Python Workshop:
Natural Language Processing

Ties de Kok

Tilburg University

Agenda

What are we going to discuss today?

1. Positioning session
2. Terminology
3. Python for NLP
4. NLP Python tools
5. Topics:
    - Process and Clean text
    - Direct feature extraction
    - Represent text numerically
    - Machine learning

Agenda

Positioning

Where does this session fit into the bigger scheme of NLP?

- Determining relevance textual data
- Finding sources textual data
- Gathering textual data
- **Processing textual data**
- **Analyzing textual data**

Agenda

Positioning

Terminology

Many inter-related names and terms:

- Computational Linguistics
- Textual Analysis
- **Text Mining**
- **Natural Language Processing**

Which programming language / software to use?

- **Python**
- R
- PERL

External NLP-relevant Python libraries

Standard NLP libraries:

1. `NLTK` and the higher-level wrapper `TextBlob`
2. `Spacy` and the higher-level wrapper `Textacy`

Standard machine learning library:

1. `scikit learn`

Topic modelling library *(not covered)*:

1. `Gensim`

Agenda

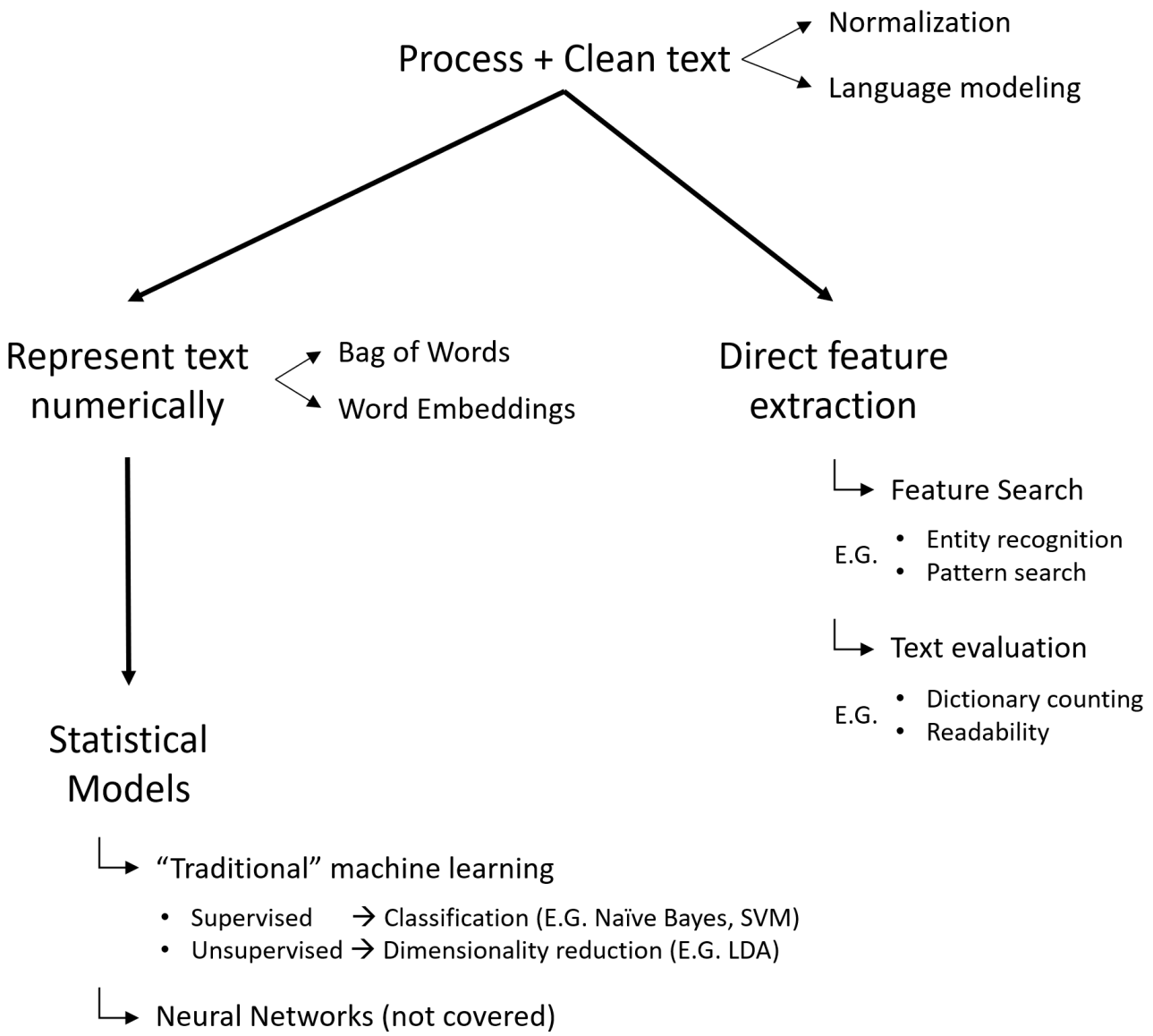
Positioning

Terminology

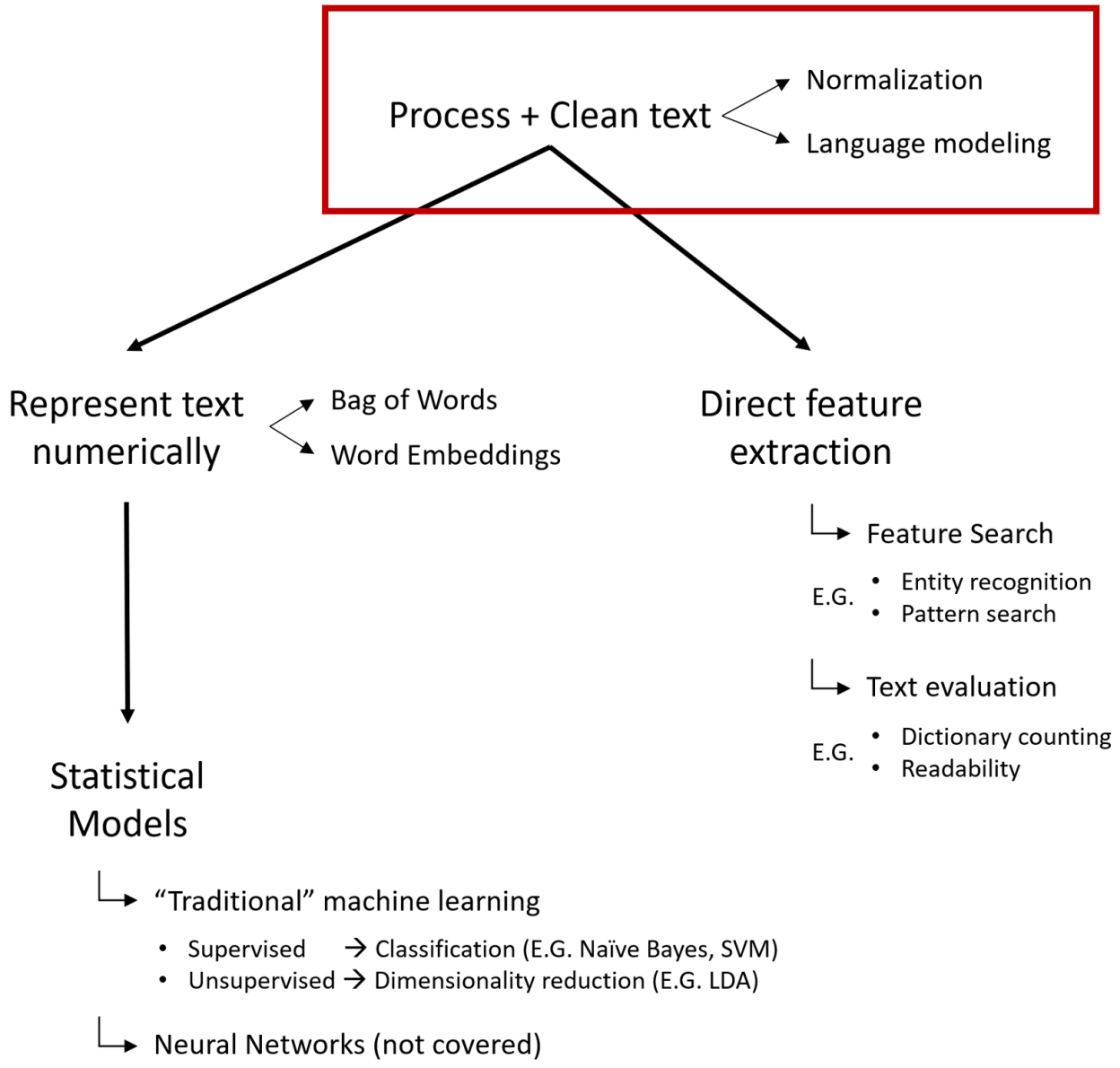
Language

NLP Python

NLP Space



Process & Clean



Process & Clean

Text normalization

- Sentence segmentation

  > i.e. split text up into sentences

- Word tokenization

  > i.e. split sentence up into tokens (i.e. words)

- Entity normalization

  > i.e. "http://www.google.com" → "URL"

- Lemmatization & Stemming

  > Convert tokens to a base representation

Process & Clean

Language modelling

Text has a complex underlying structure that you can tap into.

- Part-of-Speech tagging

  > Identify the "Word Class" of a token (e.g. noun, verb)

- Remove stop words

  > Remove words that don't carry any informational value

- Uni-Gram vs. N-Grams

  > Multi-word token: retain some of the sequential nature

Uni-Gram vs. N-Grams

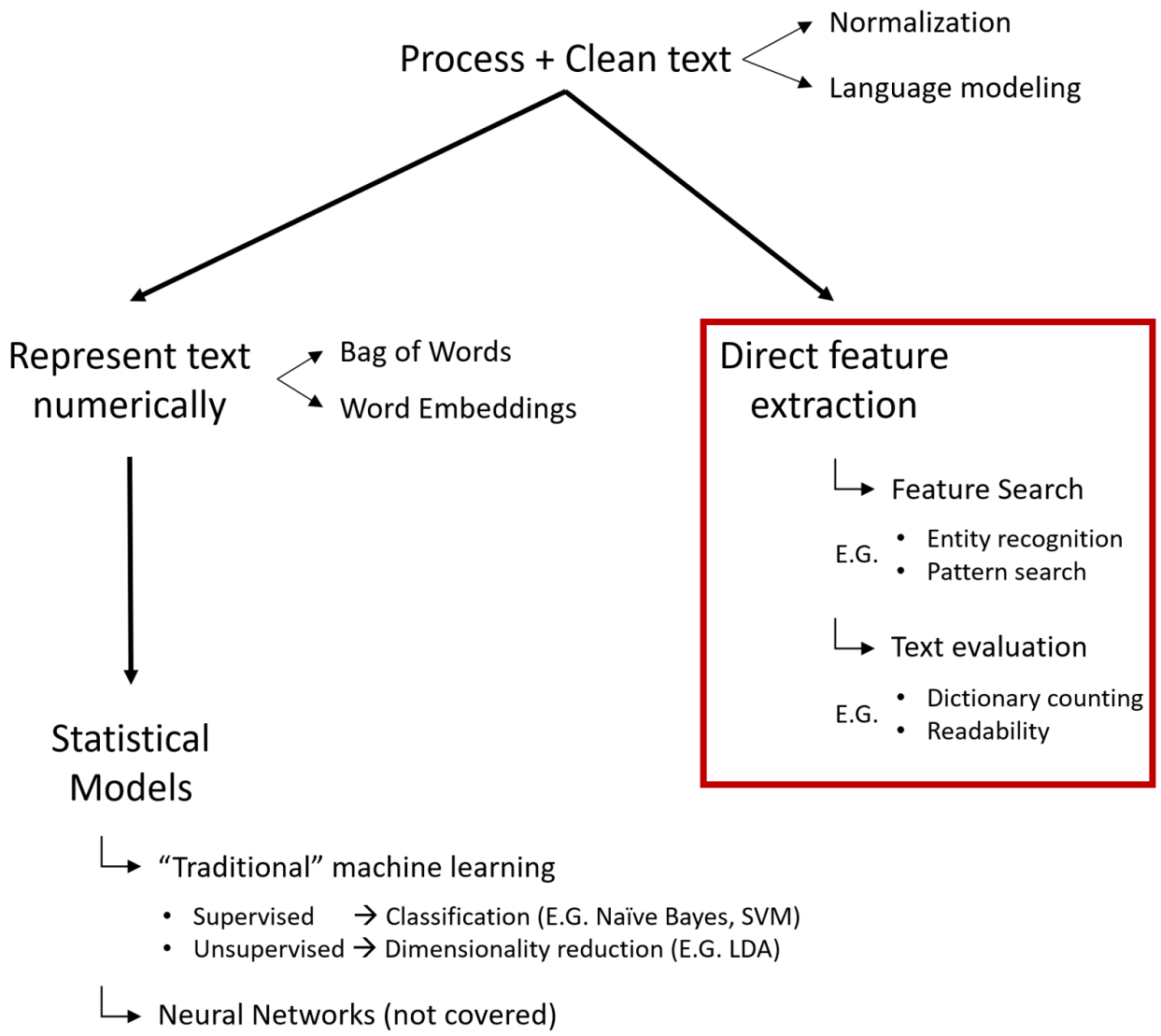
> Multi-word token: retain some of the sequential nature

"Tilburg University is located in Noord Brabant"

Unigram	Bigram	Trigram
Tilburg	Tilburg-University	Tilburg-University-is
University	University-is	University-is-located
is	is-located	is-located-in
located	located-in	located-in-Noord
in	in-Noord	in-Noord-Brabant
Noord	Noord-Brabant	
Brabant		

Process & Clean

Feature Extraction



Process & Clean

Feature Extraction

Feature search

- Entity extraction

  > e.g. extract PEOPLE / EVENTS / DATES / MONETARY VALUES

- Pattern search (`RE`)

  > i.e. use `Regular Expressions` to look for patterns

- Term (Dictionary) counting

  > i.e. count the number of times a term occurs

Process & Clean

Feature Extraction

Pattern search (RE)

```
import re

string_1 = 'Ties de Kok (#IDNUMBER: 123-AZ). Rest of text...'
string_2 = 'Philip Joos (#IDNUMBER: 663-BY). Rest of text...'

pattern = re.compile('#IDNUMBER: (\d\d\d-\w\w)')

print(pattern.findall(string_1)[0])
print(pattern.findall(string_2)[0])
```

Regular Expression

```
123-AZ
663-BY
```

TIP: Use Pythex.org or Regex101.com to try out your regular expression

- Example on Pythex: click here
- Example on Regex101: click here

Process & Clean

Feature Extraction

Term (Dictionary) counting

```
pos = ['great', 'increase']
neg = ['bad', 'decrease']

sentence = '''According to Trump everything is great, great,
and great even though his popularity is seeing a decrease.'''

pos_count = 0
for word in pos:
    pos_count += sentence.lower().count(word)
print(pos_count)

neg_count = 0
for word in neg:
    neg_count += sentence.lower().count(word)
print(neg_count)

pos_count / (neg_count + pos_count)
```

This is the hard part!

This is easy!

```
3
1

Out[12]: 0.75
```

Process & Clean

Feature Extraction

Text evaluation

- Language

> i.e. detect whether text is English

- Readability

> i.e. use the `TextStat` package to calculate text statistics

- Text similarity

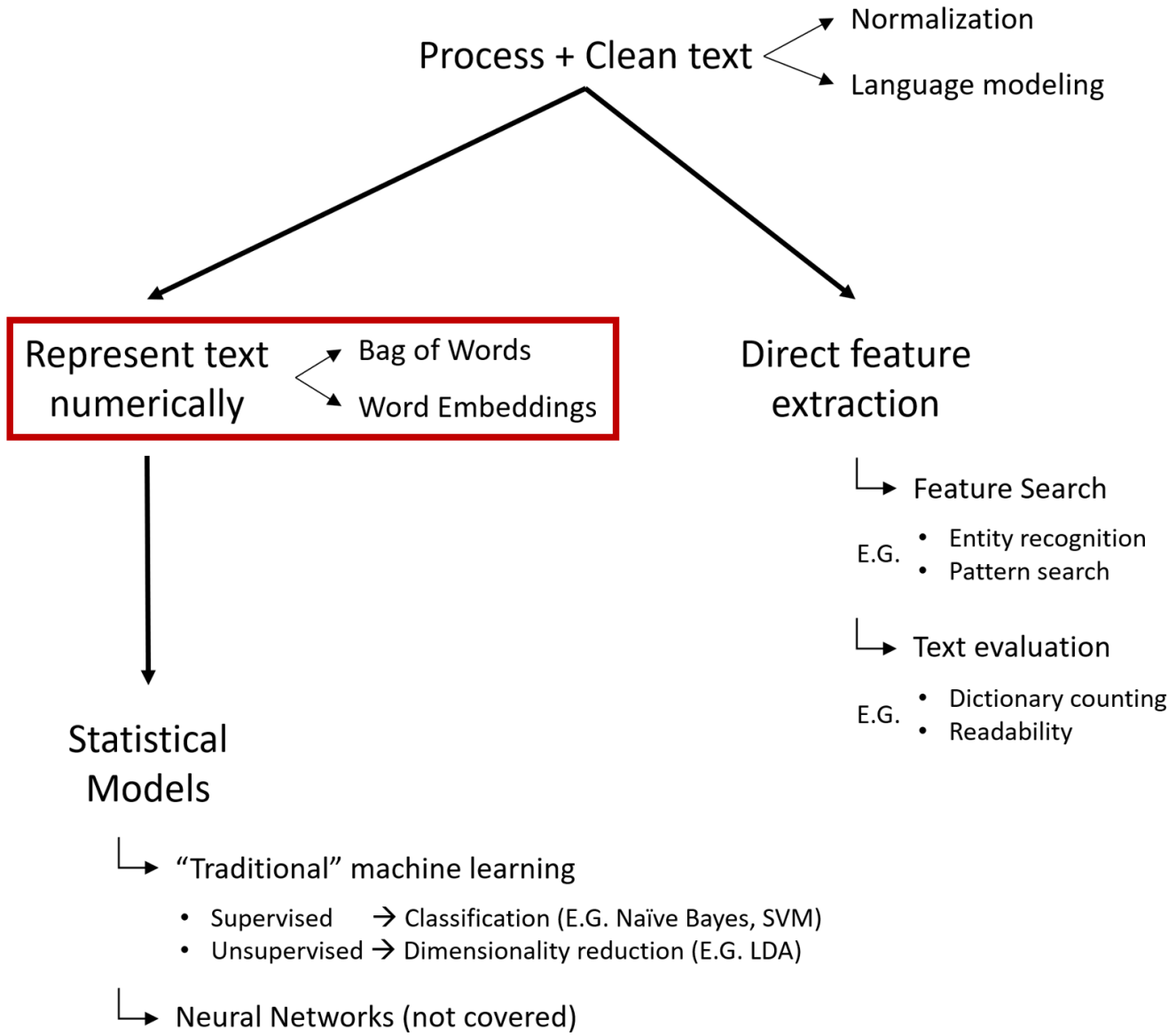
```
In [13]:
from fuzzywuzzy import fuzz

In [14]:
fuzz.ratio("fuzzy wuzzy was a bear", "wuzzy fuzzy was a bear")
Out[14]: 91
```

See the awesome `FuzzyWuzzy` package for details.

Process & Clean

Feature Extraction

Represent Numerically



Process & Clean

Feature Extraction

Represent Numerically

Bag of Words

Also labelled: *frequency based representation*

Term frequency (TF)

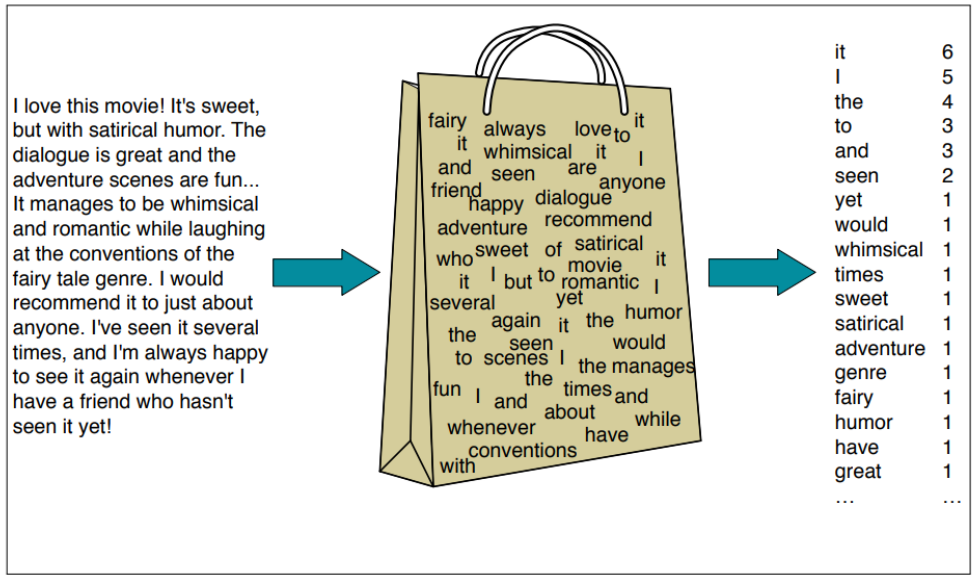


Figure 6.1 Intuition of the multinomial naive Bayes classifier applied to a movie review. The position of the words is ignored (the *bag of words* assumption) and we make use of the frequency of each word.

(Figure taken from: https://web.stanford.edu/~jurafsky/slp3/6.pdf)

Process & Clean

Feature Extraction

Represent Numerically

Term frequency (TF) example:

> [1] "The sky is blue."
>
> [2] "The sun is bright today."
>
> [3] "The sun in the sky is bright."
>
> [4] "We can see the shining sun, the bright sun."

	Docs			
Terms	1	2	3	4
blue	1	0	0	0
bright	0	1	1	1
can	0	0	0	1
see	0	0	0	1
shining	0	0	0	1
sky	1	0	1	0
sun	0	1	1	2
today	0	1	0	0

Note: the collection of all text documents is called the *corpus*

(Example taken from: http://ethen8181.github.io/machine-learning/clustering_old/tf_idf/tf_idf.html)

Process & Clean

Feature Extraction

Represent Numerically

$$w_{x,y} = tf_{x,y} \times \log(\frac{N}{1 + df_x})$$

TF-IDF

Term x within doc y

$tf_{x,y} = frequency\ of\ x\ in\ y$

$df_x = number\ of\ docs\ containing\ x$

$N = number\ of\ documents$

Process & Clean

Feature Extraction

Represent Numerically

$$w_{x,y} = tf_{x,y} \times \log(\frac{N}{1 + df_x})$$

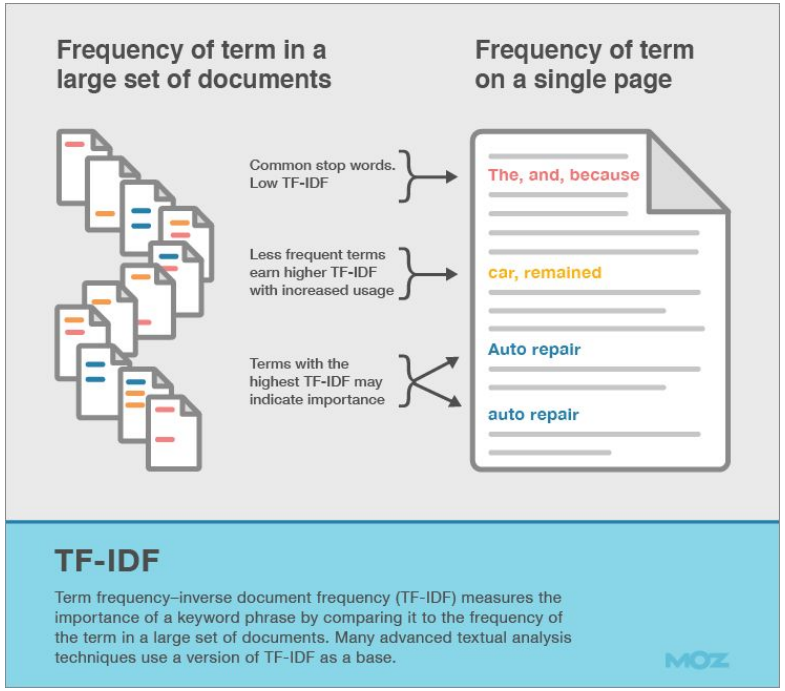
TF-IDF

Term x within doc y

$tf_{x,y} = frequency\ of\ x\ in\ y$

$df_x = number\ of\ docs\ containing\ x$

$N = number\ of\ documents$



(Figure taken from: and https://moz.com/blog/7-advanced-seo-concepts)

Process & Clean

Feature Extraction

Represent Numerically

TF-IDF example:

> [1] "The sky is blue."
>
> [2] "The sun is bright today."
>
> [3] "The sun in the sky is bright."
>
> [4] "We can see the shining sun, the bright sun."

	DF
blue	1
bright	3
can	1
see	1
shining	1
sky	2
sun	3
today	1

Docs	blue	bright	can	see	shining	sky	sun	today
1	0.69	0.00	0.00	0.00	0.00	0.29	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.69
3	0.00	0.00	0.00	0.00	0.00	0.29	0.00	0.00
4	0.00	0.00	0.69	0.69	0.69	0.00	0.00	0.00

(Example taken from: http://ethen8181.github.io/machine-learning/clustering_old/tf_idf/tf_idf.html)

Process & Clean

Feature Extraction

Represent Numerically

Word Embeddings

Are there alternatives to the frequency based representation?

- Yes, meet the new "secret sauce": **word embeddings**!

Process & Clean

Feature Extraction

Represent Numerically

Word Embeddings

Are there alternatives to the frequency based representation?

- Yes, meet the new "secret sauce": **word embeddings**!

Word embeddings are based on a "prediction based representation".

Basic idea:

> A word is characterized by the company it keeps:
>
> 1. A **Ferrari** is a fast car
> 2. A **Lamborgini** is a fast car

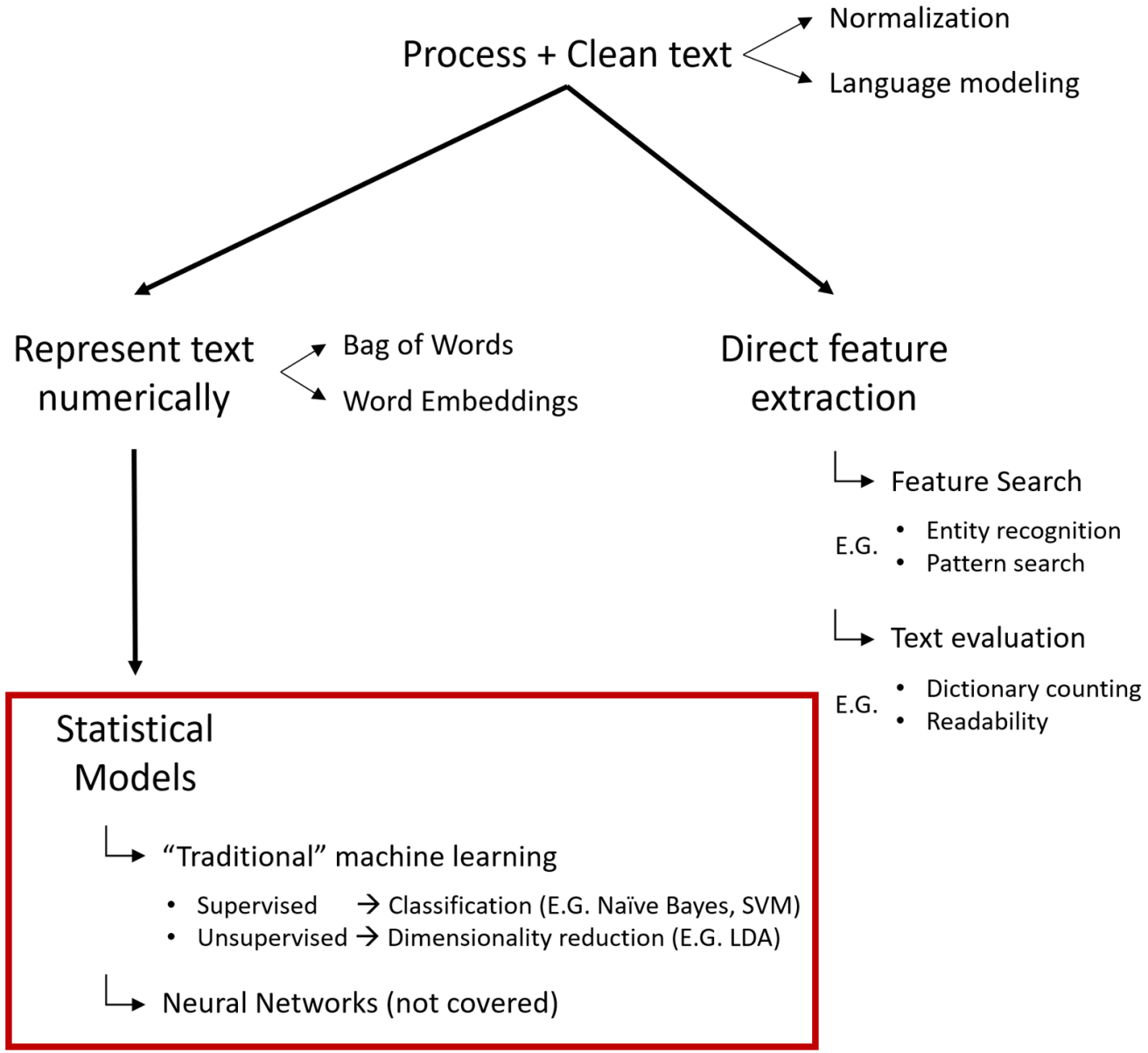
Notes: the most well-known adaptation is `Word2Vec`. Word embeddings are sometimes called *Continous Bag of Words*

Process & Clean

Feature Extraction

Represent Numerically

Machine Learning



Process & Clean

Feature Extraction

Represent Numerically

Machine Learning

What is Machine Learning?

> A machine learning algorithm is not explicitly programmed.
> Instead, the algorithm is trained based on the input + output data.

Does this sound familiar?

Process & Clean

Feature Extraction

Represent Numerically

Machine Learning

What is Machine Learning?

> A machine learning algorithm is not explicitly programmed.
> Instead, the algorithm is trained based on the input + output data.

Does this sound familiar?

- A linear regression is also machine learning!

Process & Clean

Feature Extraction

Represent Numerically

Machine Learning

What is Machine Learning?

> A machine learning algorithm is not explicitly programmed.
> Instead, the algorithm is trained based on the input + output data.

Does this sound familiar?

- A linear regression is also machine learning!

Example: sentiment analysis

Traditional method:

- manually create pos/neg word lists

Machine learning method:

- manually classify sentence pos/neg score
- pos/neg word lists determined by algorithm

Process & Clean

Feature Extraction

Represent Numerically

Machine Learning

Supervised Machine Learning

> Supervised ML algorithms are trained on classified training data.

Process & Clean

Feature Extraction

Represent Numerically

Machine Learning

Supervised Machine Learning

> Supervised ML algorithms are trained on classified training data.

Where to get training data?

1. Use a naturally classified training set
    - News categories
    - Movie reviews
    - Text books for different levels of English
2. Create your own training set
    - Manually classify text
    - Crowdsource a training set

Process & Clean

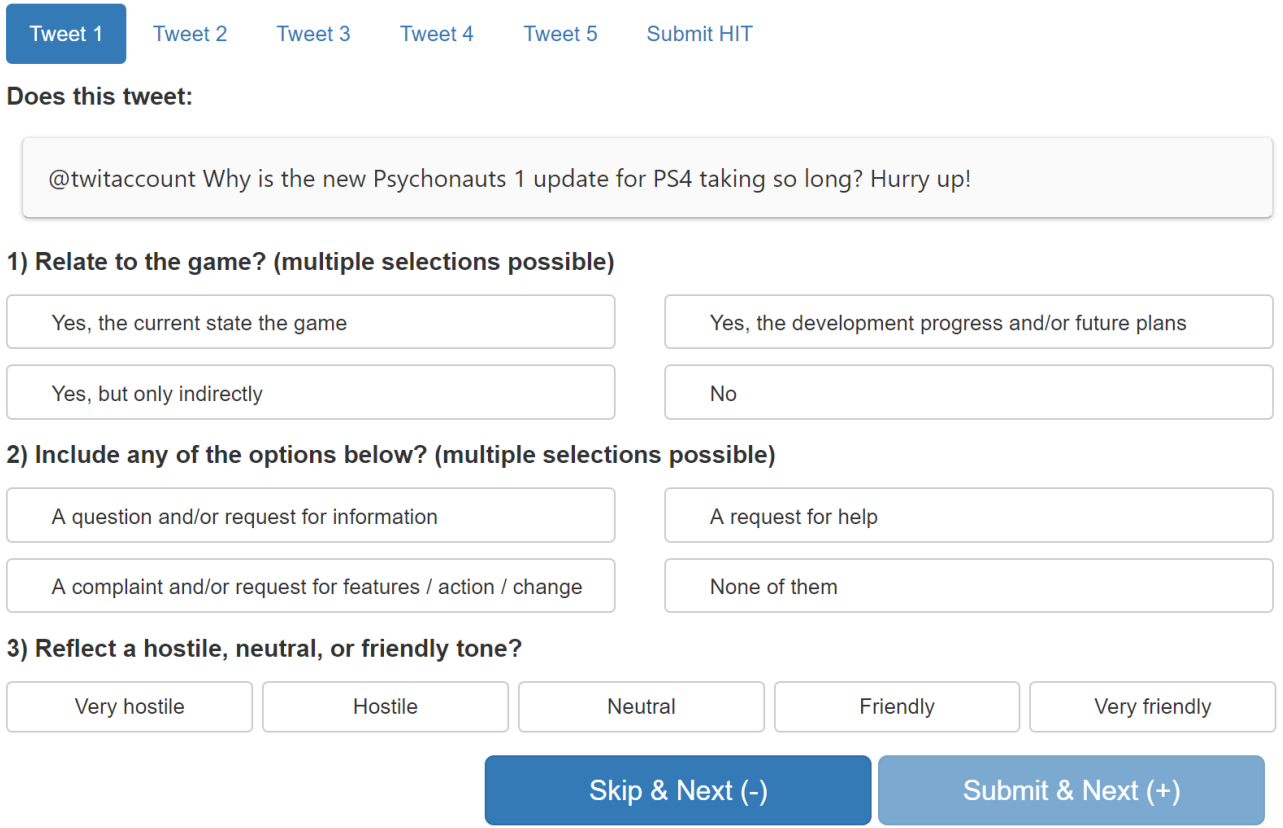
Feature Extraction

Represent Numerically

Machine Learning

Crowdsource training set

It is possible to crowd source a training set using services like Amazon Mechanical Turk.



Process & Clean

Feature Extraction

Represent Numerically

Machine Learning

Supervised Machine Learning: models

Three often used models for Supervised ML:

1. Naive Bayes classifier (sklearn link)
2. SVM: Support Vector Machines (sklearn link)
3. Decision Trees (sklearn link)

Process & Clean

Feature Extraction

Represent Numerically

Machine Learning

Supervised Machine Learning: models

Three often used models for Supervised ML:

1. Naive Bayes classifier (sklearn link)
2. SVM: Support Vector Machines (sklearn link)
3. Decision Trees (sklearn link)

My recommendation?

Always try multiple models to see which gives you the best results.

- Naive Bayes is good for small samples and quick testing.
- SVM is more sophisticated, generally better for more complex models.
- Decision Trees are more intuitive but harder to train.

Regardless of the model:

- hyperparameter optimization is very important!

Process & Clean

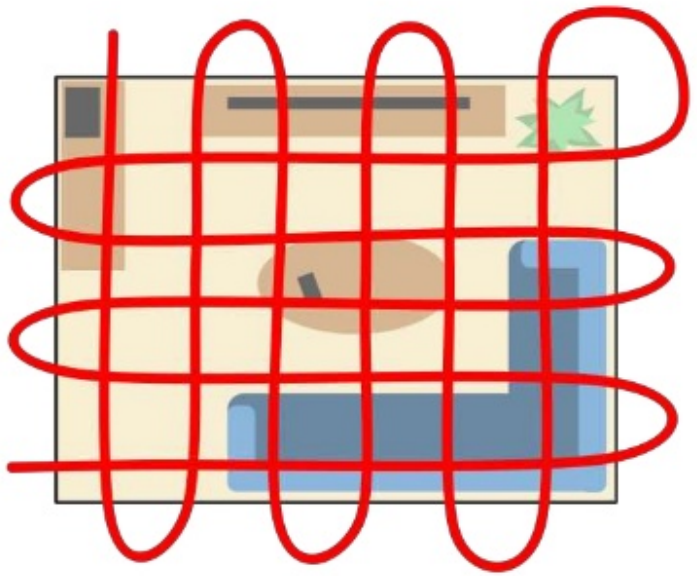
Feature Extraction

Represent Numerically

Machine Learning

What is Hyperparameter Optimzation?

- "… is the problem of choosing a set of hyperparameters for a learning algorithm, …" [1]
- **Grid search**
- Random search
- …



5 ©2015 IBM Corporation 1 November 2016

https://openclipart.org/detail/194603/grid-search-pattern

IBM

(Slide taken from: https://www.slideshare.net/sparktc/hyperparameter-optimization-sven-hafeneger)

Process & Clean

Feature Extraction

Represent Numerically

Machine Learning

Model Selection and Evaluation

> i.e. how to select the model and hyperparameters?

There are two essential metrics in ML:

1. Precision

   > High precision --> low false positive rate

2. Recall

   > High recall --> low false negative rate

For details see: Precision-Recall

Process & Clean

Feature Extraction

Represent Numerically

Machine Learning

Unsupervised Machine Learning

> Unsupervised ML algorithms are trained using only input data.

Do unsupervised ML models work for all problems?

- No! Usually only for clustering / topic modelling.

Process & Clean

Feature Extraction

Represent Numerically

Machine Learning

Unsupervised Machine Learning

> Unsupervised ML algorithms are trained using only input data.

Do unsupervised ML models work for all problems?

- No! Usually only for clustering / topic modelling.

Examples of unsupervised models:

1. Principal Component Analysis / Factor Analysis
2. **Latent Dirichlet Allocation (LDA)**

(and Word2Vec)

Process & Clean

Feature Extraction

Represent Numerically

Machine Learning

Latent Dirichlet Allocation (LDA)

> Unsupervised topic model technique to discover abstract topics from a collection of documents.

Process & Clean

Feature Extraction

Represent Numerically

Machine Learning

Latent Dirichlet Allocation (LDA)

> Unsupervised topic model technique to discover abstract topics from a collection of documents.

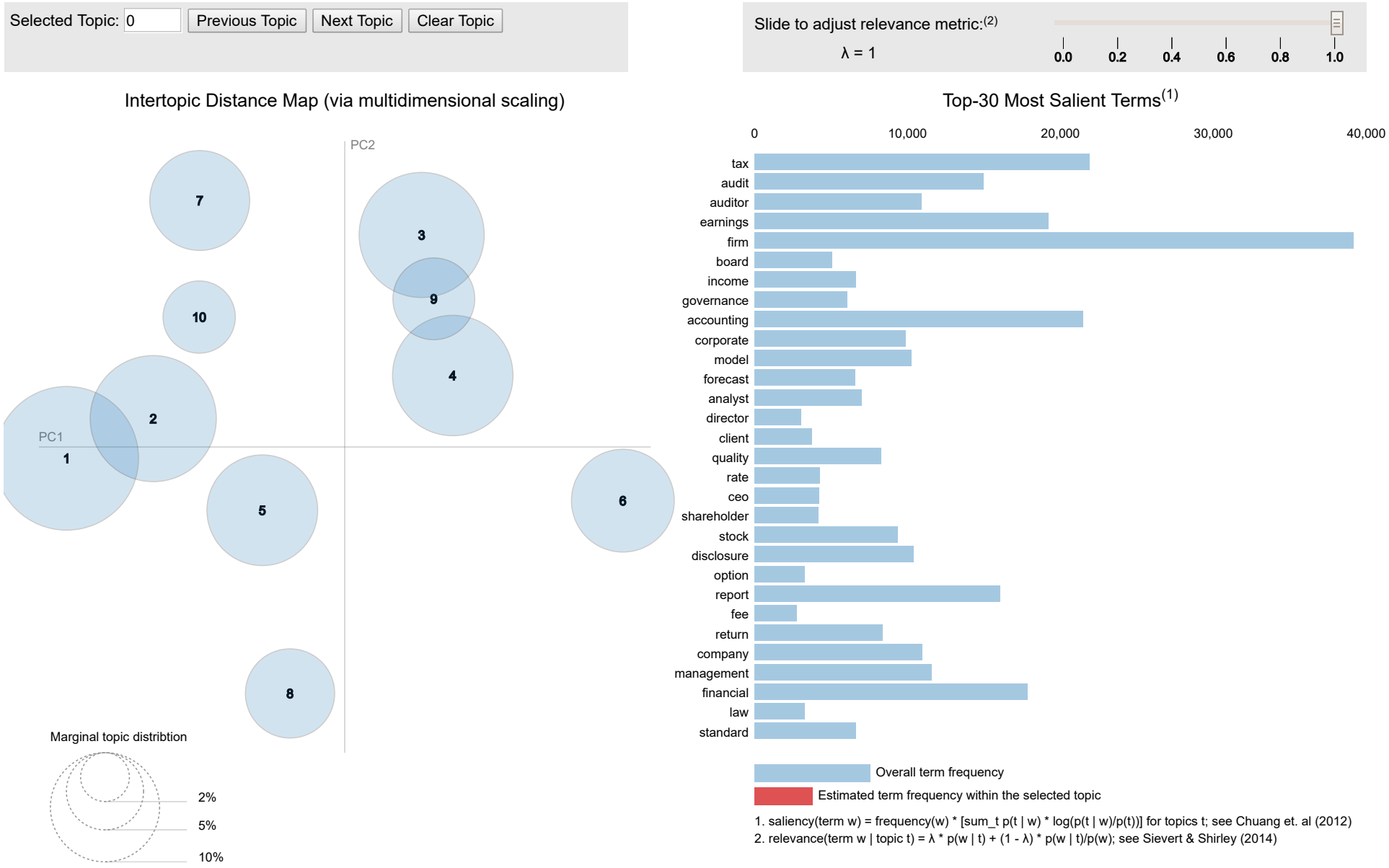
LDA procedure

You define the number of topics (*N*) and the other hyperparameters.

LDA then assigns each document a vector with *N* topic probabilities.

Important: topics are not labeled and there is a degree of randomness

> i.e. running the same model twice can result in different topic labels!


Selected Topic: 0
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0
0.2
0.4
0.6
0.8
1.0
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
4
5
6
7
8
9
10
Marginal topic distribtion
2%
5%
10%
Top-30 Most Salient Terms(1)
0
10,000
20,000
30,000
40,000
tax
audit
auditor
earnings
firm
board
income
governance
accounting
corporate
model
forecast
analyst
director
client
quality
rate
ceo
shareholder
stock
disclosure
option
report
fee
return
company
management
financial
law
standard
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Process & Clean

Feature Extraction

Represent Numerically

Machine Learning

Neural Networks

Neural Networks for NLP

Natural Language is very complex, NLP is hard:

1. The Pope's baby steps on gays
2. Scientists study whales from space
3. Boy paralyzed after tumor fights back to gain black belt

(Examples from: http://web.stanford.edu/class/cs224n/lectures/cs224n-2017-lecture1.pdf)

Process & Clean

Feature Extraction

Represent Numerically

Machine Learning

Neural Networks

Neural Networks for NLP

Natural Language is very complex, NLP is hard:

1. The Pope's baby steps on gays
2. Scientists study whales from space
3. Boy paralyzed after tumor fights back to gain black belt

(Examples from: http://web.stanford.edu/class/cs224n/lectures/cs224n-2017-lecture1.pdf)

Deep Neural Networks

DNN allow to model complex phenomenon, promising progress for NLP!

Interested? Check out the Stanford course CS224n (Syllabus)!

Notebook

How to get started with NLP and Python?

- Take a look at my NLP repository!

Github.com/TiesdeKok/Python_NLP_Tutorial

Notebook

Closing remarks

Questions?

Setup:

1. Make sure you have Spacy + Language models installed

Get Started:

Goal: Perform basic operations on an example text.

1. Open up a notebook in the `Materials` folder.
2. Import `os` and `re`
3. Load up the text file: `day_3 > mini_task > data > example_text.txt`
4. Count the words "Holmes" and "Watson" using Python.
5. Create a Regular Expression to capture all names following "Mr."
6. Load the text in using Spacy and split into sentences.

For help: see notebook on NLP with Python

Mini Task

Goal: Get hands-on experience with basic NLP tasks.

Instructions

1. Open (start) a Jupyter Notebook in the `Materials` folder
2. Solve tasks in: `day_3 > mini_task > nlp_mini_task.ipynb`

For help:

- Python tutorial
- Python Basics Notebook
- Data handling with Pandas
- NLP with Python