UNIVERSITÄT MANNHEIM

Data Mining I
Text Mining

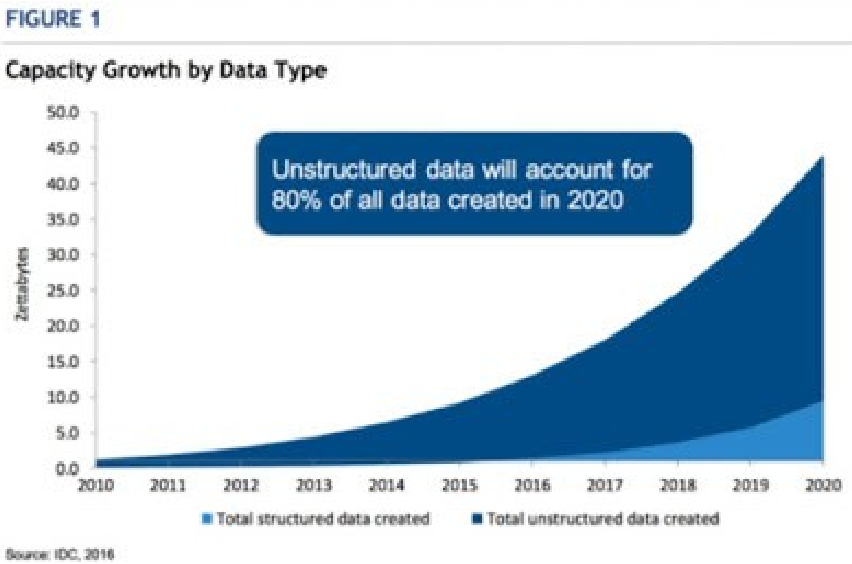
Heiko Paulheim

Outline

1) What is Text Mining?
2) Text Preprocessing
3) Feature Creation
4) Feature Selection
5) Pattern Discovery
6) Processing Text from Social Media

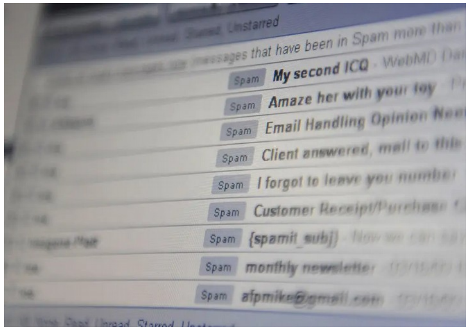
Motivation for Text Mining

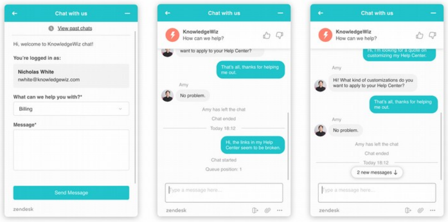
- Structured data: databases, excel sheets, XML, …
- Unstructured data: text, images, audio, video, ...



Motivation for Text Mining

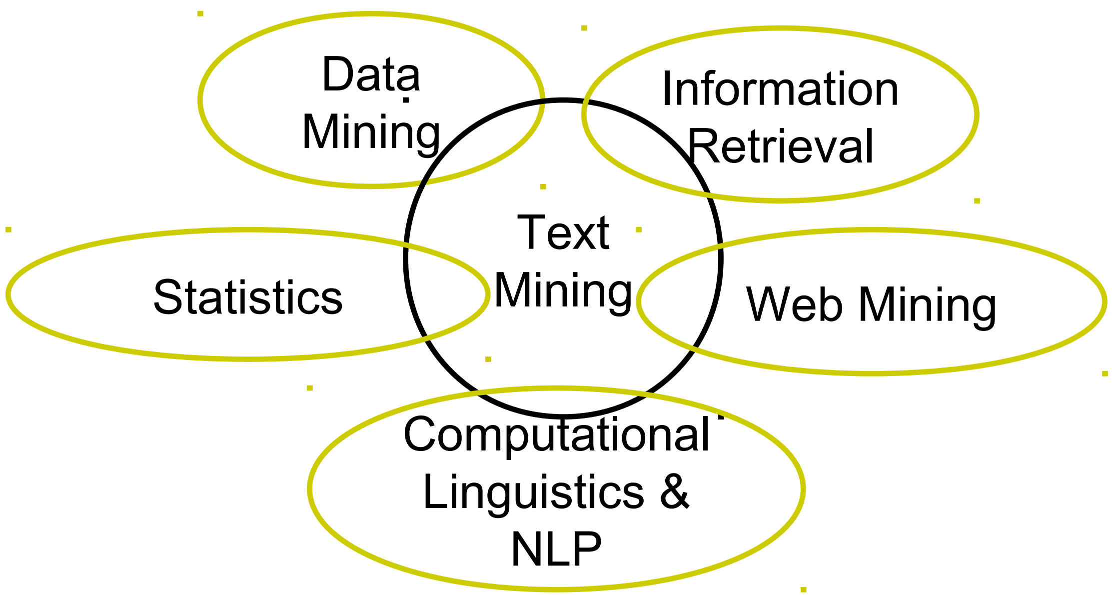
- A lot of unstructured data is text, e.g.,
  - Web pages
  - E-mails
  - Chat conversations
  - Technical documents
  - Corporate documents
  - Digital libraries





Text Mining

- The extraction of implicit, previously unknown and potentially useful information from a large amount of **textual resources**



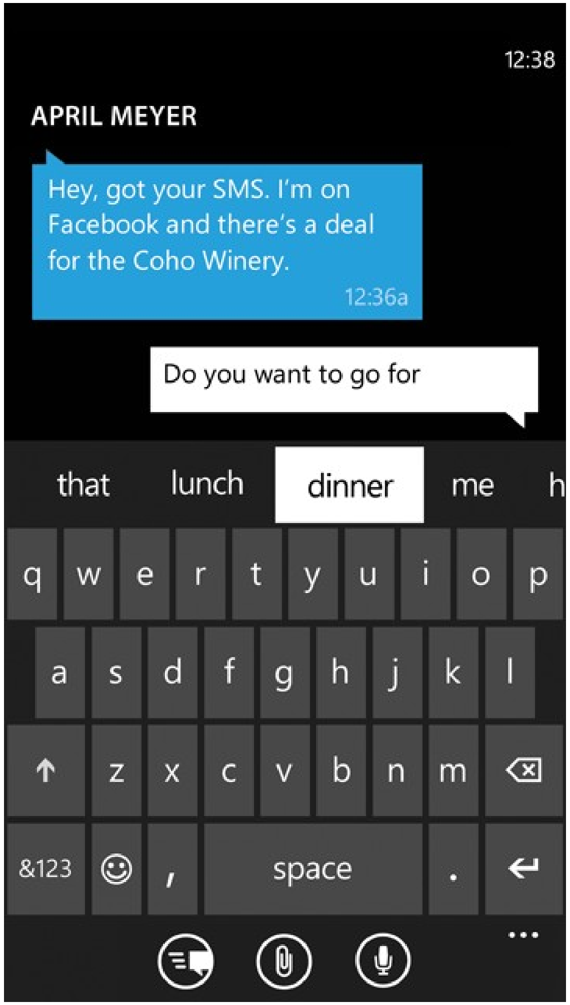
Search Versus Discovery

	Search/Query (Goal-oriented)	Discovery (Opportunistic)
Structured Data	Query Processing	Data Mining
Text	Information Retrieval	Text Mining

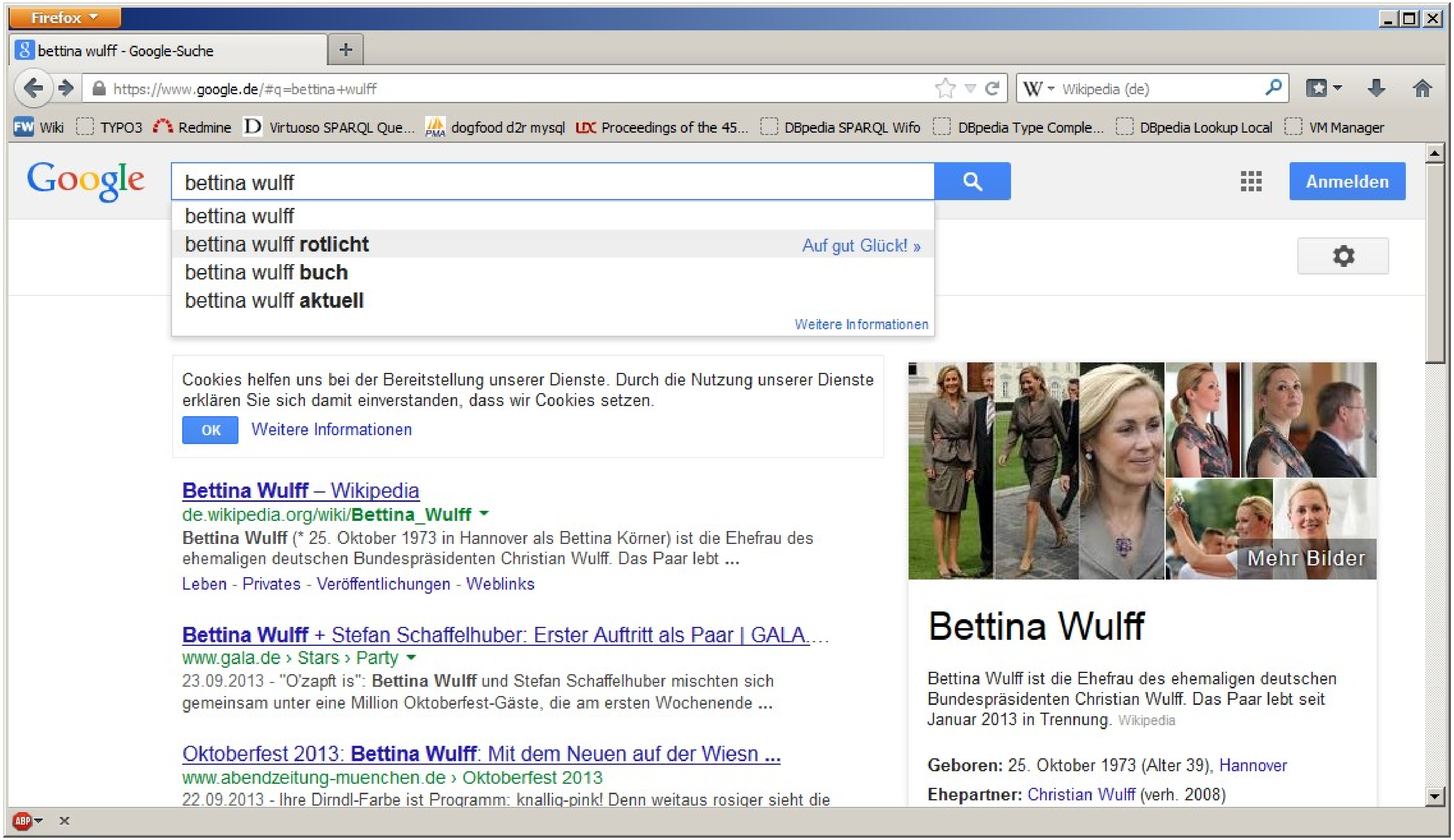
Typical Text Mining Applications

- Classification and clustering of news stories or web pages
- Email and news filtering / Spam detection
  - Also: fake review classification
- Sentiment Analysis
- Query suggestion / auto complete
- Gain insights about relations between people, places or organizations described in a document corpus

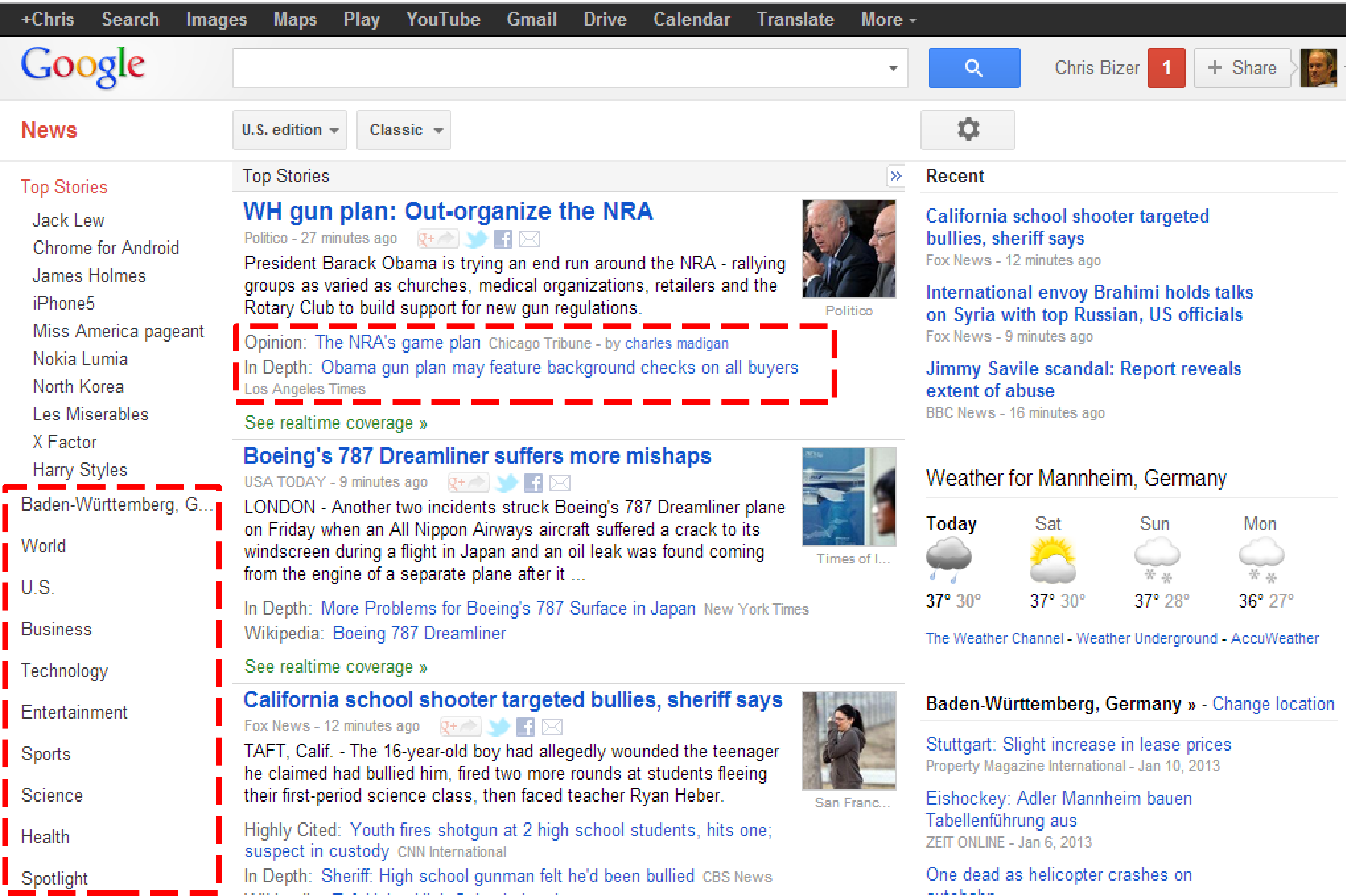
Examples


12:38
APRIL MEYER
Hey, got your SMS. I'm on Facebook and there's a deal for the Coho Winery.
12:36a
Do you want to go for
that
lunch
dinner
me
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
&123
space

Example: Search Query Completion

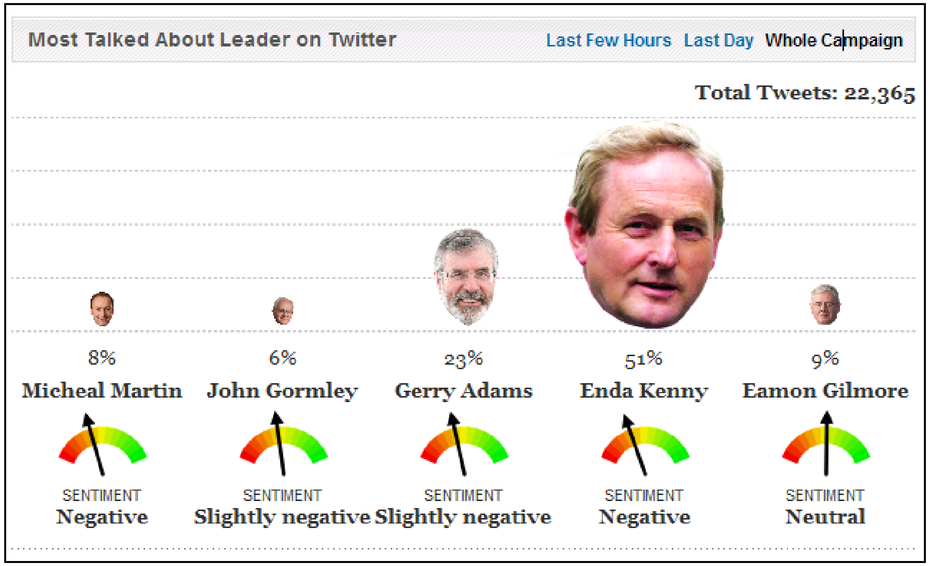


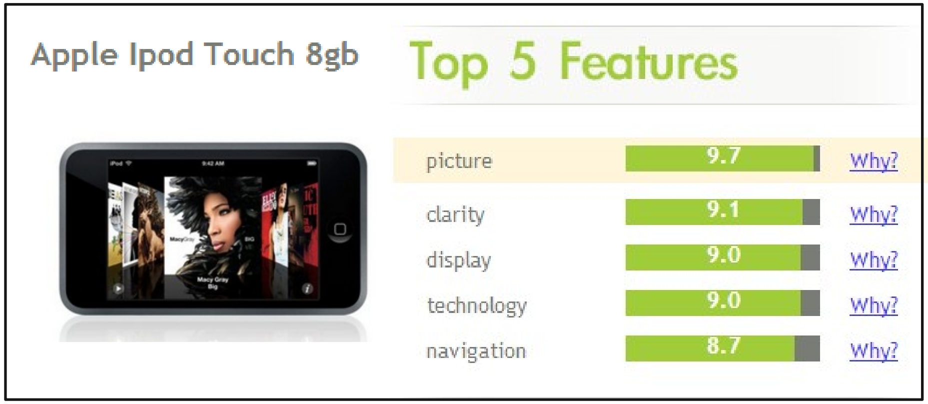
Example: Search Result Organization



Example: Sentiment Analysis

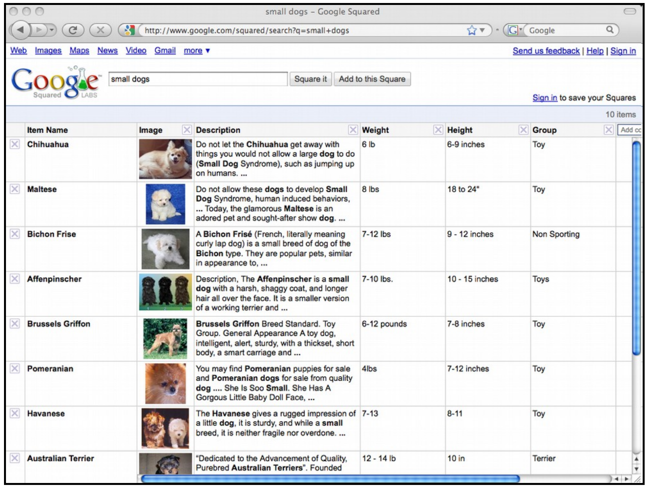
- Determine *polarity*
  - Polarity values, e.g.:
    - positve, neutral, negative
    - likert scale (1 to 10)
  - Application examples
    - Document level
      - analysis of tweets about politicians
    - Feature/aspect level
      - analysis of product reviews

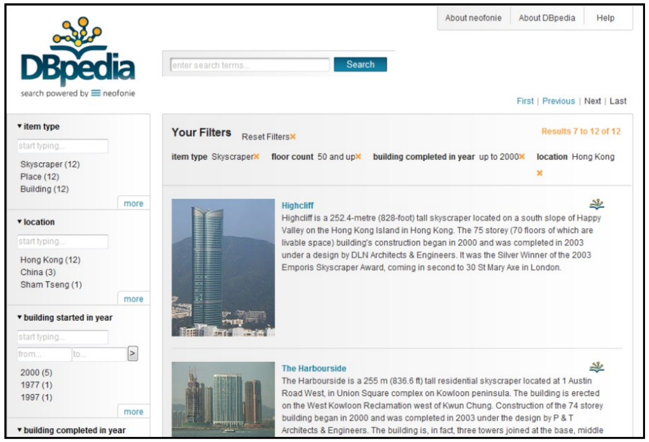




Example: Information Extraction

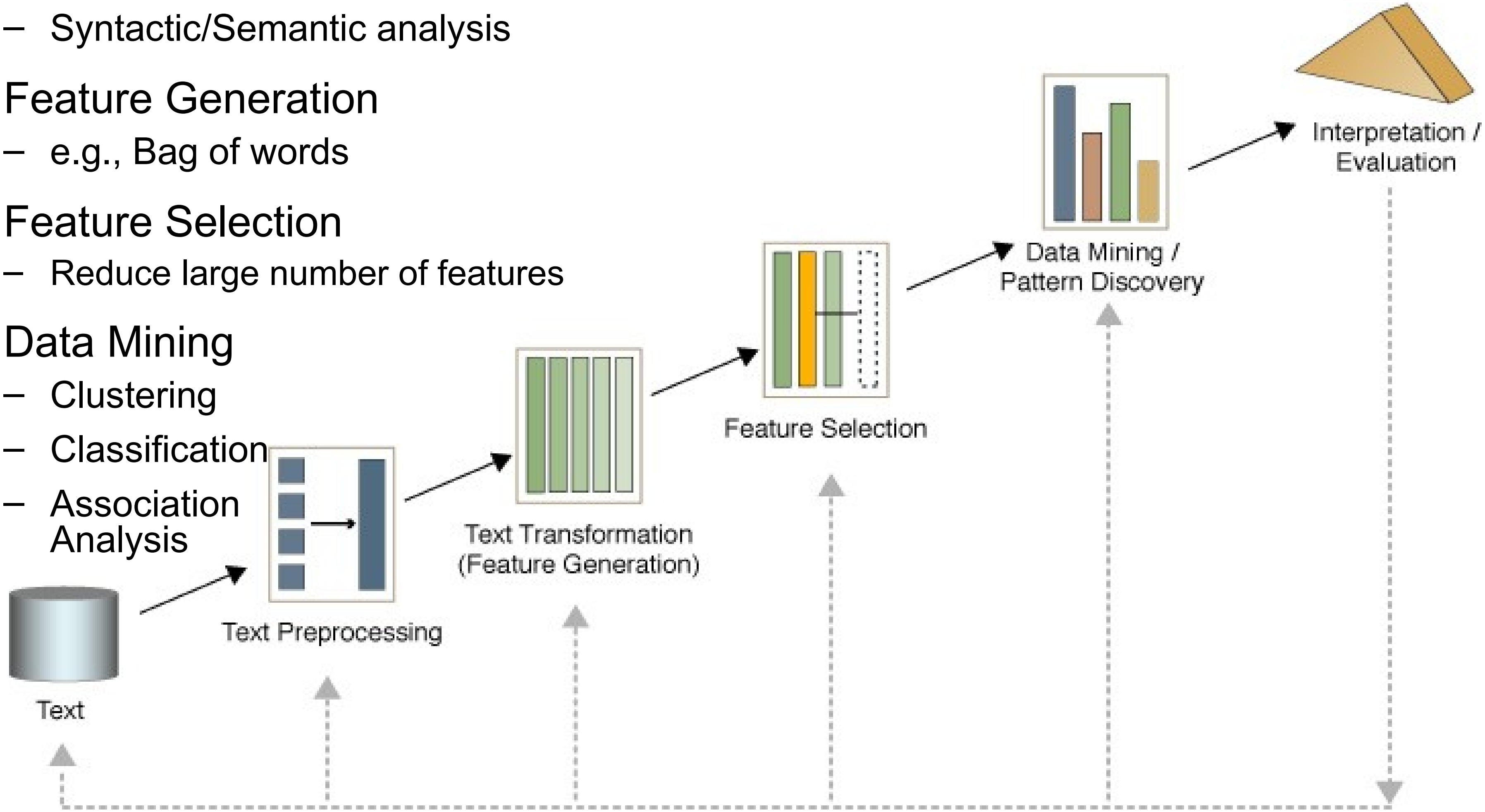
- Automatically extracting structured information from documents
- Subtasks
  - Named Entity Recognition and Disambiguation
    - “The parliament in Berlin has decided …“
    - Which parliament? Which Berlin?
  - Relationship Extraction
    - PERSON works for ORGANIZATION
    - PERSON located in LOCATION
  - Fact Extraction
    - CITY has population NUMBER
    - COMPANY has turnover NUMBER [Unit]





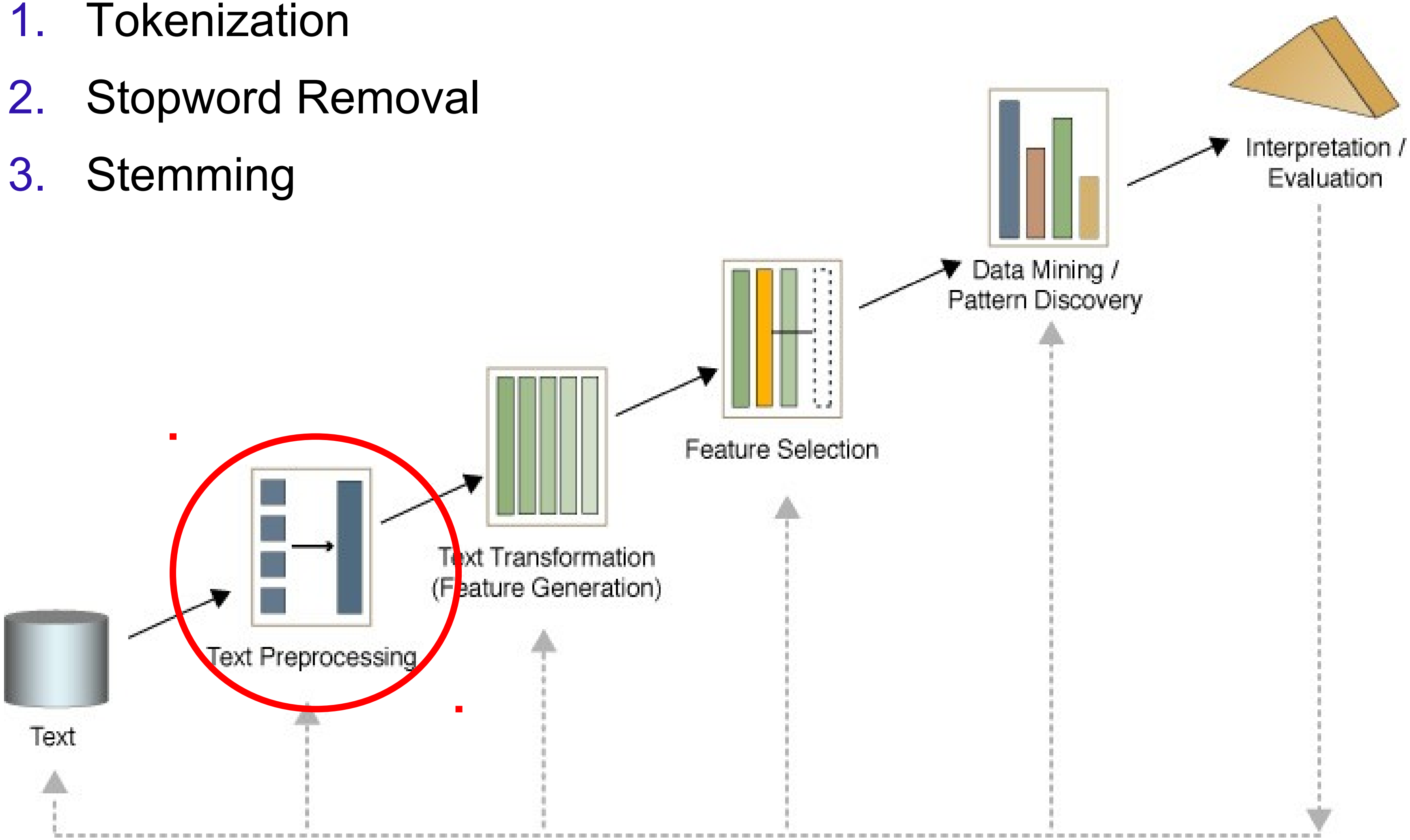
The Text Mining Process

1. Text Preprocessing
   – Syntactic/Semantic analysis
2. Feature Generation
   – e.g., Bag of words
3. Feature Selection
   – Reduce large number of features
4. Data Mining
   – Clustering
   – Classification
   – Association Analysis



Text Preprocessing

1. Tokenization
2. Stopword Removal
3. Stemming



Syntactic / Linguistic Text Analysis

- Simple Syntactic Analysis
  - Text Cleanup (remove punctuation, HTML tags, …)
  - Normalize case
  - Tokenization (break text into single words or N-grams)
- Advanced Linguistic Analysis
  - Word Sense Disambiguation
    - Determine which sense a word is having
    - Normalize synonyms (United States, USA, US)
    - Coreference resolution – normalize pronouns (he, she, it)
  - Part Of Speech (POS) tagging
    - Parse sentences according to grammar
    - Determine function of each term
    - e.g. John (noun) gave (verb) the (det) ball (noun).

Synonym Normalization & Spelling Correction

- Usually using catalogs
  - such as WordNet

- Example for a large-scale catalog
  - Wikipedia Surface Forms
- Normalized forms: titles of Wikipedia pages
  - e.g., “United States Armed Forces”
- Other forms: anchor texts of links to that page
  - “The music of Nine Inch Nails has reportedly been used by the U.S. military as music torture to break down the resolve of detainees.”

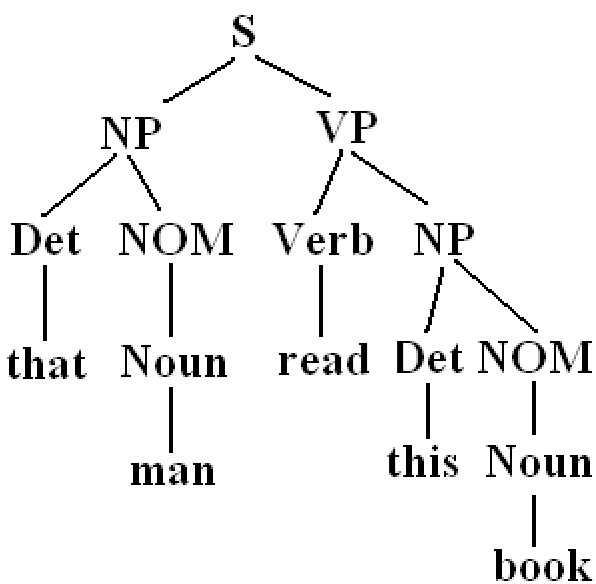
Extracted normalization pattern:
“U.S. military” → “United States Armed Forces”

Synonym Normalization & Spelling Correction

- Catalogs work great for common knowledge
  - not so well for special domains
- Possible remedy: string similarity
- Example: edit distance
  - Notion: the minimum number of edits needed to transform one string into the other
  - Allowed edit operations:
    - insert a character into the string
    - delete a character from the string
    - replace one character with a different character
- Examples:
  - levenshtein('John Smith', 'John K. Smith ') = 3 (3 inserts)
  - levenshtein('John Smith', 'Jack Smith') = 3 (3 substitutions)

POS Tagging

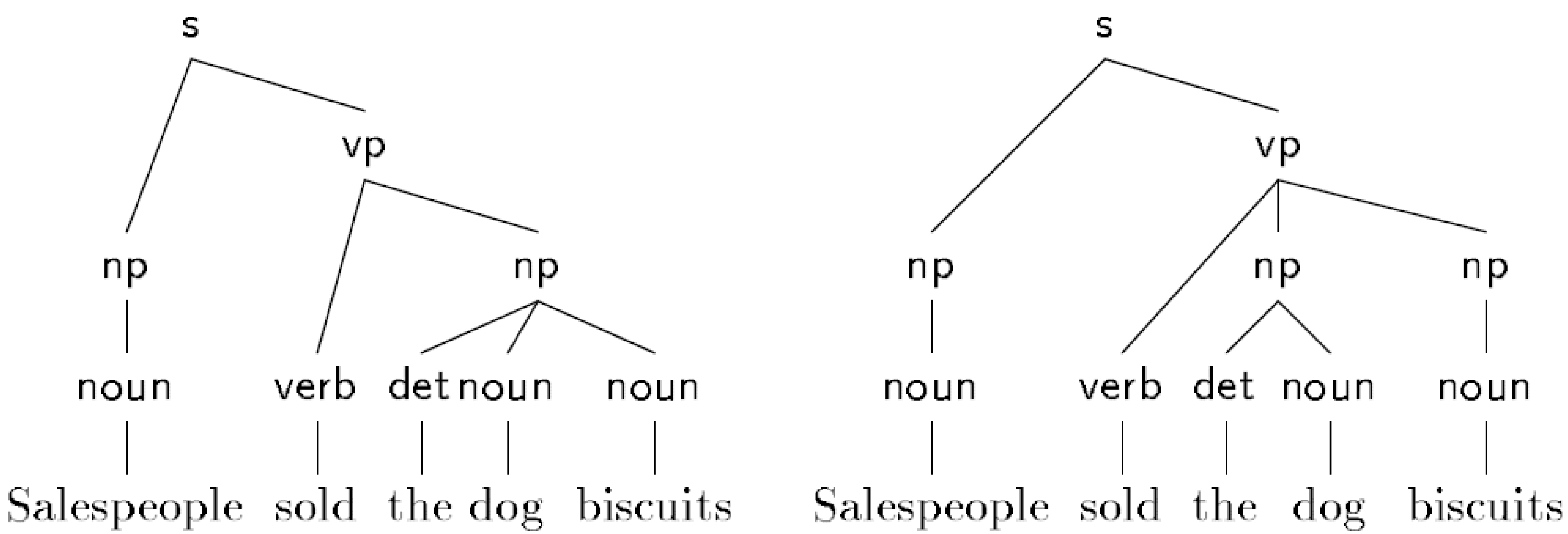
- Task
  - determining word classes and syntactic functions
  - finding the structure of a sentence



http://cs.oberlin.edu/~jdonalds/333/lecture12.html

POS Tagging

- Sometimes, multiple results are possible
  - language is ambiguous!



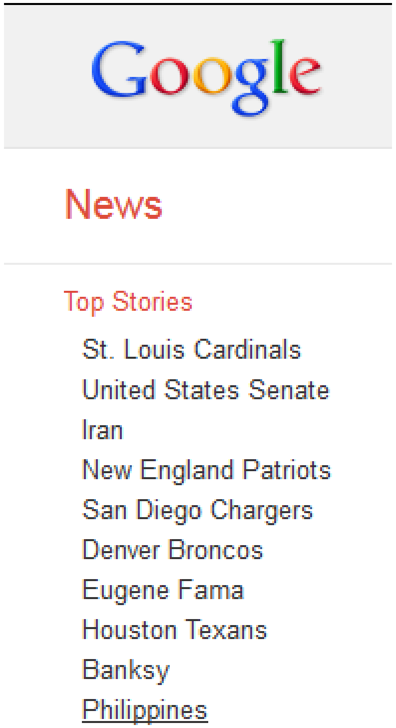
Charniak: Statistical techniques for natural language parsing (1997)

POS Tagging

- Supervised approach
  - Use an annotated corpus of text
  - i.e., a set of sentences with human-created POS tags
- Note: words may have different functions in different contexts
  - *I move (VERB) to Mannheim next year.*
  - *He made a clever move (NOUN).*
- Naive Algorithm by Charniak (1997)
  - Use the most common tag for each word
  - Assign NOUN to every unknown word
  - Result: 90% accuracy, using a training corpus of 300,000 words

POS Tagging

- Simple algorithm for key phrase extraction
  - e.g., annotation of text corpora
- Use all NP of the form ADJ+NOUN*
- Example sentence:
  - *Text mining refers to the process of deriving high-quality information from text.*
- Key phrases:
  - *text mining* (NOUN+NOUN)
  - *process* (NOUN)
  - *high-quality information* (ADJ NOUN NOUN)
  - *text* (NOUN)



Stop Words Removal

- Many of the most frequent words are likely to be useless
- These words are called *stop words*
  - examples (English): *the, of, and, to, an, is, that, …*
  - typically text contains about 400 to 500 such words
  - additional domain specific stop words lists may be constructed
- Why should we remove stop words?
  - Reduce data set size
    - stop words account for 20-30% of total word counts
  - Improve efficiency and effectiveness
    - stop words may confuse the mining algorithm

More Examples of Stopwords

a about above across after again against all almost alone along already also although always am among an and another any anybody anyone anything anywhere are area areas aren't around as ask asked asking asks at away b back backed backing backs be became because become becomes been before began behind being beings below best better between big both but by c came can cannot can't case cases certain certainly clear clearly come could couldn't d did didn't differ different differently do does doesn't doing done don't down downed downing downs during e each early either end ended ending ends enough even evenly ever every everybody everyone everything everywhere f face faces fact facts far felt few find finds first for four from full fully further furthered furthering furthers g gave general generally get gets give given gives go going good goods got great greater greatest group grouped grouping groups h had hadn't has hasn't have haven't having he he'd he'll her here here's hers herself he's high higher highest him himself his how however how's i i'd if i'll i'm important in interest interested interesting interests into is isn't it its it's itself i've j just k keep keeps kind knew know known knows l large largely last later latest least less let lets let's like likely long longer longest m made make making man many may me member members men might more most mostly mr mrs much must mustn't my myself n necessary need needed needing needs never new newer newest next no nobody non noone nor not nothing now nowhere number numbers o of off often old older oldest on once one only open opened opening opens or order ordered ordering orders other others ought our ours ourselves out over own p part parted parting parts per perhaps place places point pointed pointing points possible present presented presenting presents problem problems put puts q quite r rather really right room rooms s said same saw say says second seconds see seem seemed seeming seems sees several shall shan't she she'd she'll she's should shouldn't show showed showing shows side sides since small smaller smallest so some somebody someone something somewhere state states still such sure t take taken than that that's the their theirs them themselves then there therefore there's these they they'd they'll they're they've thing things think thinks this those though thought thoughts three through thus to today together too took toward turn turned turning turns two u under until up upon us use used uses v very w want wanted wanting wants was wasn't way ways we we'd well we'll wells went were we're weren't we've what what's when when's where where's whether which while who whole whom who's whose why why's will with within without won't work worked working works would wouldn't x y year years yes yet you you'd you'll

Stopword Removal

- Note: words may have different functions in different contexts
  - *He can (AUX VERB) read.*
  - *The can (NOUN) will rust.*

- After removing stopwords naively
  - “can” is removed
  - We cannot find out that the text is about cans
  - We cannot query for texts about cans
  - etc.

POS Tagging Revisited

- Improvement over naïve algorithm
  - respect *transition probabilities*

The	can	will	rust
det	modal-verb	**modal-verb**	noun
	noun	noun	**verb**
	verb	verb	

- Improves accuracy to 96-97%
- Upper limit: 98%

Charniak: Statistical techniques for natural language parsing (1997)

Stemming

- Techniques to find out the root/stem of a word.
  - Words: User, users, used, using → Stem: use
  - Words: Engineering, engineered → Stem: engineer
- Usefulness for Text Mining
  - improve effectiveness text mining methods
    - matching similar words
  - reduce term vector size
    - combing words with same roots may reduce indexing size as much as 40-50%

Lookup-based Stemming

- Create a lookup table with all inflected forms
  - e.g. WordNet, Wiktionary

- Example:

Base Form	Inflected Forms
move	moves, moved, moving
go	goes, went, gone, going
apple	apples
...	...

Rule-based Stemming

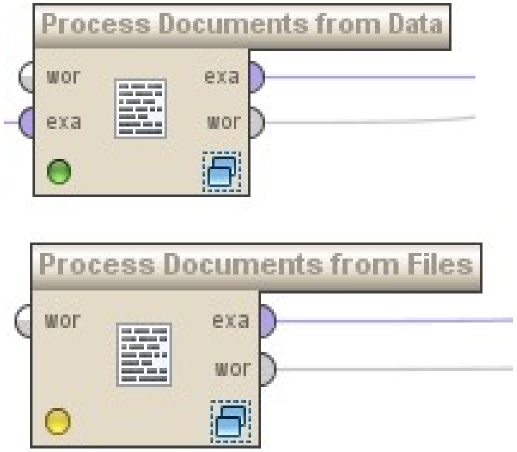
- remove endings
  - if a word ends with a consonant other than *s*, followed by an *s*, then delete *s* (*puts* → *put*)
  - if a word ends in *es*, drop the *s* (*uses* → *use*)
  - if a word ends in *ing*, delete the *ing* unless the remaining word consists only of one letter or of *th* (*reading* → *read*)
  - If a word ends with *ed*, preceded by a consonant, delete the *ed* unless this leaves only a single letter (*founded* → *found*)
  - …
- transform words
  - if a word ends with *ies* but not *eies* or *aies* then *ies* → *y* (*flies* → *fly*)

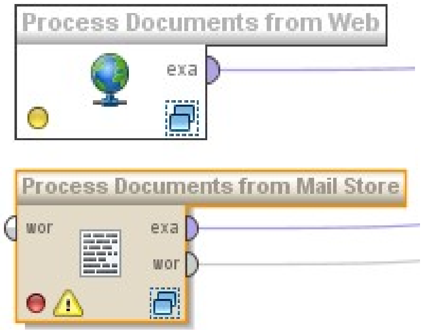
Stemming Comparison

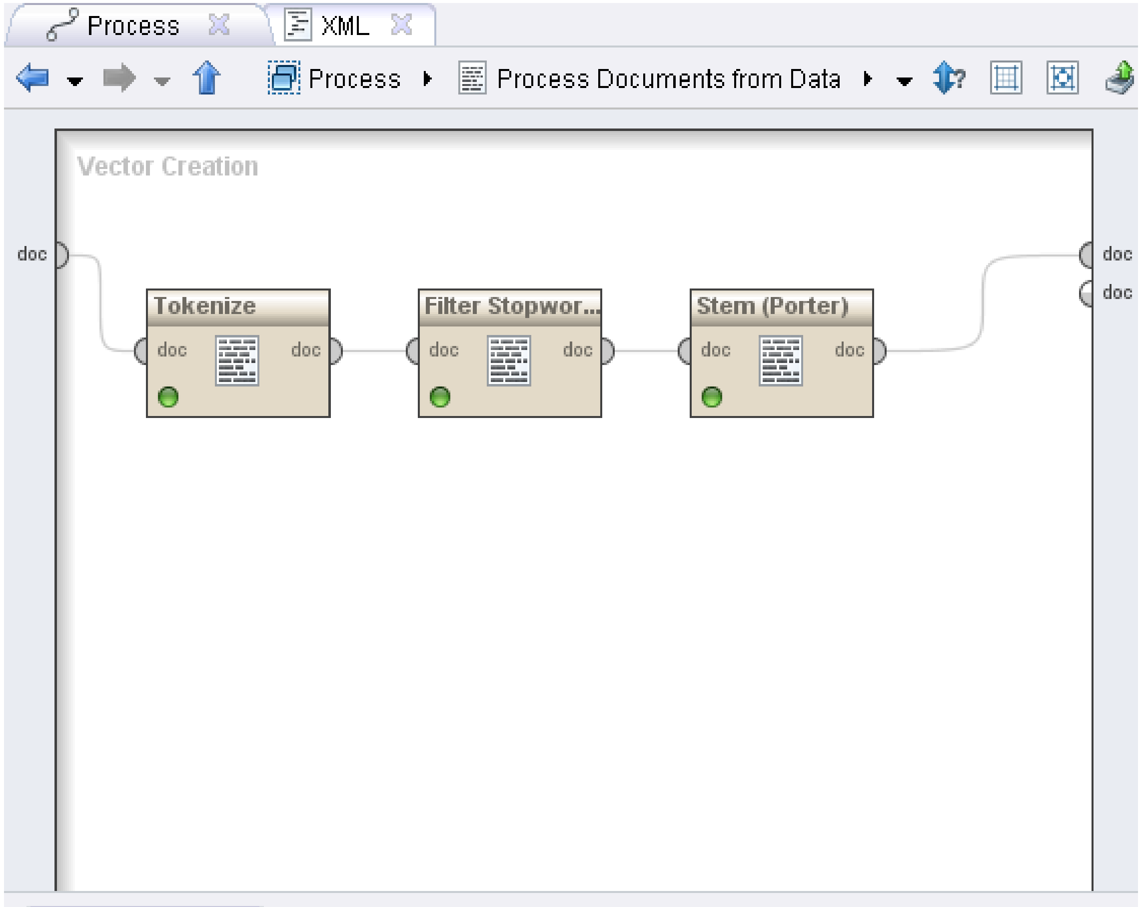
- Lookup tables
  - are accurate
  - exceptions are handled easiliy (e.g., *went* → *go*)
  - consume much space, in particular for highly inflected languages (e.g., Latin, Greek, Spanish, Baltic languages)
- Rule-based stemming
  - low space consumption
  - works for emerging words without update (e.g., *iPads* → *iPad*)
  - prone to *overstemming* errors, e.g.
    - *sling* → *sl*
    - *sled* → *sl*

Preprocessing Operators in RapidMiner

- To use the operators, you need to install the Text Processing Extension first.







Text Preprocessing in Python

Simple preprocessing in sklearn:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.datasets import load_files

# Load documents
docs = load_files('directory_of_files',encoding='utf-8')

# Vectorize documents
vectorizer = CountVectorizer(analyzer='word', stop_words='english')
matrix = vectorizer.fit_transform(docs)
```

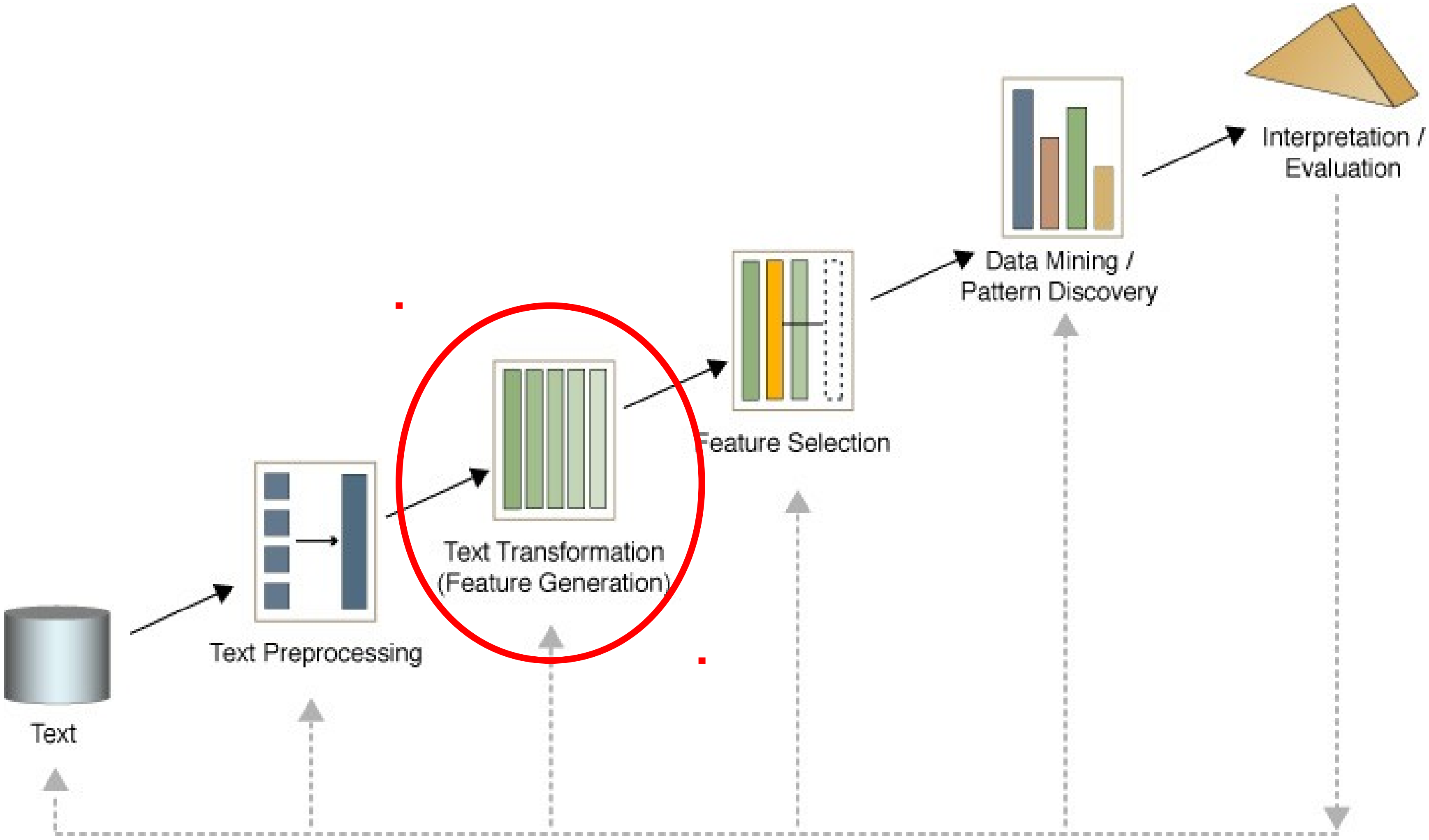
Stemming using the Natural Language Toolkit (NLTK) library:

```
from nltk.stem.porter import PorterStemmer

# Stem tokens
stemmer = PorterStemmer()
tokens = ['Jupiter', 'is', 'the', 'largest', 'gas', 'planet']
stems = []
for item in tokens:
    stems.append(stemmer.stem(item))
```

https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html
https://www.nltk.org/book/ch03.html

Feature Generation



Term-Document Matrix

Term	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	Σ
									Dokument												
oil	5	12	2	1	1	7	3	3	5	9	5	4	5	4	3	4	5	3	3	1	85
price	5	6	2	2	0	8	1	2	2	10	5	1	5	2	0	3	3	3	3	0	63
opec	0	15	0	0	0	8	1	2	2	6	5	2	2	4	0	0	0	0	0	0	47
mln	0	4	0	0	2	4	1	0	0	3	9	0	0	0	0	3	3	0	0	2	31
market	2	5	0	0	0	3	0	2	0	10	1	2	2	0	0	0	0	0	3	0	30
barrel	2	0	1	1	0	4	0	0	1	3	3	0	1	1	0	3	3	1	0	2	26
bpd	0	4	0	0	0	7	0	0	0	2	8	0	0	2	0	0	0	0	0	0	23
dlrs	2	0	1	2	2	2	1	0	0	4	2	0	0	0	0	1	1	5	0	0	23
crude	2	0	2	3	0	2	0	0	0	0	5	2	0	2	0	0	0	2	0	1	21
saudi	0	0	0	0	0	0	0	1	0	5	7	1	4	0	0	0	0	0	0	0	18
kuwait	0	0	0	0	0	10	0	1	0	3	0	1	0	2	0	0	0	0	0	0	17
offici	0	0	0	0	0	5	1	1	0	1	4	3	1	0	0	0	0	0	1	0	17
meet	0	6	0	0	0	3	0	1	0	1	0	1	0	2	0	0	0	0	0	0	14
pct	0	0	0	0	2	0	2	2	2	1	0	0	1	0	0	1	1	0	0	2	14
product	1	6	0	0	0	1	0	0	0	0	4	0	0	0	0	0	0	0	0	1	13
accord	0	0	0	0	0	0	0	0	0	5	1	0	2	0	0	0	0	0	4	0	12
futur	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	9	0	12
minist	0	0	0	0	0	3	0	0	1	3	1	2	1	1	0	0	0	0	0	0	12
govern	0	0	0	0	0	0	5	0	6	0	0	0	0	0	0	0	0	0	0	0	11
month	0	1	0	0	0	2	2	0	1	0	5	0	0	0	0	0	0	0	0	0	11
report	0	1	0	0	0	1	8	0	0	0	0	1	0	0	0	0	0	0	0	0	11
sheikh	0	0	0	0	0	3	0	0	5	2	0	0	0	1	0	0	0	0	0	0	11
industri	0	2	0	0	0	1	1	1	1	0	0	0	0	0	0	1	2	0	1	0	10
produc	0	0	0	0	0	4	1	1	0	3	0	0	0	0	0	0	0	0	0	1	10
quota	0	2	0	0	0	4	0	0	1	1	1	0	0	1	0	0	0	0	0	0	10
reserv	0	0	0	0	3	0	0	0	1	0	0	0	0	0	0	3	3	0	0	0	10
world	0	1	0	0	0	1	3	0	1	1	0	0	1	1	0	0	0	0	1	0	10
⋮																					
Σ	48	204	34	39	46	219	219	73	161	180	208	57	61	54	56	68	89	44	147	32	2039

Feature Generation

- Document is treated as a bag of words (or terms)
  - each word or term becomes a feature.
  - order of words/terms is ignored.
- Each document is represented by a vector.
- Different techniques for vector creation:
  1. Binary Term Occurrence: Boolean attributes describe whether or not a term appears in the document.
  2. Term Occurrence: Number of occurences of a term in the document (problematic if documents have different length).
  3. Terms frequency: Attributes represent the frequency in which a term appears in the document (Number of occurrences / Number of words in document)
  4. TF-IDF: see next slide

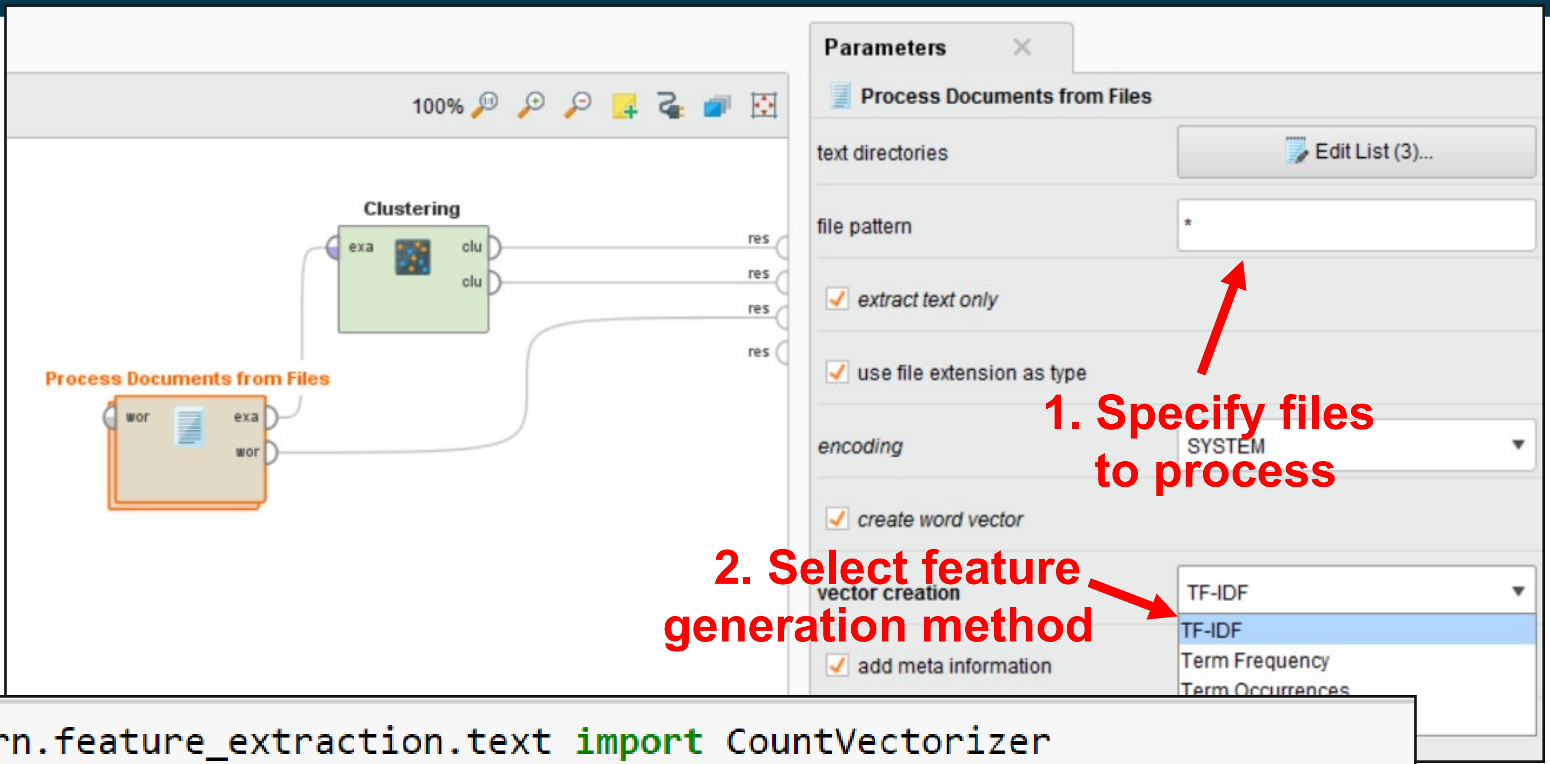
The TF-IDF Term Weighting Scheme

- The TF-IDF weight (term frequency–inverse document frequency) is used to evaluate how important a word is to a corpus of documents.
  - TF: Term Frequency (see last slide)
    - Tf_{ij}: term frequency of term i in document j
  - IDF: Inverse Document Frequency
    - N: total number of docs in corpus
    - df_i: the number of docs in which term i appears
  - Gives more weight to rare words
  - Give less weight to common words (domain-specific “stopwords”)

$$idf_i = \log \frac{N}{df_i}$$

$$tfidf_{ij} = tf_{ij} \times idf_i$$

Feature Generation in RapidMiner and Python

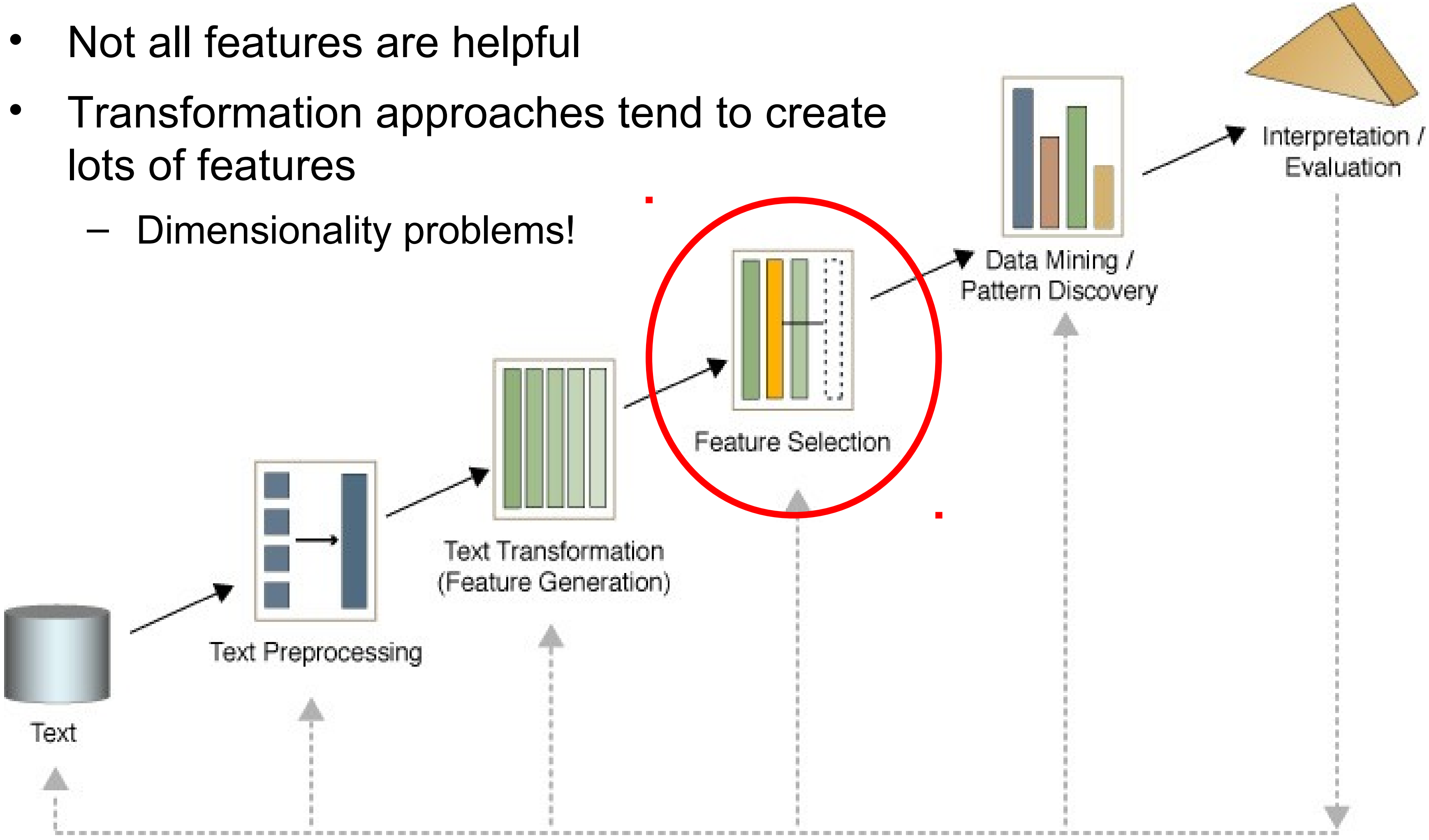



```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

binary_term_occurances_vectorizer = CountVectorizer(binary='True')
term_occurances_vectorizer = CountVectorizer(binary='False')
term_frequency_vectorizer = TfidfVectorizer(use_idf='False')
tf_idf_vectorizer = TfidfVectorizer(use_idf='True')
```

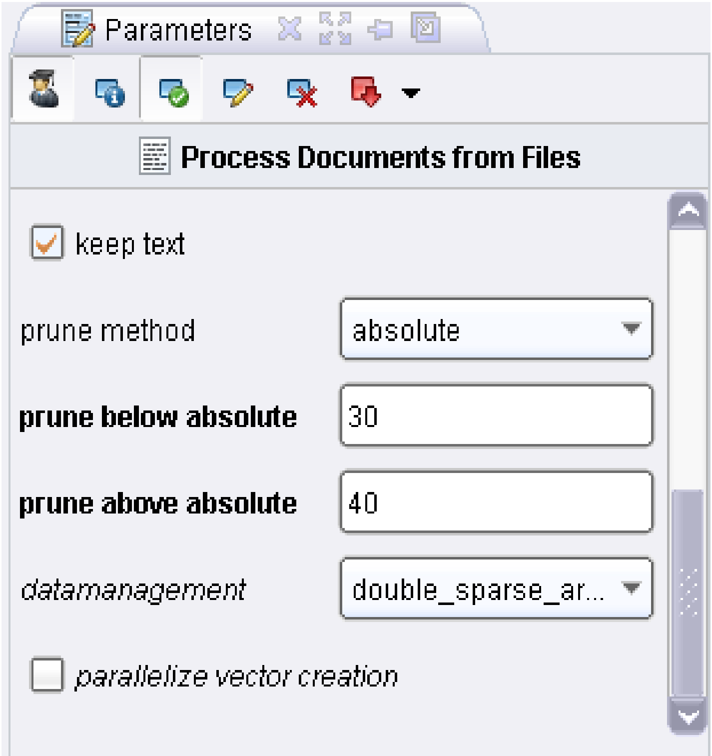
Feature Selection

- Not all features are helpful
- Transformation approaches tend to create lots of features
  - Dimensionality problems!



Pruning Vectors in RapidMiner & Python

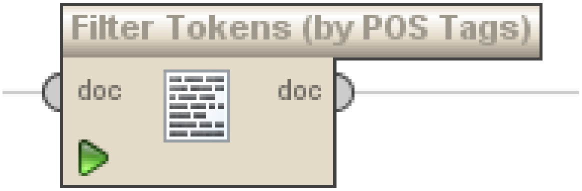
- Pruning methods
  - Specify if and how too frequent or too infrequent words should be ignored
- Different options:
  - *Percentual*: ignore words that appear in less / more than this percentage of all documents
  - *Absolute*: ignore words that appear in less / more than that many documents
  - *By Rank*: Specifies how many percent of the most infrequent / infrequent words are ignored




```
vectorizer = TfidfVectorizer(min_df=0.1, max_df=0.3) # Percentual
vectorizer = TfidfVectorizer(min_df=5, max_df=20) # Absolute
```

POS Tagging Revisited

- POS tags may help with feature selection
  - sometimes, certain classes of words may be discarded
  - e.g., modal verbs
  - e.g., adjectives
    - texts about red and blue cars are similar
    - texts about red and blue trousers are similar
  - but
    - texts about red cars and red trousers are not similar




```
In [1]: import nltk

        s = "The red car is standing in the garage"
        tokens = nltk.word_tokenize(s)
        tags = nltk.pos_tag(tokens)
        print(tags)

[('The', 'DT'), ('red', 'JJ'), ('car', 'NN'), ('is', 'VBZ'), ('standing', 'VBG'), ('in', 'IN'), ('the', 'DT'), ('garage', '
NN')]

In [2]: filtered_tags = [t for t in tags if (t[1] == "NN" or t[1] == "VBG")]
        print(filtered_tags)

[('car', 'NN'), ('standing', 'VBG'), ('garage', 'NN')]
```

Named Entity Recognition and Linking

- Named Entity Recognition (NER):
  - identifying persons, places, organizations, …

- Example:
  - “Stock quote of Apple Inc. expected to exceed $600.”

    → “Stock quote of <ORGANIZATION>Apple Inc.</ORGANIZATION> expected to exceed <AMOUNT>$600</AMOUNT>.”

- The classes of NER may be useful features
  - the exact amount of money does not matter
  - useful to know that any amount is mentioned

Named Entity Recognition and Linking

- Named Entity Linking
  - Identify named entities in a knowledge base
  - e.g., Link to Wikipedia

- May be used to create additional features
  - e.g., Wikipedia categories

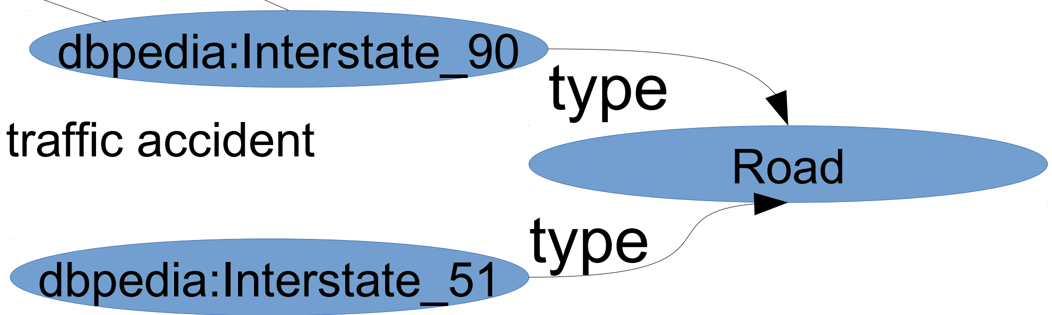
    “Stock quote of <ORGANIZATION link=”http://en.wikipedia.org/wiki/Apple_Inc.”>Apple Inc.</ORGANIZATION> expected to exceed <AMOUNT>$600</AMOUNT>.”
  - Categories: *Mobile phone manufacturers*, *Technology companies of the United States*, ...

Named Entity Recognition and Linking

- Example: RapidMiner Linked Open Data Extension
  - Can use DBpedia
    (a structured subset of Wikipedia)
  - Named Entity Linking with DBpedia Spotlight
  - Feature extraction: e.g., all types of the identified entities

Named Entity Recognition and Linking

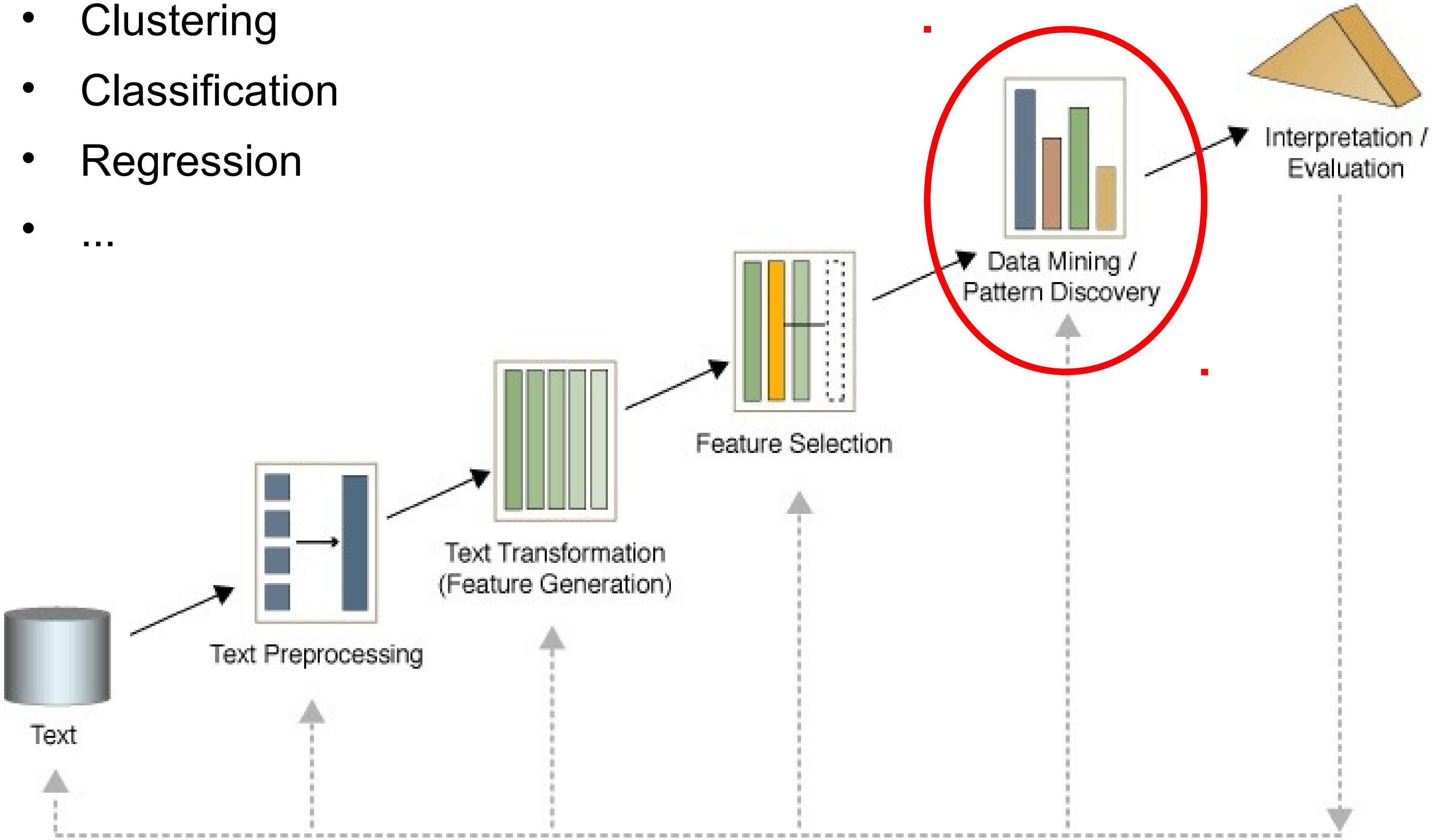
- Example set of texts:
  - “Again crash on I90”
  - “Accident on I90”
- Model:
  - type=Road → indicates traffic accident



- Applying the model:
  - “Two cars crashed on I51” → indicates traffic accident
- Note:
  - The feature “I90” alone is not as useful!

Pattern Discovery

- Clustering
- Classification
- Regression
- ...

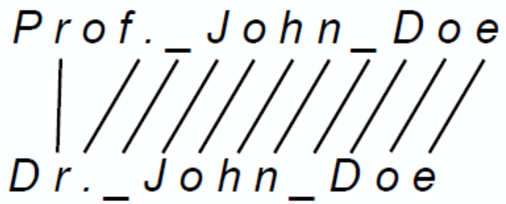


Text Mining: Clustering Definition

- Given a set of documents and a similarity measure among documents
- find clusters such that:
  - Documents in one cluster are more similar to one another
  - Documents in separate clusters are less similar to one another
- Question: Which similarity measures are a good choice for comparing document vectors?

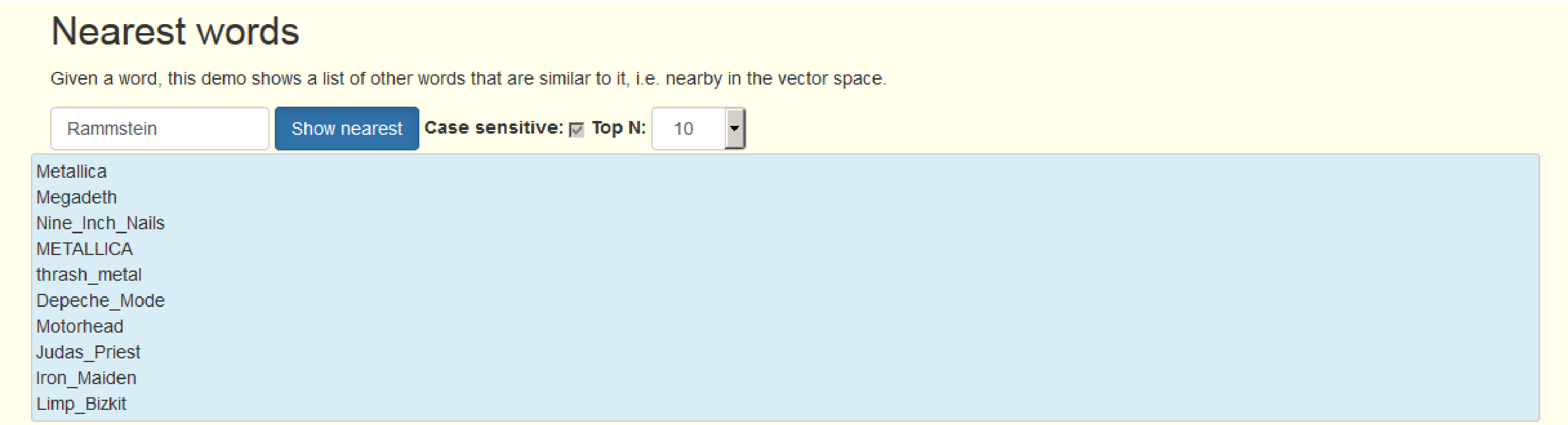
Jaro Distance

- Measures the dissimilarity of two strings
- Developed for name comparison in the U.S. Census
- Optimized for comparing person names
- Based on the number of common characters within a specific distance
- Example:



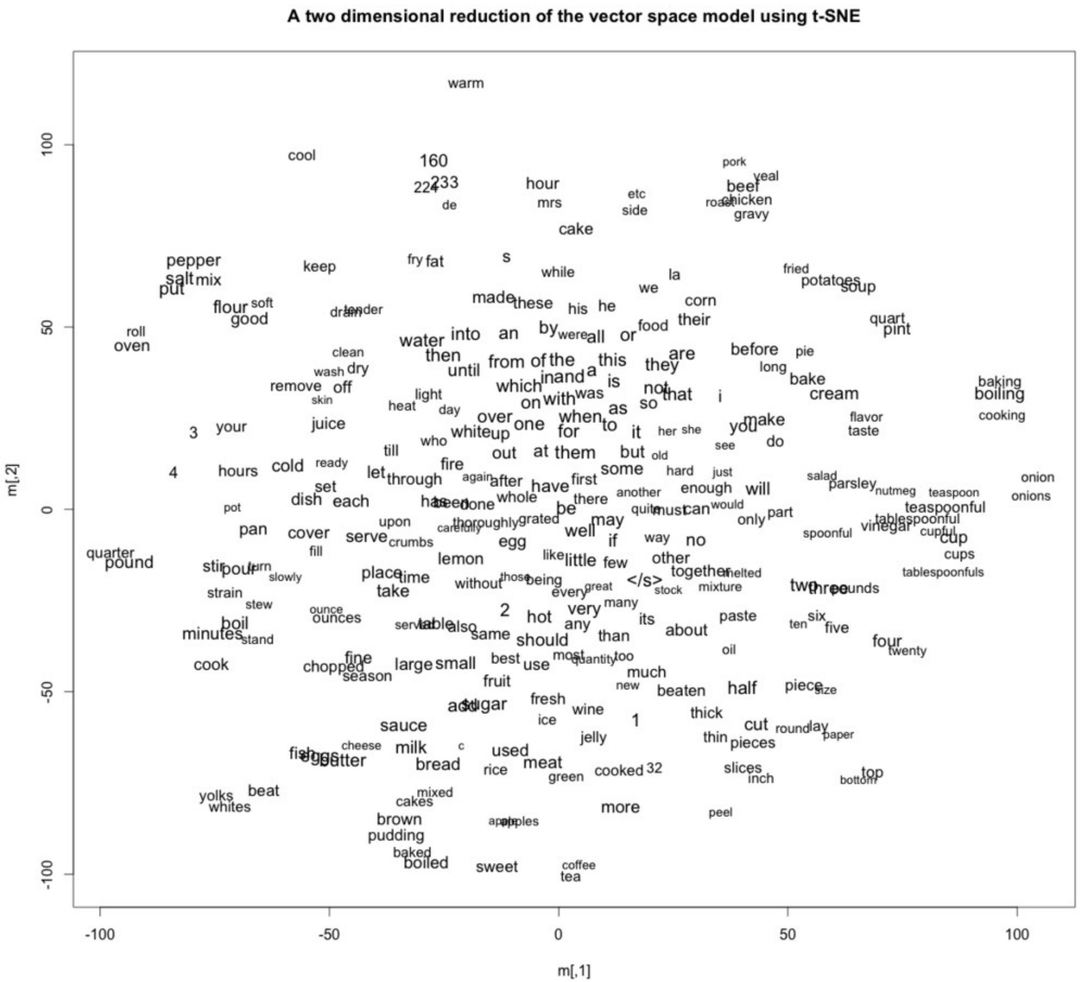
word2vec Distance

- word2vec (and other *embedding* techniques) represent a word by an n-dimensional feature vector
  - details: see Data Mining II
- Distance can then be understood as metric distance in that vector space



http://bionlp-www.utu.fi/wv_demo/

word2vec distance



http://yamano357.hatenadiary.com/entry/2015/11/04/000332

n-gram Based Similarity

- Measures the similarity of two strings
- split string into set of trigrams:
  - e.g., “similarity” becomes “sim”, “imi”,”mil”,”ila”, “lar”, ..,
- measure overlap of trigrams
  - e.g., Jaccard: |common trigrams| / |all trigrams|

- Example: clustering similar product offers on eBay
- “iPhone5 Apple” vs. “Apple iPhone 5”
  - common trigrams: “iPh”, “Pho”, “hon”, “one”, “App”, “ppl”, “ple”
  - other trigrams: “ne5”, “e5 “, “5 A”, “ Ap” (1), “le “, “e i”, “ iP”, “e 5” (2)
  - Jaccard: 7/15 = 0.47

Jaccard Coefficient

- Asymmetric binary attributes: If one of the states is more important or more valuable than the other.
  - By convention, state 1 represents the more important state
  - 1 is typically the rare or infrequent state
  - Example: Binary Term Occurences
- Jaccard coefficient is a popular measure

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

Number of 11 matches / number of not-both-zero attributes values

Jaccard Coefficient

- Sample document set:
  - d1 = “Saturn is the gas planet with rings.”
  - d2 = “Jupiter is the largest gas planet.”
  - d3 = “Saturn is the Roman god of sowing.”

- Documents as vectors:
  - Vector structure:
    (Saturn, is, the, gas, planet, with, rings, Jupiter, largest, Roman, god, of, sowing)

    d1: 1111111000000

    d2: 0111100110000

    d3: 1110000001111

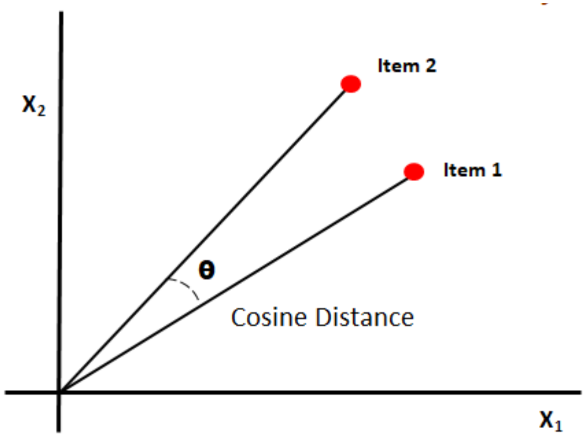
- **sim(d1,d2) = 0.44**
- sim(d1,d3) = 0.27
- sim(d2,d3) = 0.18

Cosine Similarity

- Often used for computing the similarity of documents
- If d_1 and d_2 are two document vectors, then

$$\cos(d_1, d_2) = \frac{d_1 \circ d_2}{\|d_1\| \times \|d_2\|}$$

- Intuitive interpretation: *angle* of two documents
  - Advantage: length of document does not matter



Cosine Similarity and TF-IDF

- A commonly used combination for text clustering
- Each document is represented by vectors of TF-IDF weights

- Sample document set:
  - “Saturn is the gas planet with rings.”
  - “Jupiter is the largest gas planet.”
  - “Saturn is the Roman god of sowing.”

- First document as TF-IDF vector:
  - (1/7 * log(3/2), 1/7*log(3/3), 1/7*log(3/1), …, 0, 0, 0, ...)

Saturn is the Jupiter largest Roman

Cosine Similarity and TF-IDF

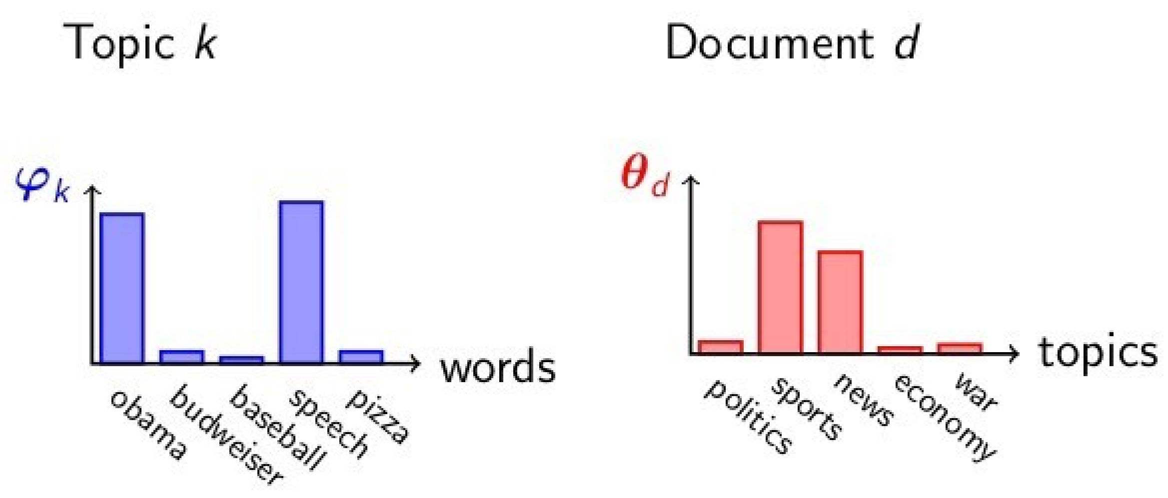
- Sample document set:
  - d1 = “Saturn is the gas planet with rings.”
  - d2 = “Jupiter is the largest gas planet.”
  - d3 = “Saturn is the Roman god of sowing.”

- Documents as vectors:
  - Vector structure:
    (Saturn, is, the, gas, planet, with, rings, Jupiter, largest, Roman, god, of, sowing)
  - d1 = (0.03, 0, 0, 0.03, 0.03, 0.07, 0.07, 0, 0, 0, 0, 0, 0)
  - d2 = (0, 0, 0, 0.03, 0.03, 0, 0, 0.08, 0.08, 0, 0, 0, 0)
  - d3 = (0.03, 0, 0, 0, 0, 0, 0, 0, 0, 0.07, 0.07, 0.07, 0.07)

- **sim(d1,d2) = 0.13**
- sim(d1,d3) = 0.05
- sim(d2,d3) = 0.0

Alternative Document Representations

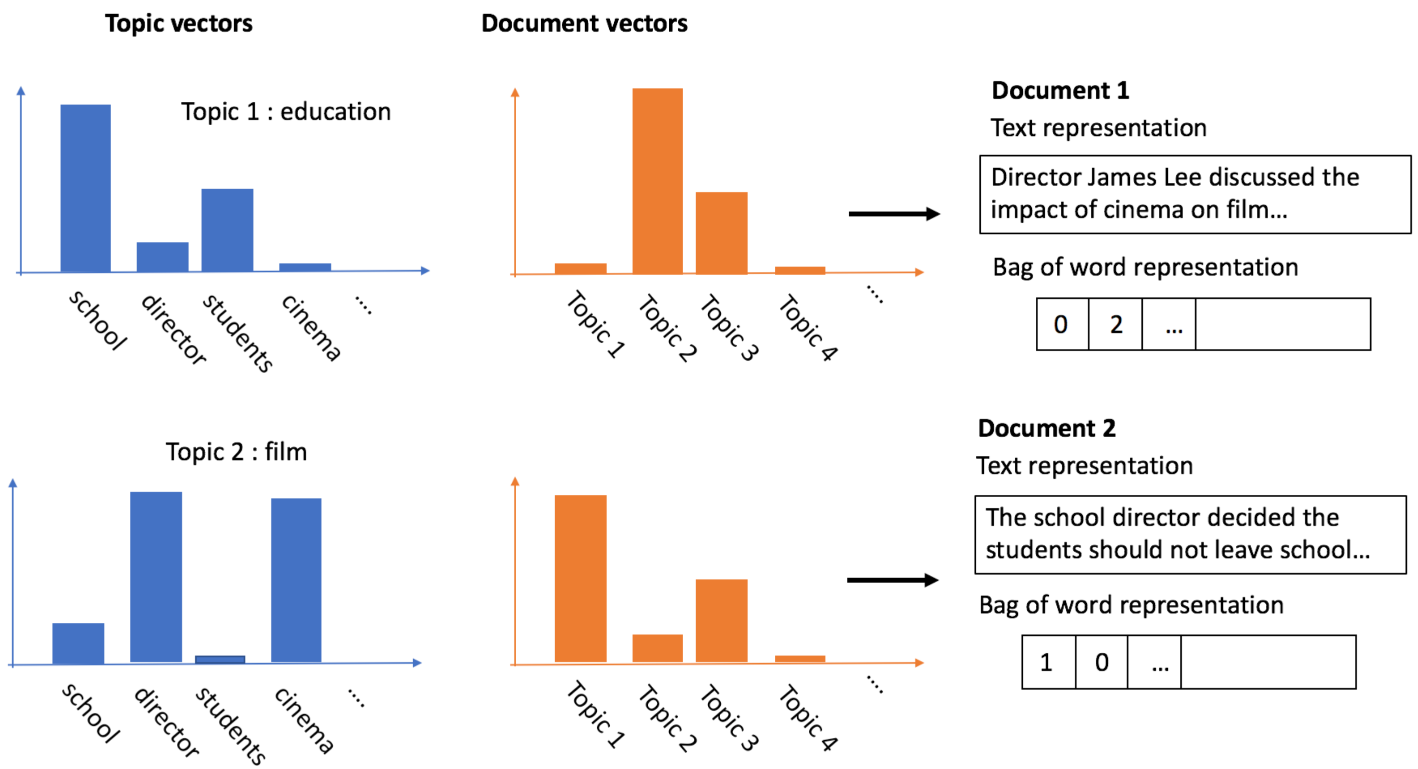
- Topic Modeling (e.g., Latent Dirichlet Allocation)
  - Each document consists of words
  - Words have a certain probability to be used in topics
  - Each document belongs to one or more topics to a certain degree



https://towardsdatascience.com/latent-dirichlet-allocation-15800c852699

Alternative Document Representations

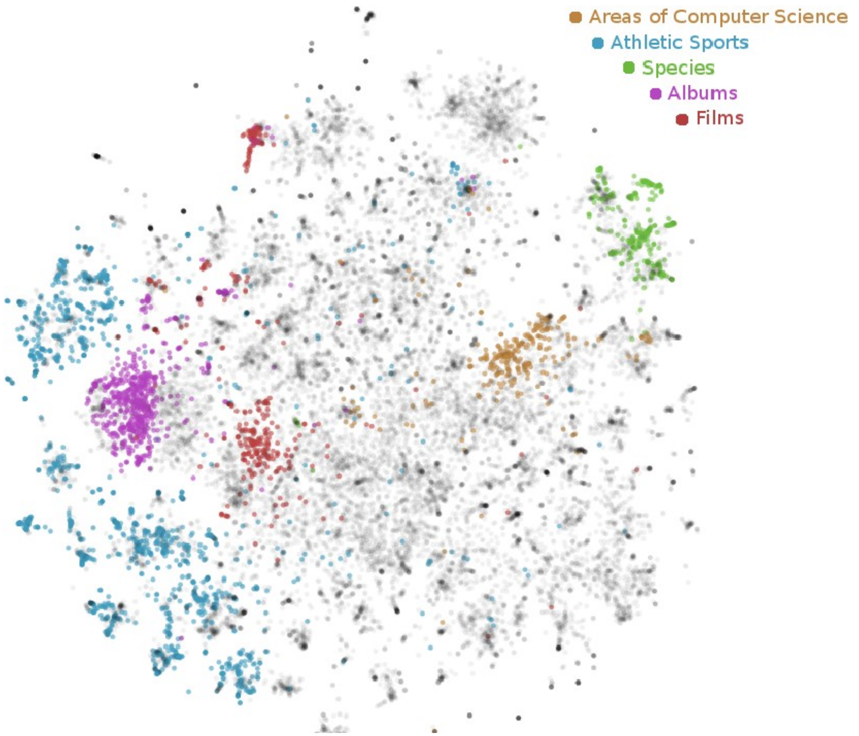
- Topic Modeling (e.g., Latent Dirichlet Allocation)
  - A document is represented by a numerical vector of *n* topics



https://www.datacamp.com/community/tutorials/lda2vec-topic-model

Alternative Document Representations

- doc2vec
  - an extension of word2vec
  - each document is projected into a vector space



Dai et al. (2015): Document Embedding with Paragraph Vectors

Text Mining: Classification Definition

- Given: A collection of labeled documents (training set)
- Find: A model for the class as a function of the values of the features.
- Goal: Previously unseen documents should be assigned a class as accurately as possible.
- Classification methods commonly used for text
  - Naive Bayes, SVMs
  - Neural Networks
  - Random Forests (see Data Mining 2)

Text Mining: Sentiment Analysis

- A specific classification task
- Given: a text
- Target: a class of sentiments
  - e.g., positive, neutral, negative
  - e.g., sad, happy, angry, surprised
- Alternative: numerical score (e.g., -5...+5)

- Can be implemented as supervised classification/regression task
  - requires training data
  - i.e., pairs like <text;sentiment>

Text Mining: Sentiment Analysis

- Labeling data for sentiment analysis
  - is expensive
  - like every data labeling task

- Example public data sets for labeling: reviews

173 of 213 people found the following review helpful

Listen Closer

Trent Reznor should just release an album with a new title, new artwork, and new song titles. But instead of actual new material, it should all just be the songs from The Downward Spiral.

It can be called There You Go, ****heads.

After all, it's what everyone wants.

I remember the day I bought The Downward Spiral. My first thought after...

Read the full review ›

Published 1 month ago by Philip Atherton

Vs.

19 of 21 people found the following review helpful

Good, But Not Their Best

Its funny how immediately after an established band that's been around for a while comes out with a new album all the fan-boys give reviews saying it's the greatest thing ever. I am a Nine Inch Nails fan too and have all their albums, so I'd thought I'd give my review which I hope is a little more fair.

It's an electronic based album with some guitar, bass,...

Read the full review ›

Published 1 month ago by JKat

- e.g., uclassify: trained on 40,000 Amazon reviews, ~80% accuracy

Preprocessing for Sentiment Analysis

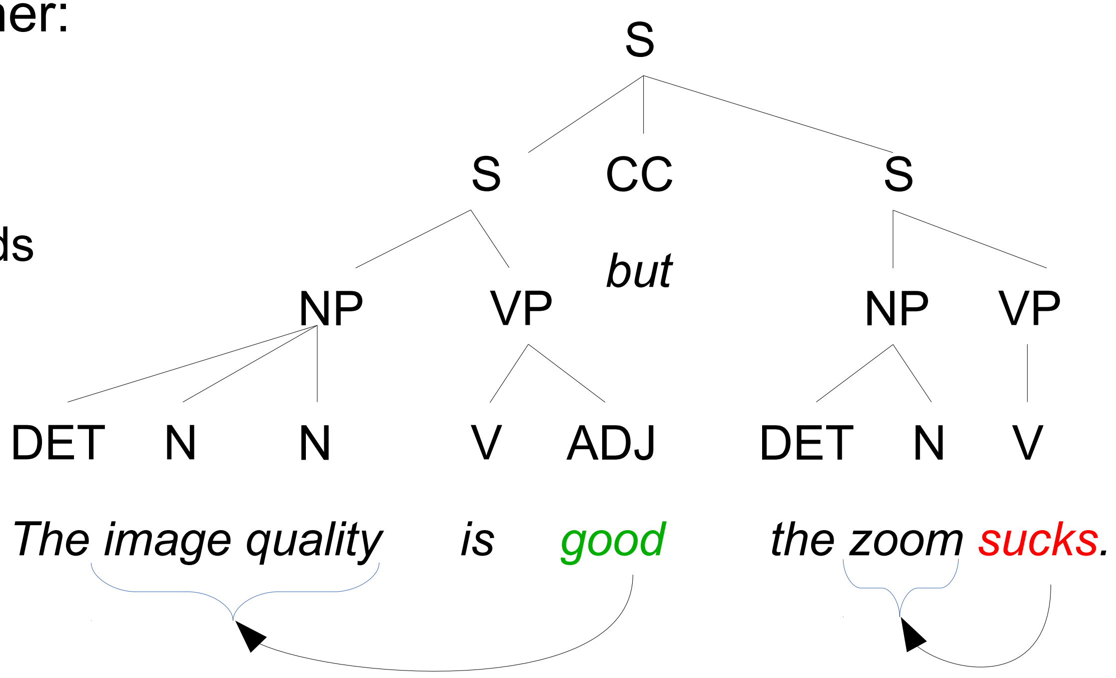
- Recap – we started our processing with:
  Simple Syntactic Analysis
  - Text Cleanup (remove punctuation, HTML tags, …)
  - Normalize case
  - …

- Suitable for some text processing tasks
- However, reasonable features for sentiment analysis might include
  - punctuation: use of “!”, “?”, “?!”
  - smileys (usually encoded using punctuation: ;-))
  - use of visual markup, where available (red color, bold face, ...)
  - amount of capitalization (“screaming”)

Sentiment Analysis for Aspects

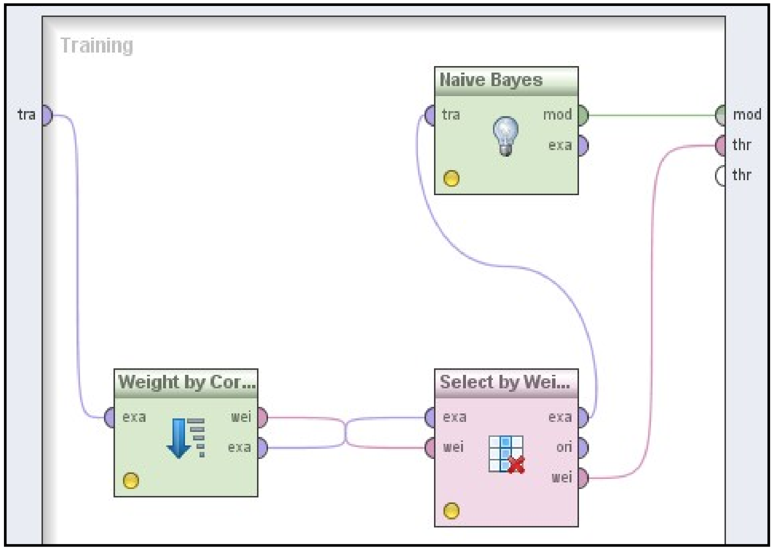
- Example product review:
  - *“The image quality is good, but the zoom sucks.”*

- Putting the pieces together:
  - POS tagging
  - Keyphrase extraction
  - Marking sentiment words



Some Text Classification Tricks

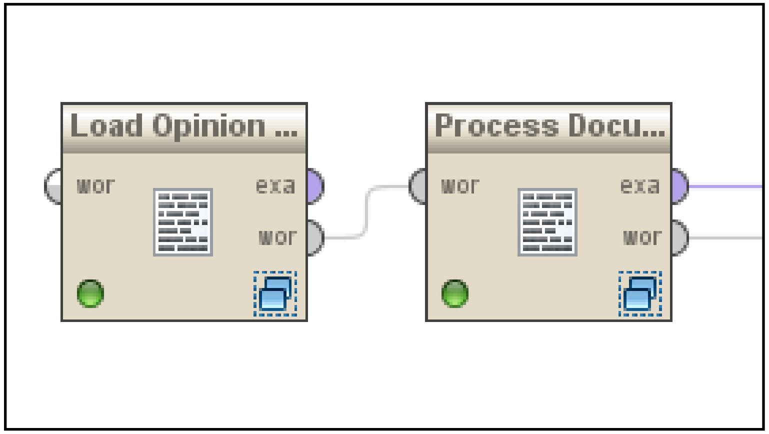
- Finding selective words
  - weight words according to their correlation with label
  - select Top-K words with highest correlation/information gain...
  - Python: SelectKBest



- Removing low variance features
  - RapidMiner: Remove Useless Attributes
  - Python: VarianceThreshold

Some Text Classification Tricks

- Sentiment Analysis
  - use external dictionary of opinion words
  - Bing Liu‘s List http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar
  - restrict word list to these words

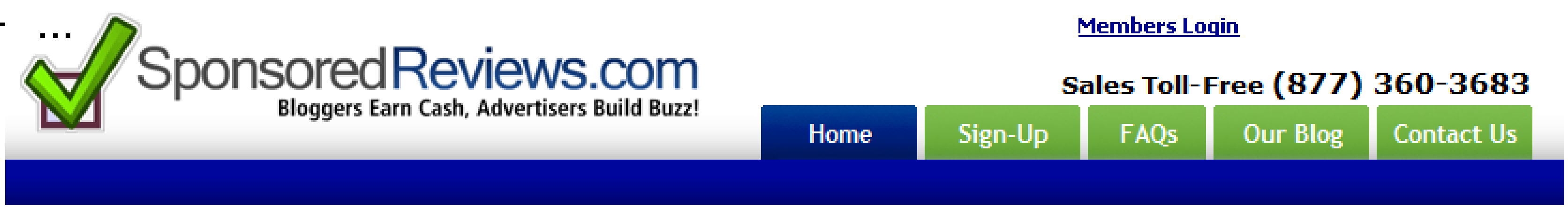


- AFINN: A list of ~2.5k sentiment conveying words with scores
  - Python package afinn
  - afinn.score(“Interesting lecture”) → 2.0
  - afinn.score(“Boring lecture”) → –0.3

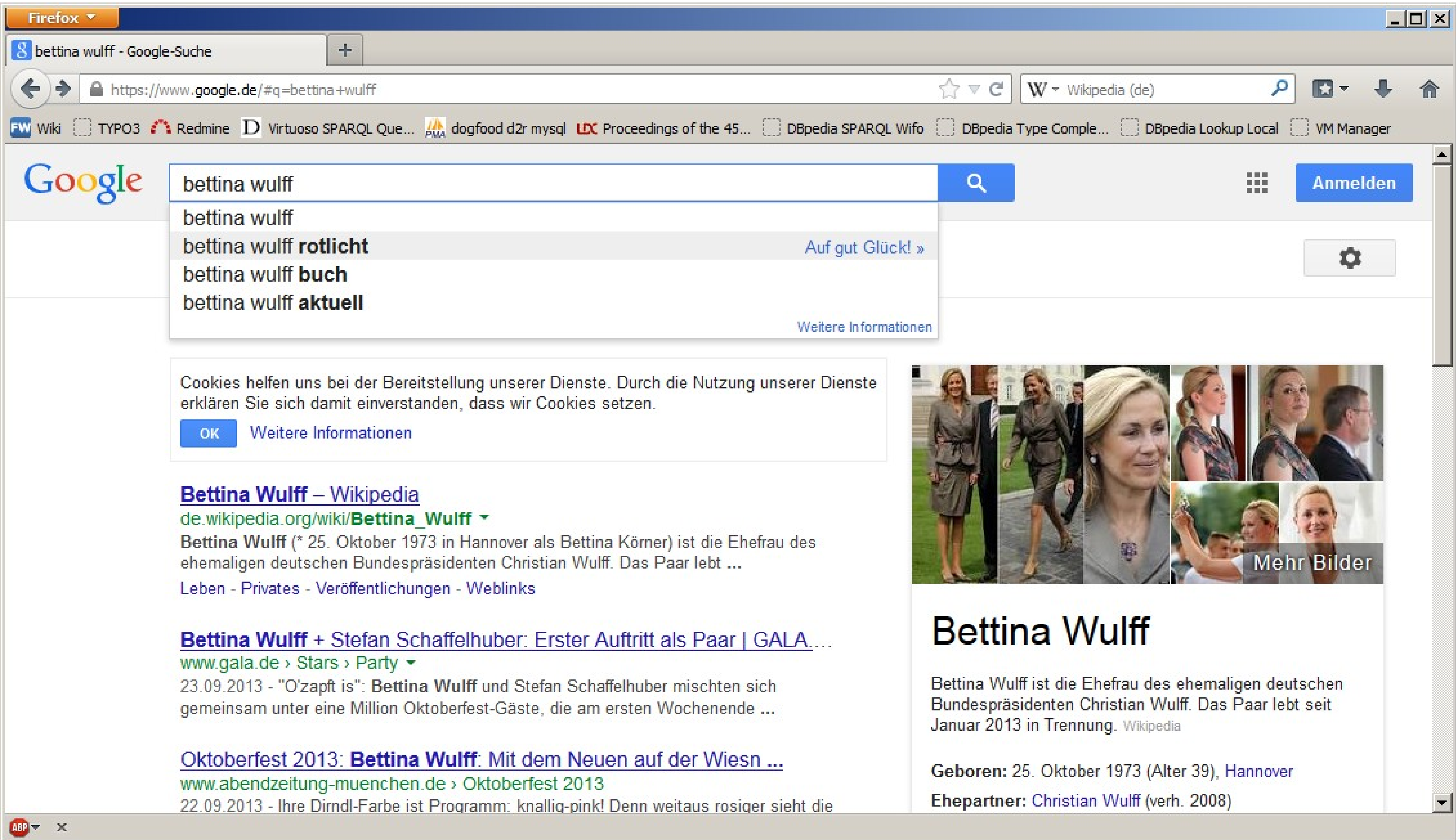
Text Classification: Identifying Fake Reviews

- Useful features (besides text):
  - length of review
  - use of positive sentiment words (e.g., SentiWordNet)
  - …
- However, text classification alone only yields a low accuracy

  Other ways to go:
  - include other reviews of the same reviewer, find typical patterns
  - review frequency
  - typical rating behavior
  - similarity of product description and review
  - ...



Query Completion Revisited

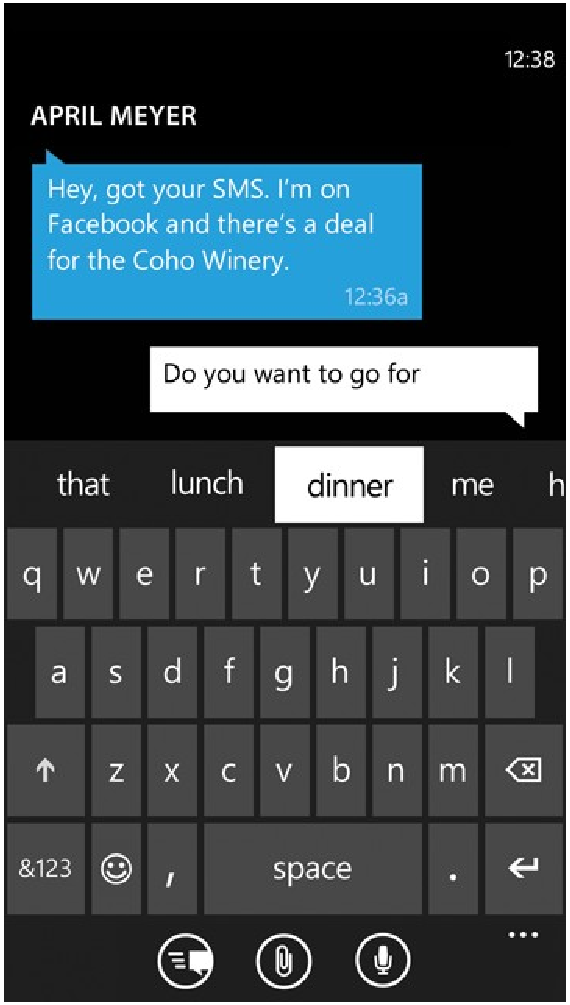


Query Completion Revisited

- How to refine a query?
  - Terms that frequently co-occur with the terms entered (corpus: documents)
  - Terms that are frequently searched together with the terms entered (corpus: query logs)
- Given some terms entered: t1, t2
  - look for t3 so that t1, t2, t3 is a *frequent pattern*
- Approach: use a corpus of texts
  - represent them as binary vectors
  - look for frequent patterns (see next lecture)

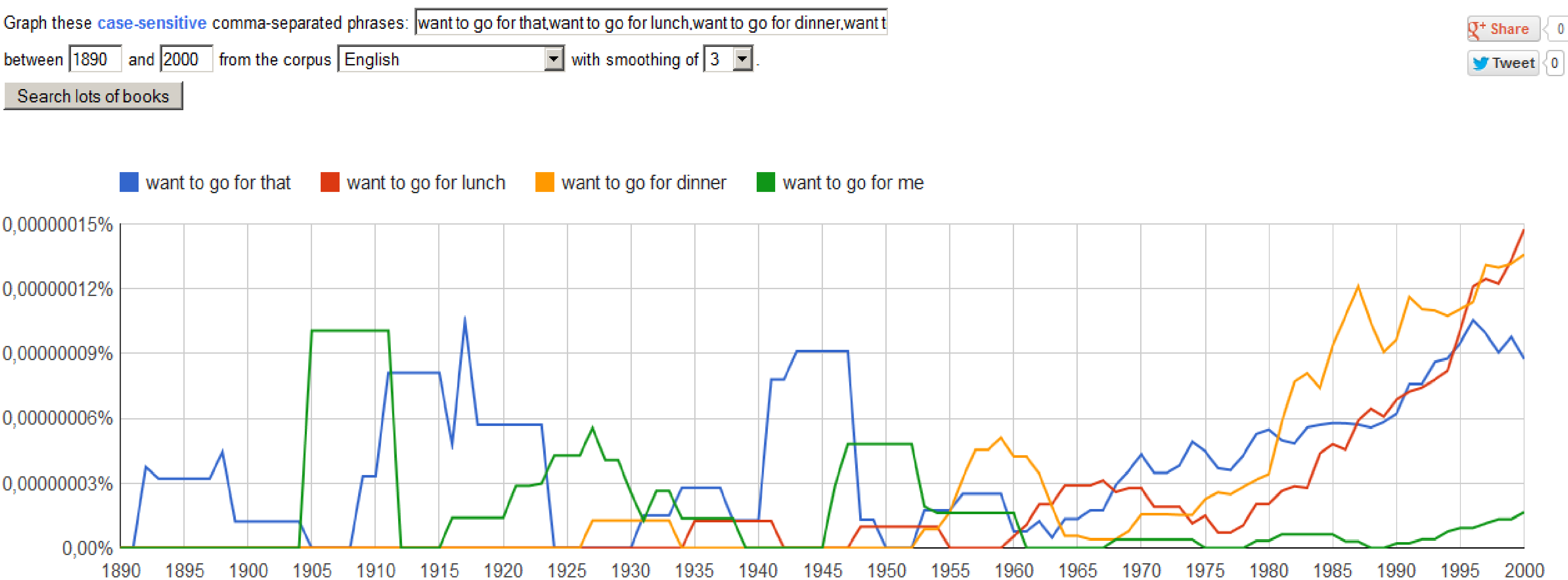
Auto-complete Revisited

- Method: sequential pattern mining
  - find frequent *sequences* that start with a given root
  - see lecture Data Mining II



Auto-complete Revisited

- Google hosts a corpus of frequent patterns
- mined from Google books
- see http://books.google.com/ngrams/



Processing Text from Social Media

- An interesting source of knowledge
  - e.g., market research
  - e.g., opinion mining
- However, challenging to process with standard methods
- Example (a real tweet):
  - “ikr smh he asked fir yo last name so he can add u on fb lololol”

Processing Text from Social Media

- Respect special characters
  - e.g., hashtags and user mentions
  - may be treated separately

- Normalizing
  - unfolding abbreviations (“2moro” → “tomorrow”)
  - replacing slang words with standard English
  - spelling corrections

Processing Text from Social Media

- POS Tagging
  - the POS tagger by Charniak was trained on news texts
  - will work very poorly on social media data
  - there are specialized POS taggers trained, e.g., on Twitter data

- Named Entity Recognition
  - often relies on capitalized words
    - *“The document was signed by the US congress.”*
    - *The document was signed by us.”*
  - there are particular NER tools for social media

Summary

- Main task: Preprocessing of text in order be able to apply well known Data Mining algorithms
- There are lots of alternative preprocessing techniques
    - Mind the task!
- Text Mining is tricky, but “ok”-ish results are easily achieved
- If you want to hear more
    - visit lectures on *Text Analytics* and *Web Search and Information Retrieval* (Ponzetto, Glavaš & colleagues)

Questions?