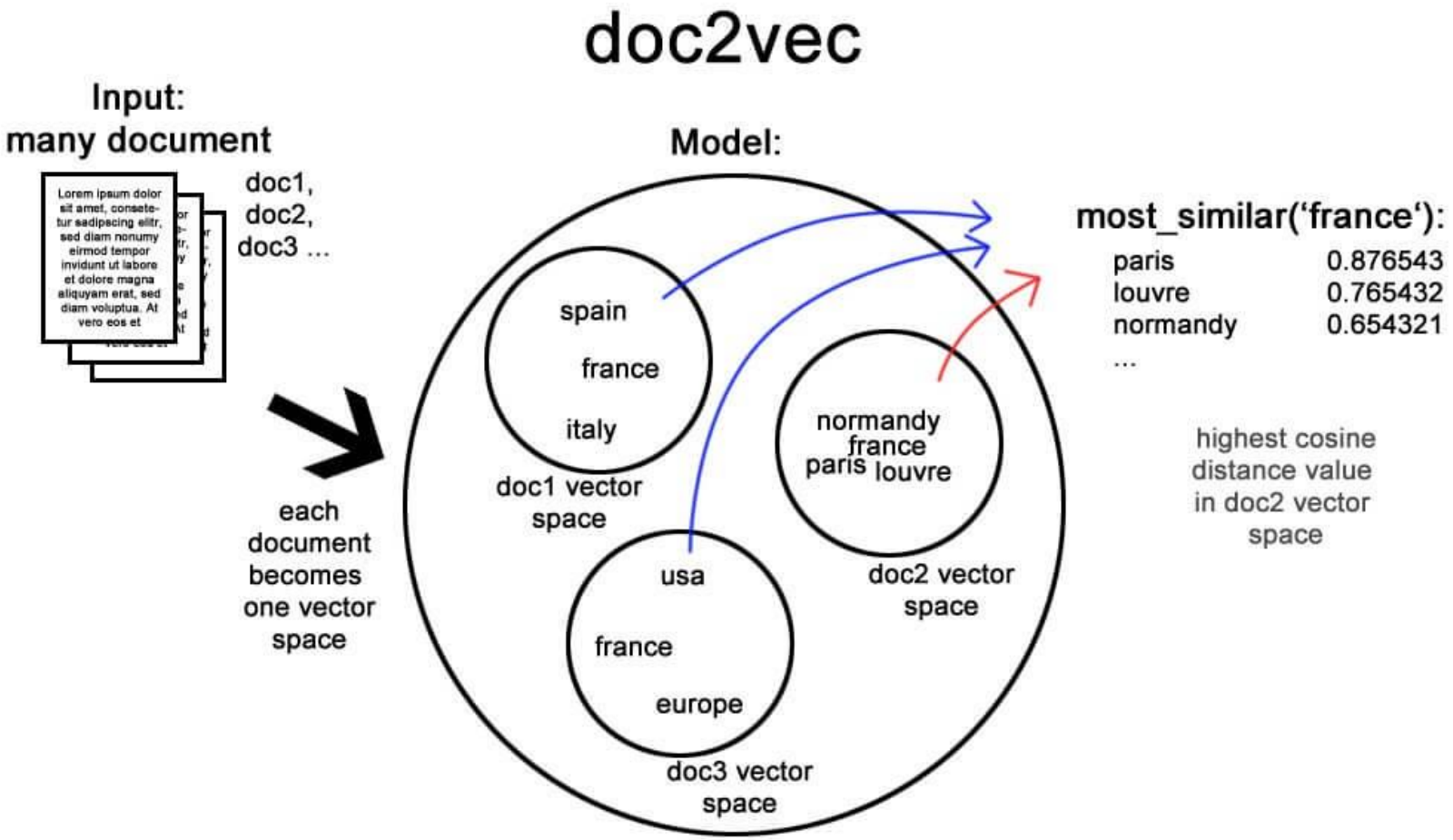
Word embeddings

- Word embeddings is currently one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc.

- Popular implementations:
  - Word2Vec (Google)
  - GloVe (Stanford)
  - fastText (Facebook)

  - Make use of word co-occurences. The (cosine) distance between words is an import property.

Word embeddings (2)



Creating a text classifier

– Process overview

Webpages/
Documents

Extract
text

1. 'Text indexing'

Preprocessing (tokenization, lowercase, only characters, stop word removal, character length, stemming, ….)

2. 'Text Encoding'

Feature extraction ('words' and word combinations)

Document Term Matrix with values

Class (0/1) training/test set

3. 'Text Categorisation'

'Machine Learning' algorithm

Training

Testing

Text based classification model

- Multistage model development
- Most common approach applied

Center for Big Data Statistics

More in: Taeho, J. (2019). Text mining: concepts, implementation, and big data challenge. Springer.

3. ‘Text categorization’

- Often Machine learning is used (including NN/DL)
  - Automatic optimization of classification
  - Many algorithms can be used
  - Test to see what algorithm (and indexing and preprocessings steps) works best
  - Critically check the findings

- Use an annotated Training and Test set (80%/20%)
  - Preferably also with an external validation of the final model!

An example: Innovative company detection

- Goal: Detecting innovative companies via the text on their web site
  - Can the text on a web page be used to detect if a company is innovative?
  - Web pages can be 'scraped' fairly easy

- In this study we looked at:
  - The potential of *web pages* to provide information on the *innovative* character of a company
  - Data from the Community Innovation Survey was used to create a training and test set

The Community Innovation Survey

- The Community Innovation Survey (CIS)
  - Focusses on the innovativeness of companies
  - Is a European standardized survey

- The questionnaire is send every other year to about 10,000 companies in the Netherlands
  - Stratified sample of companies
  - Only companies with 10 or more working persons (WP)
  - Company provides info on innovation character and type

  - We focussed on technological innovation
  - *(CIS lacks info on small companies, such as start-ups!)*

Model building: import considerations

- Web pages were processed (html-files) and words extracted
  - Effect of various pre-processing steps
  - Later on: additional removal of words

- A supervised classification task
  - Tested various algorithms (80/20 training/test set)
  - Compared various metrics (Accuracy is best)
  - Started with TF-IDF value in DTM (Log(TF-IDF+1) is better)
  - Effect of including words above a specific number of characters (2, 3)
  - Effect of including Word embeddings

***Table 1**. Results for the various classification algorithms tested. Default settings were used. The average and standard deviation of 1000 tries are shown as percentages.*

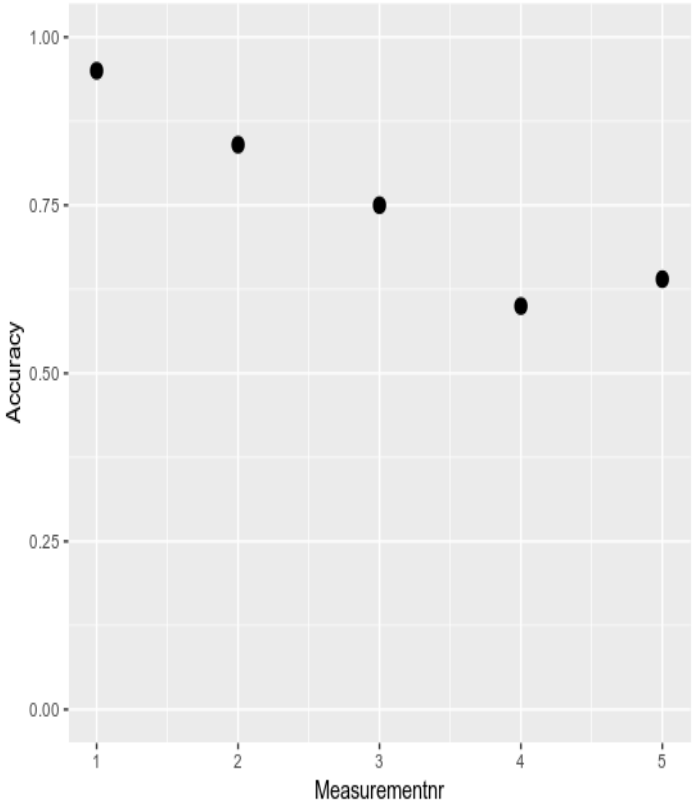
	Words of 2 and more characters Accuracy (%)
Bernoulli Naïve Bayes	87 ± 1
Logistic Regression (L1 regularization)	94 ± 1
Nearest Neighbors (k = 2)	61 ± 2
Support Vector Machine (linear)	93 ± 1
Support Vector Machine (radial basis)	53 ± 1
Stochastic Gradient Decent	93 ± 1
Quadratic Discriminant Analysis	77 ± 8
Neural Network (multi-layer perceptron)	92 ± 1
Decision Tree	94 ± 1
Random Forests	94 ± 1
Gradient Tree Boosting	94 ± 1

2 character words (such as ‘nl’, ‘be’, ‘en’) dominated the model and were therefore ignored

External validation & model stability

- Tested the model on:
  - Web sites of start-ups (92% innovative)
  - Web sites of small companies (WP < 10) (around 33%)

- However long-term stability was found to be an issue
  - Solved this by including additional classified data
  - A standard Machine Learning way to deal with this problem



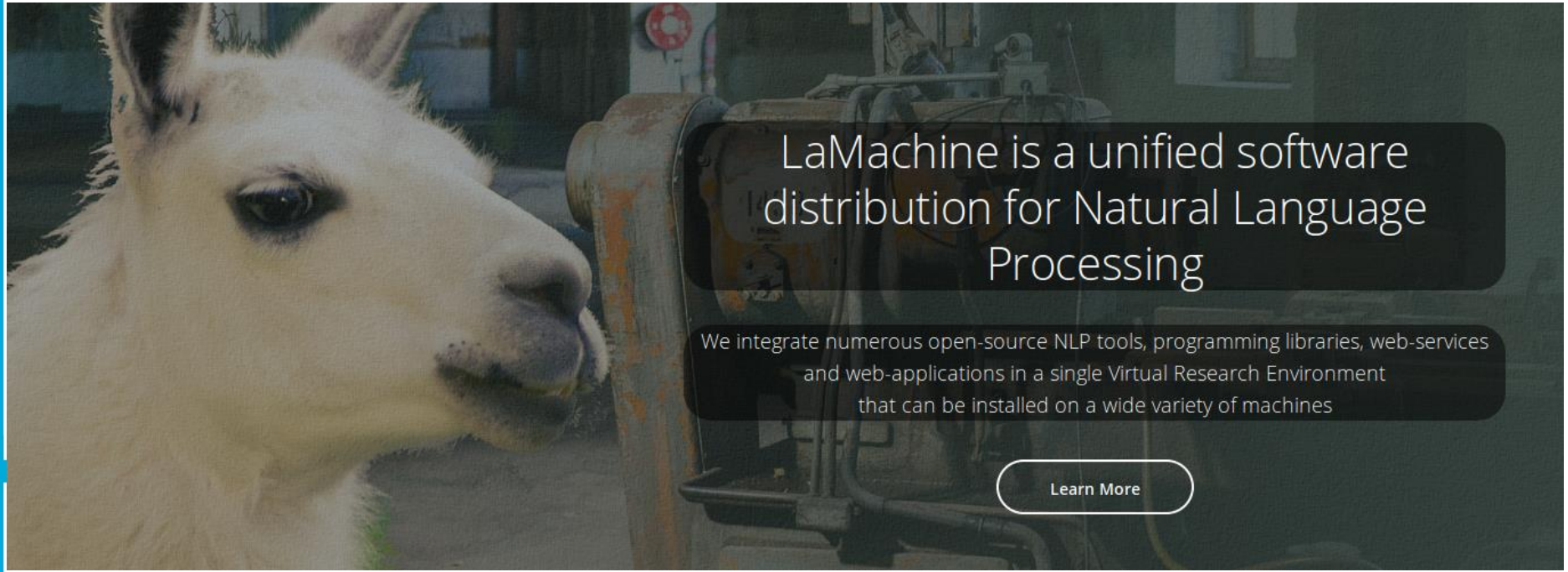
Final model details

- 88% Accuracy over various datasets
- A single model including words in both languages
- 580 stemmed words included in the model

- An English website is a positive indication for innovation (compared to Dutch)
- Depending on the language, there are words that are clearly positive associated with innovation.
  - ***Positive***: **Technology, innovation, software, data**
  - *Negative*: Sale, buy, shopping car, exclusive, service

NLP and the Dutch language

- More info on Dutch NLP tools can be found at:
  - LaMachine website: https://proycon.github.io/LaMachine/
  - Contains tools and libraries for analysing *Dutch* texts




Center for
Big Data Statistics
Towards Smart Statistics

Creating a text-based model

Parsing

Pre-processing text

- Remove scripts
- Language detection
- To lower case
- Number and punctuation mark removal
- Remove Stop words
- Remove words below a specific length
- Stemming

Feature engineering

- Bag of words
- Minimal word freq.
- Minimum number of words (>= 10)
- TF-IDF / Log(TF-IDF + 1)
- Effect of Word embeddings

Model building

- Dataset(s) used
- Algorithm selection
- Hyperparameter tuning

Model evaluation

- Training/test split.
- Evaluation metric
- External validation

Model stability

- Performance over time

Center for Big Data Statistics