

Data Mining

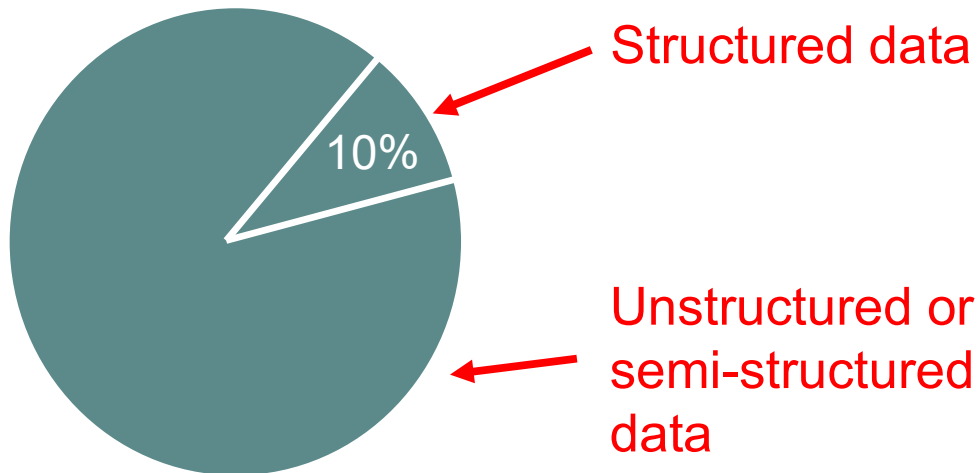
Text Mining

1. What is Text Mining?
2. Text Preprocessing
3. Feature Creation
4. Feature Selection
5. Solving Downstream Tasks

Motivation for Text Mining

Approximately 90% of the world's data is held in unstructured formats.

Source: Oracle Corporation

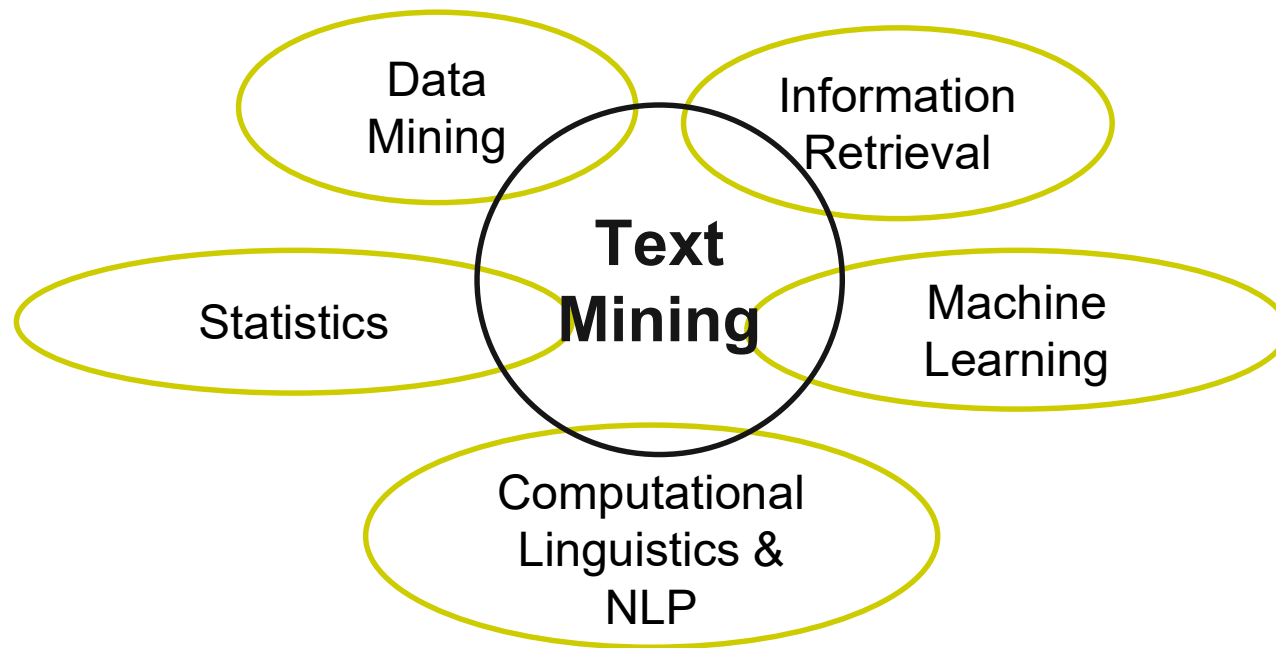


Examples:

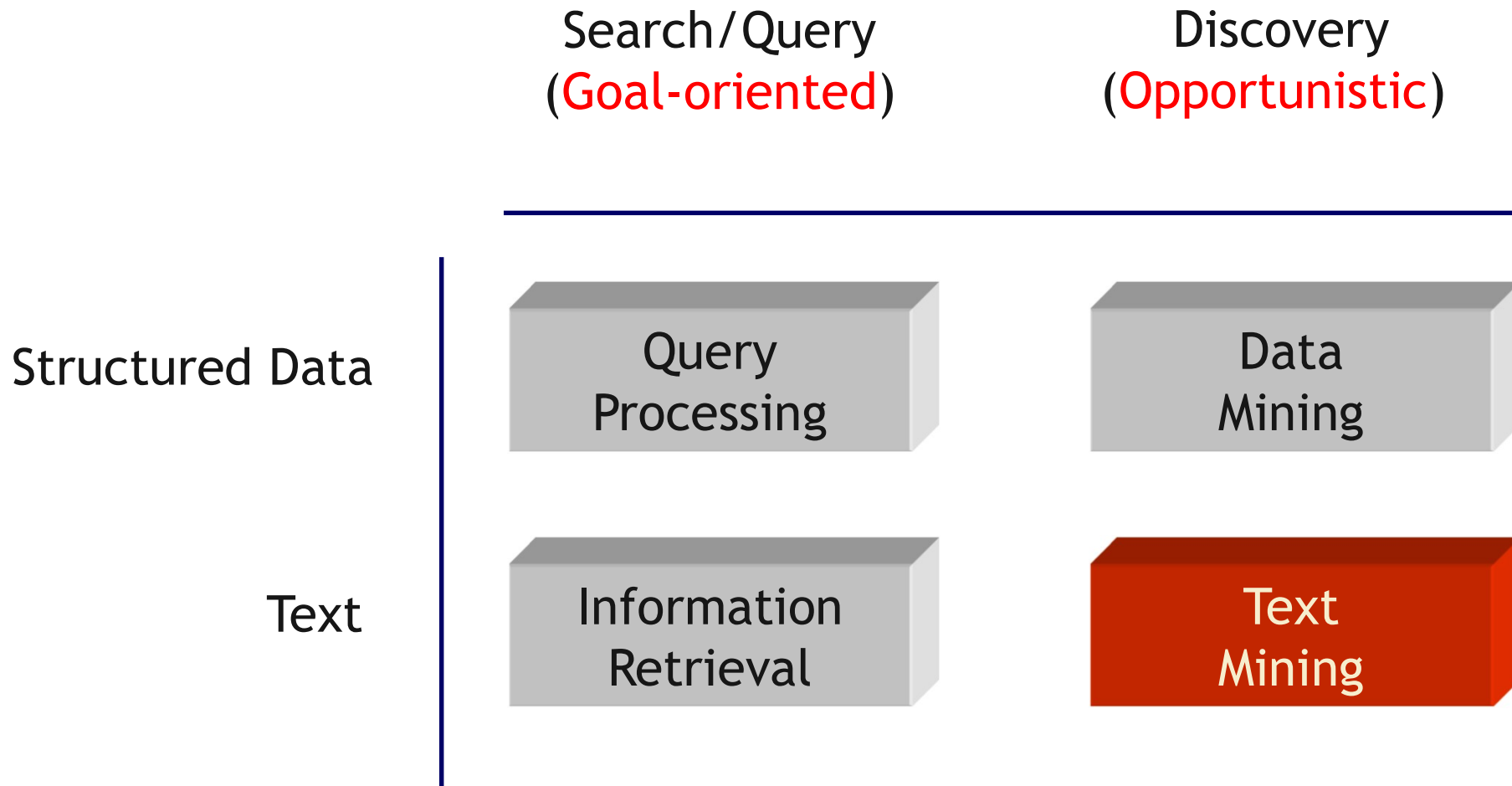
- web pages
- emails
- customer complaint letters
- corporate documents
- scientific papers
- books in digital libraries

Text Mining

The extraction of implicit, **previously unknown** and **potentially useful** information from large amounts of **textual resources**.



Search versus Discovery



Some Text Mining Applications

1. Classification of news stories
2. Email and news filtering / SPAM detection
3. Sentiment analysis
4. Grouping of documents or web pages
5. Information extraction
6. Text summarization / Text generation
7. Question answering

Mixture of Document Clustering and Classification

+Chris Search Images Maps Play YouTube Gmail Drive Calendar Translate More

Google

Chris Bizer 1 + Share

News U.S. edition Classic

Top Stories

- Jack Lew
- Chrome for Android
- James Holmes
- iPhone5
- Miss America pageant
- Nokia Lumia
- North Korea
- Les Miserables
- X Factor
- Harry Styles

Baden-Württemberg, G...
World
U.S.
Business
Technology
Entertainment
Sports
Science
Health
Spotlight

Top Stories

WH gun plan: Out-organize the NRA
Politico - 27 minutes ago
President Barack Obama is trying an end run around the NRA - rallying groups as varied as churches, medical organizations, retailers and the Rotary Club to build support for new gun regulations.
Opinion: The NRA's game plan Chicago Tribune - by charles madigan
In Depth: Obama gun plan may feature background checks on all buyers Los Angeles Times
See realtime coverage »

Boeing's 787 Dreamliner suffers more mishaps
USA TODAY - 9 minutes ago
LONDON - Another two incidents struck Boeing's 787 Dreamliner plane on Friday when an All Nippon Airways aircraft suffered a crack to its windscreen during a flight in Japan and an oil leak was found coming from the engine of a separate plane after it ...
In Depth: More Problems for Boeing's 787 Surface in Japan New York Times
Wikipedia: Boeing 787 Dreamliner
See realtime coverage »

California school shooter targeted bullies, sheriff says
Fox News - 12 minutes ago
TAFT, Calif. - The 16-year-old boy had allegedly wounded the teenager he claimed had bullied him, fired two more rounds at students fleeing their first-period science class, then faced teacher Ryan Heber.
Highly Cited: Youth fires shotgun at 2 high school students, hits one; suspect in custody CNN International
In Depth: Sheriff: High school gunman felt he'd been bullied CBS News





Recent

California school shooter targeted bullies, sheriff says
Fox News - 12 minutes ago

International envoy Brahimi holds talks on Syria with top Russian, US officials
Fox News - 9 minutes ago

Jimmy Savile scandal: Report reveals extent of abuse
BBC News - 16 minutes ago

Weather for Mannheim, Germany

Today	Sat	Sun	Mon
			
37° 30°	37° 30°	37° 28°	36° 27°

The Weather Channel - Weather Underground - AccuWeather

Baden-Württemberg, Germany » - Change location

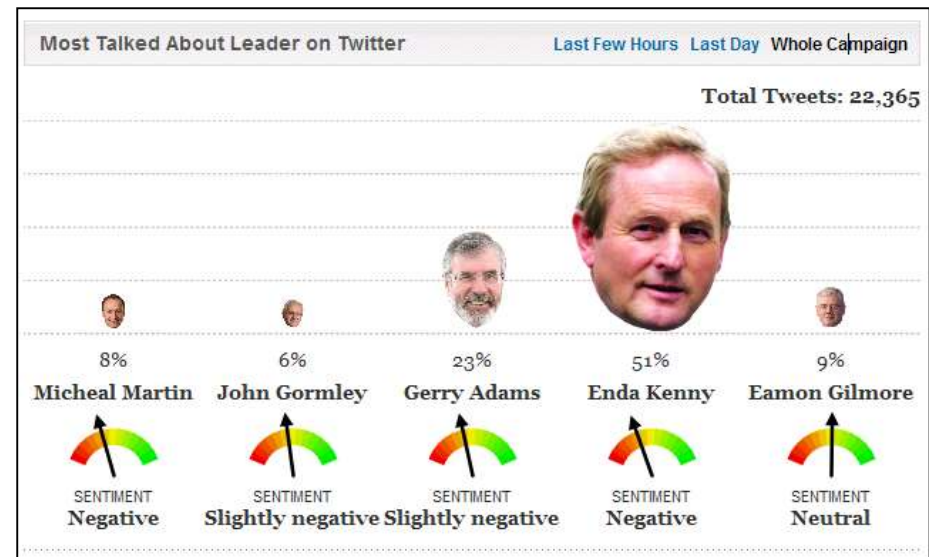
Stuttgart: Slight increase in lease prices
Property Magazine International - Jan 10, 2013

Eishockey: Adler Mannheim bauen Tabellenführung aus
ZEIT ONLINE - Jan 6, 2013

One dead as helicopter crashes on

Sentiment Analysis

- The goal of sentiment analysis is to determine the polarity of a given text at the document, sentence, or feature/aspect level
- Polarity values
 - positive, neutral, negative
 - likert scale (1 to 10)
- Application examples
 - Document level
 - analysis of tweets about politicians
 - Feature/aspect level
 - analysis of product reviews



Information Extraction

- Information extraction is the task of automatically extracting structured information from unstructured or semi-structured documents.

- Subtasks

1. Named Entity Recognition and Disambiguation

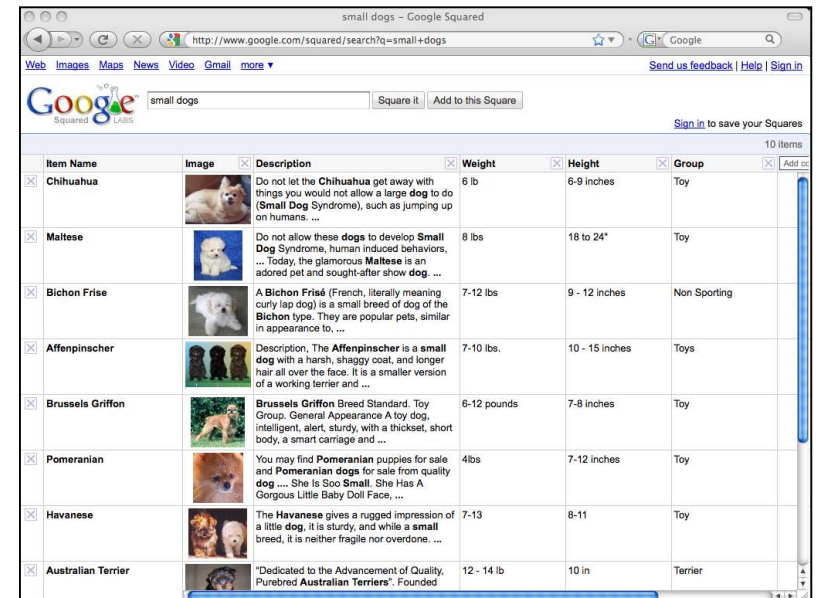
- “The parliament in Berlin has decided ...“
- Which parliament? Which Berlin?









2. Relationship Extraction

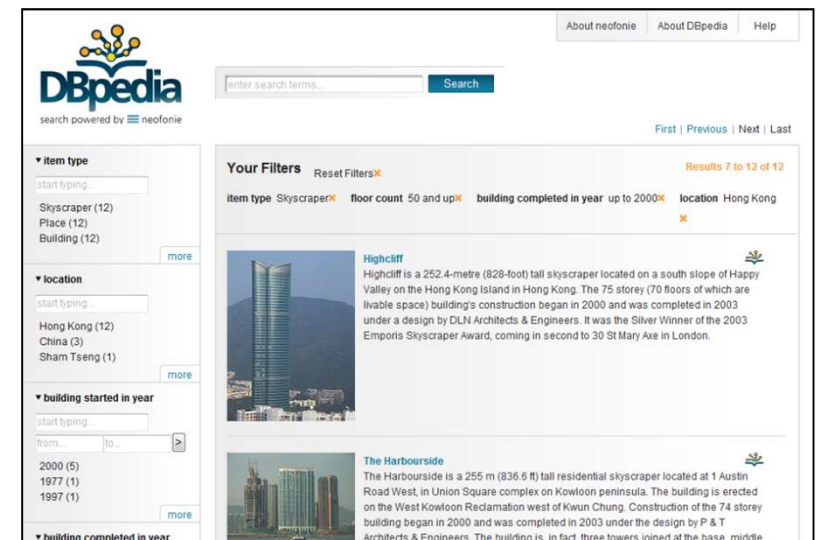
- PERSON works for ORGANIZATION
- PERSON located in LOCATION

3. Fact Extraction

- CITY has population NUMBER
- COMPANY has turnover NUMBER [Unit]



Item Name	Image	Description	Weight	Height	Group
Chihuahua		Do not let the Chihuahua get away with things you would not allow a large dog to do (Small Dog Syndrome), such as jumping up on humans. ...	6 lb	6-9 inches	Toy
Maltese		Do not allow these dogs to develop Small Dog Syndrome, human induced behaviors, ... Today, the glamorous Maltese is an adored pet and sought-after show dog. ...	8 lbs	18 to 24"	Toy
Bichon Frise		A Bichon Frise (French, literally meaning curly lap dog) is a small breed of dog of the Bichon type. They are popular pets, similar in appearance to, ...	7-12 lbs	9 - 12 inches	Non Sporting
Affenpinscher		Description, The Affenpinscher is a small dog with a harsh, shaggy coat, and longer hair all over the face. It is a smaller version of a working terrier and ...	7-10 lbs.	10 - 15 inches	Toys
Brussels Griffon		Brussels Griffon Breed Standard, Toy Group, General Appearance A toy dog, intelligent, alert, sturdy, with a thickset, short body, a smart carriage and ...	6-12 pounds	7-8 inches	Toy
Pomeranian		You may find Pomeranian puppies for sale and Pomeranian dogs for sale from quality dog ... She Is So Small. She Has A Gorgeous Little Baby Doll Face, ...	4lbs	7-12 inches	Toy
Havanese		The Havanese gives a rugged impression of a little dog, it is sturdy, and while a small breed, it is neither fragile nor overdone. ...	7-13	8-11	Toy
Australian Terrier		"Dedicated to the Advancement of Quality, Purebred Australian Terriers". Founded	12 - 14 lb	10 in	Terrier



DBpedia search powered by neofonie

enter search terms... Search

First | Previous | Next | Last

Results 7 to 12 of 12

Your Filters: item type Skyscraper, floor count 50 and up, building completed in year up to 2000, location Hong Kong

Highcliff
Highcliff is a 252.4-metre (828-foot) tall skyscraper located on a south slope of Happy Valley on the Hong Kong Island in Hong Kong. The 75 storey (70 floors of which are livable space) building's construction began in 2000 and was completed in 2003 under a design by DLN Architects & Engineers. It was the Silver Winner of the 2003 Emportis Skyscraper Award, coming in second to 30 St Mary Axe in London.

The Harbourside
The Harbourside is a 255 m (836.6 ft) tall residential skyscraper located at 1 Austin Road West in Union Square complex on Kowloon peninsula. The building is erected on the West Kowloon Reclamation west of Kwun Chung. Construction of the 74 storey building began in 2000 and was completed in 2003 under the design by P & T Architects & Engineers. The building is, in fact, three towers joined at the base, middle

The Text Mining Process

1. Text Preprocessing

- syntactic and/or semantic analysis

2. Feature Generation

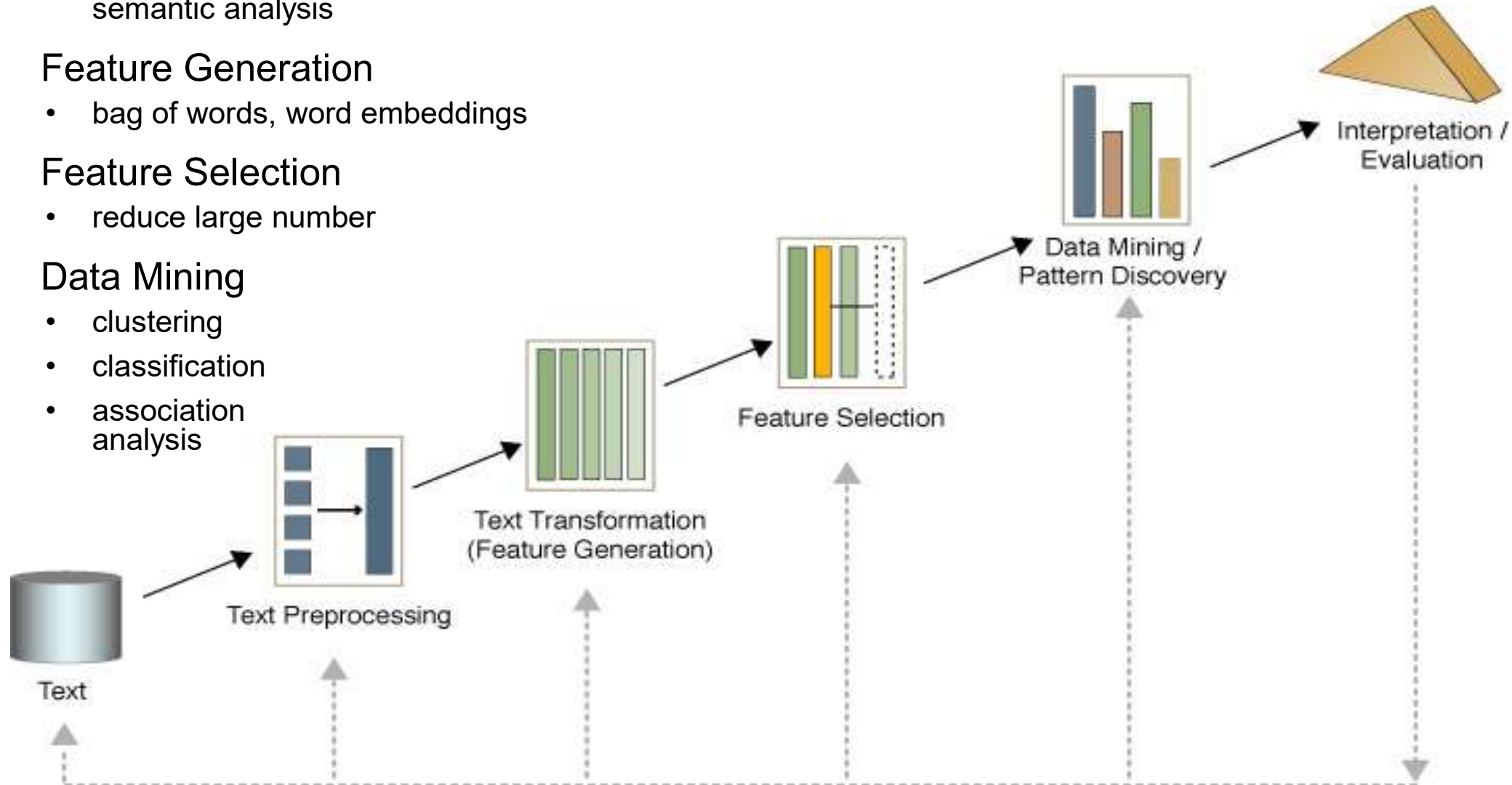
- bag of words, word embeddings

3. Feature Selection

- reduce large number

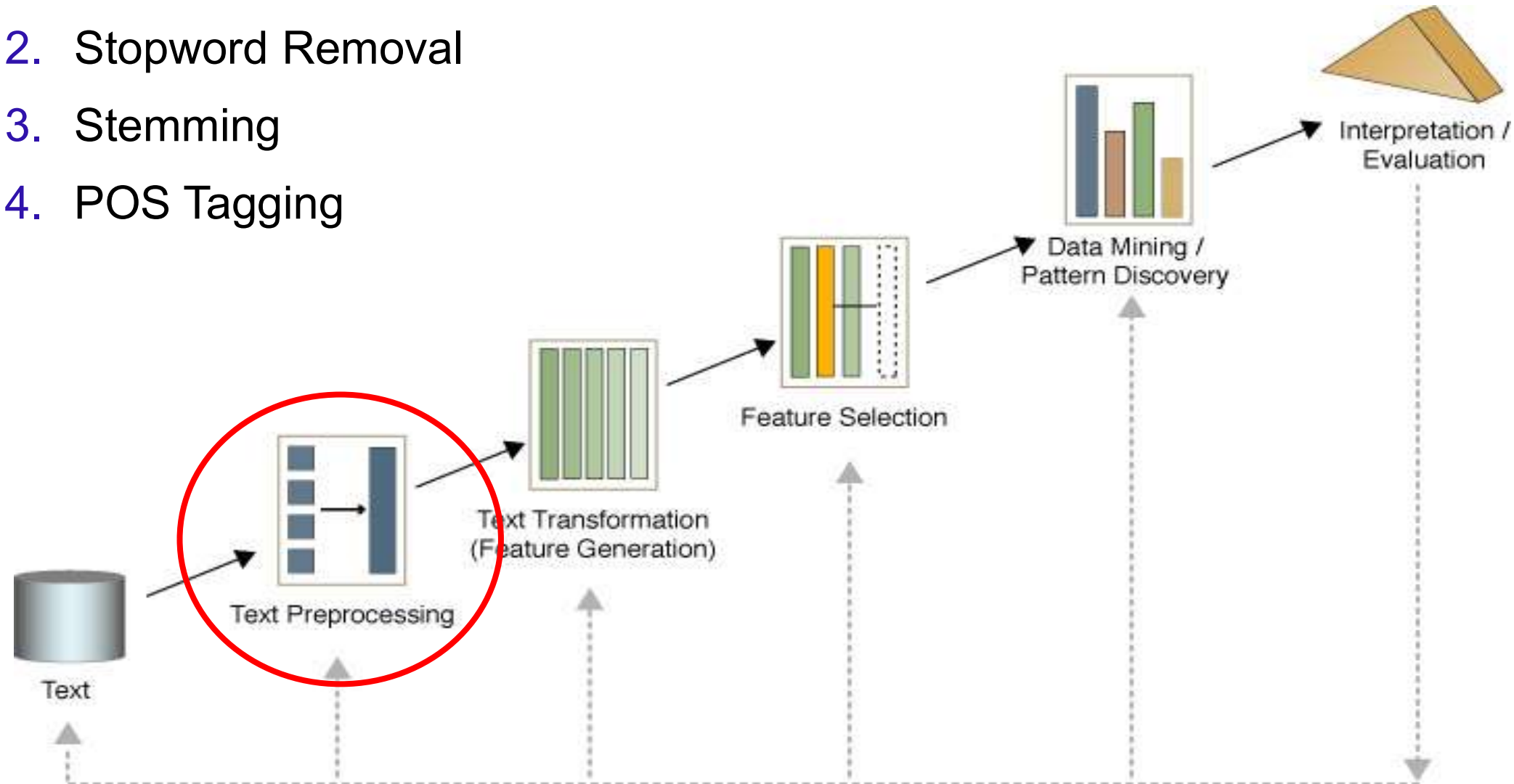
4. Data Mining

- clustering
- classification
- association analysis



2. Text Preprocessing

1. Tokenization
2. Stopword Removal
3. Stemming
4. POS Tagging



Syntactic and Linguistic Text Preprocessing

– Simple Syntactic Processing

- Text Cleanup (remove punctuation and HTML tags)
- Tokenization (break text into single words or N-grams)

– Advanced Linguistic Processing

• Word Sense Disambiguation

- determine which sense a word is having.
- normalize synonyms (United States, USA, US)
- 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)

Stopword Removal

- Many of the most frequently used words in English are likely to be **useless** for text mining
- These words are called **Stopwords**
 - examples: the, of, and, to, an, is, that, ...
 - typically text contains about 400 to 500 such words
 - for an application, an additional domain specific stopwords list may be constructed
- Why should we remove stopwords?
 - Reduce data set size
 - stopwords account for 20-30% of total word count
 - Improve effectivity of text mining methods
 - stopwords 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

Stemming

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

Some Basic Stemming Rules

– remove endings

- if a word ends with a consonant other than s, followed by an s, then delete s.
- if a word ends in es, drop the s.
- if a word ends in ing, delete the ing unless the remaining word consists only of one letter or of th
- if a word ends with ed, preceded by a consonant, delete the ed unless this leaves only a single letter
-

– transform words

- if a word ends with “ies” but not “eies” or “aies” then “ies → y”

Text Preprocessing in Python

- Simple preprocessing including stopwords removal

```
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>

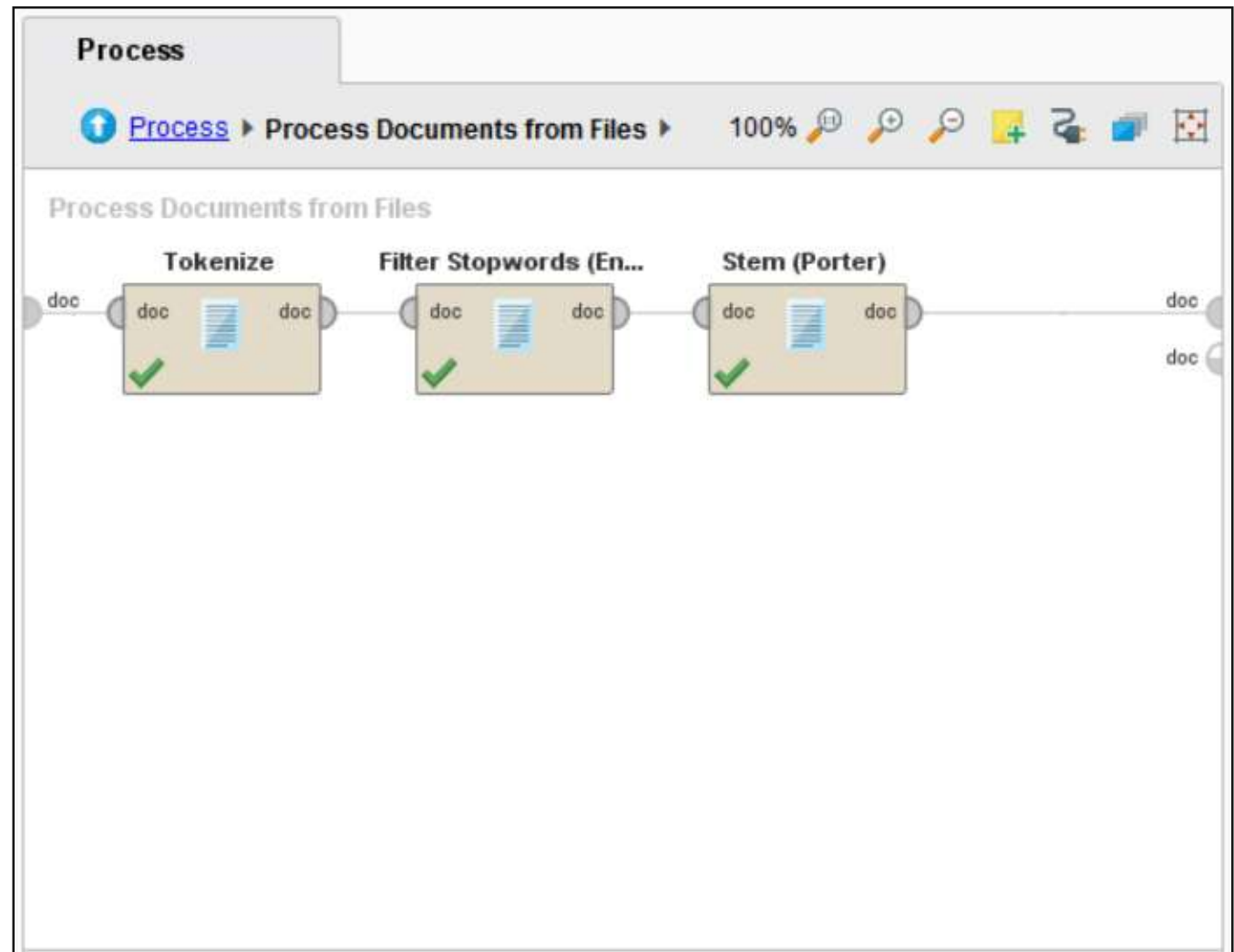
Text Preprocessing in RapidMiner

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

Process Documents from Files

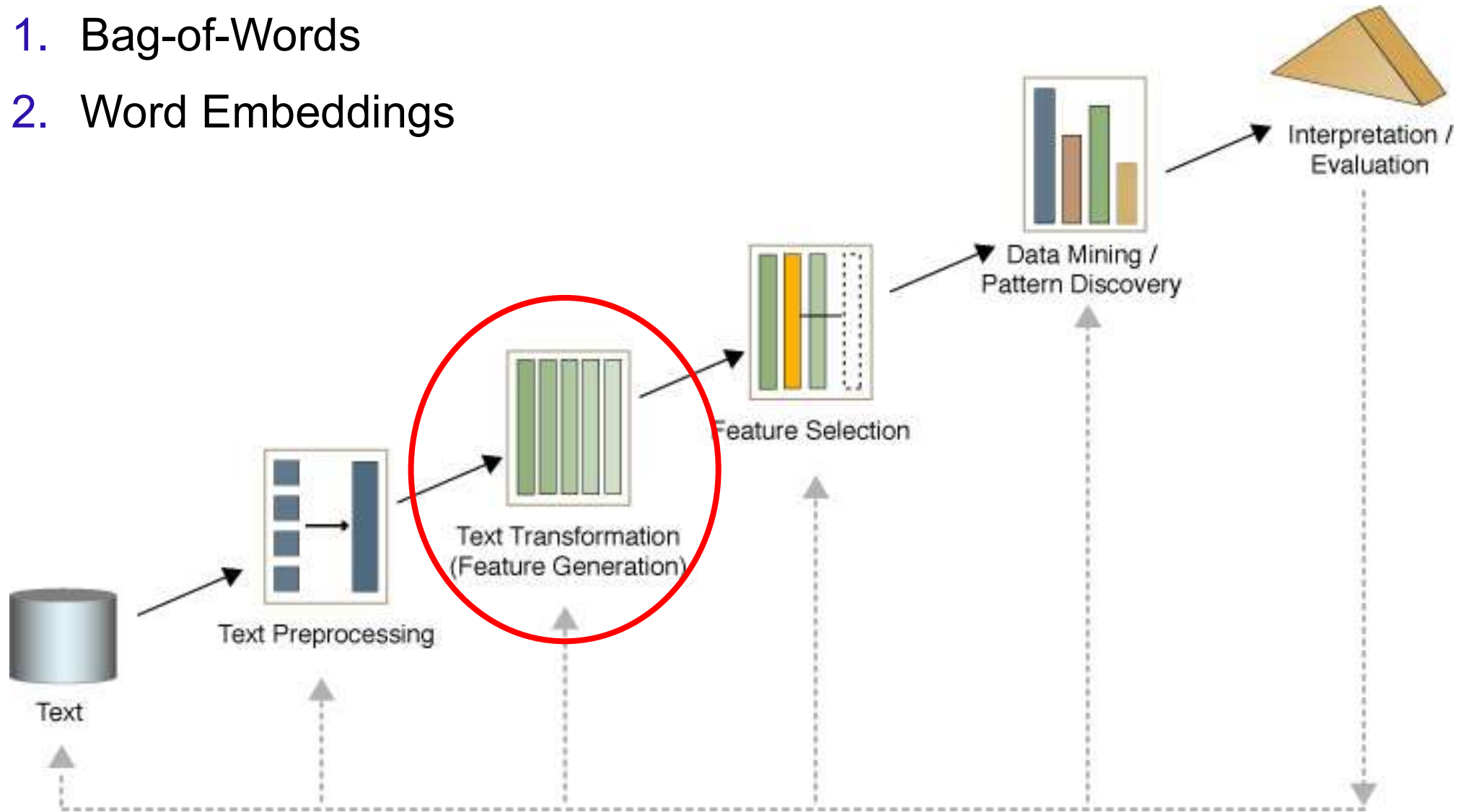


Process Documents from Data



3. Feature Generation

- 1. Bag-of-Words
- 2. Word Embeddings



Bag-of-Words: The Term-Document Matrix

Term	Dokument																				Σ
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	
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
dtrs	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

Bag-of-Words: Feature Generation

- Document is treated as a **bag of words** (or terms)
 - each word/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 (one-hot encoding)
 2. **Term Occurrence**: Number of occurrences 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)
- IDF: Inverse Document Frequency.

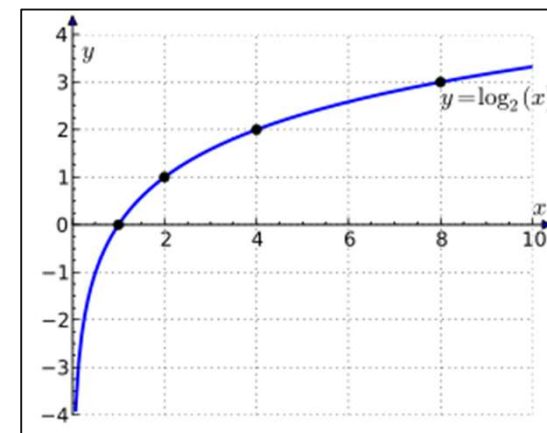
N : total number of docs in corpus

df_i : the number of docs in which t_i appears

$$w_{ij} = tf_{ij} \times idf_i.$$

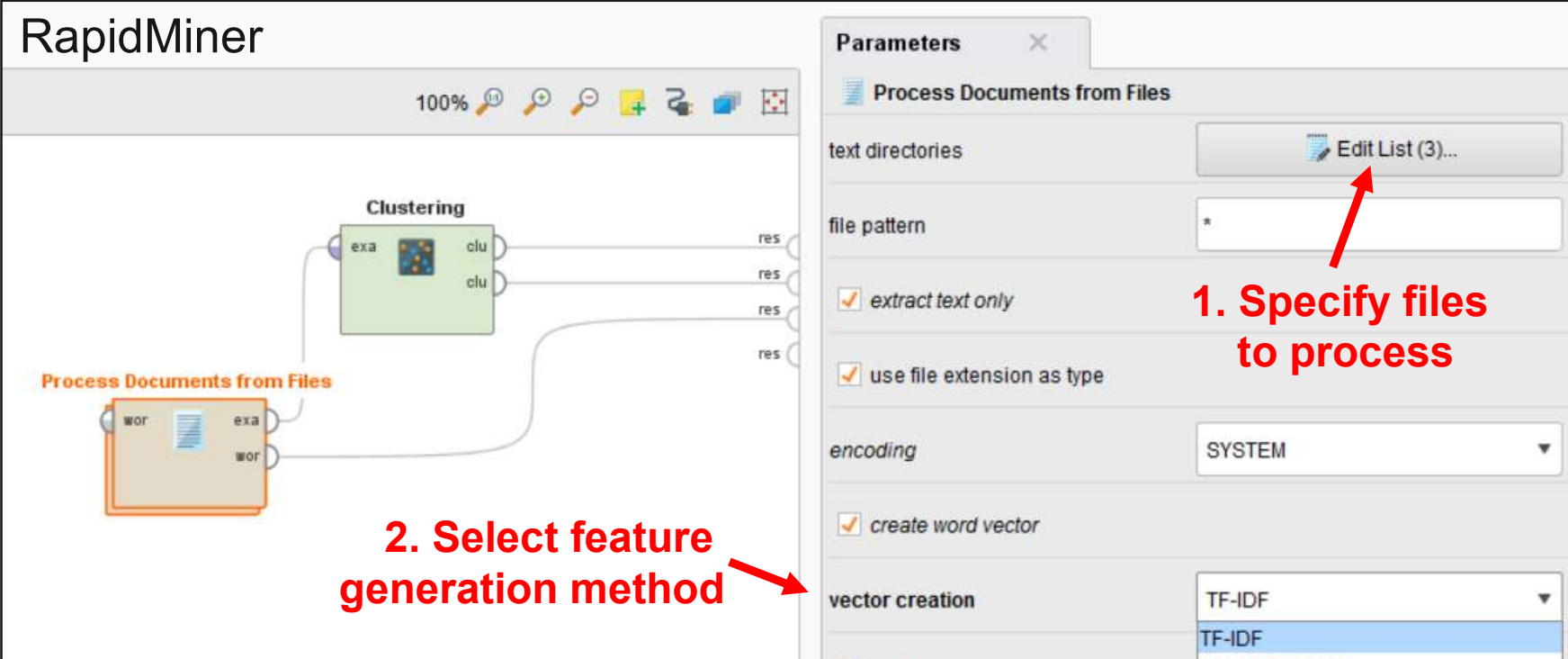
$$idf_i = \log \frac{N}{df_i}$$

- Gives more weight to rare words
- Give less weight to common words (domain-specific stopwords)



Feature Generation in RapidMiner and Python

RapidMiner



1. Specify files to process

2. Select feature generation method

Parameters

Process Documents from Files

text directories

file pattern *

extract text only

use file extension as type

encoding SYSTEM

create word vector

vector creation

- TF-IDF
- TF-IDF**
- Term Frequency
- Term Occurrences
- Binary Term Occurrences

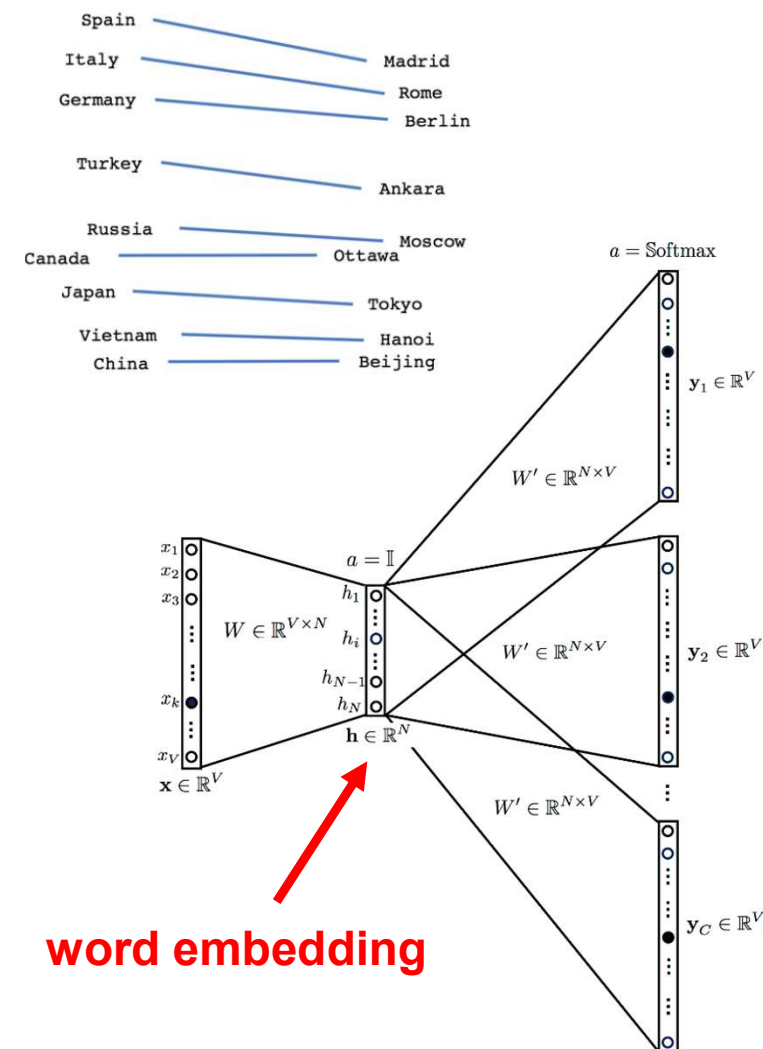
Python

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

binary_term_occurrences_vectorizer = CountVectorizer(binary='True')
term_occurrences_vectorizer = CountVectorizer(binary='False')
term_frequency_vectorizer = TfidfVectorizer(use_idf='False')
tf_idf_vectorizer = TfidfVectorizer(use_idf='True')
```

Word Embeddings

- Embeddings represent words not as a single number in a word vector but represent each word as a vector of real numbers (**distributed representation**)
e.g. 300 numbers
- Embeddings are chosen in a way that **semantically related words** (e.g. dog, puppy) end up at similar locations in the vector space
 - thus, embeddings can deal better with synonyms and related terms than bag-of-words vectors
- Embeddings are calculated based on the assumption that similar words appear in similar contexts (**distributional similarity**)
 - Skip-gram approach used by Word2Vec: predict context words for each word using a neural net



Embedding Methods and Pretrained Models

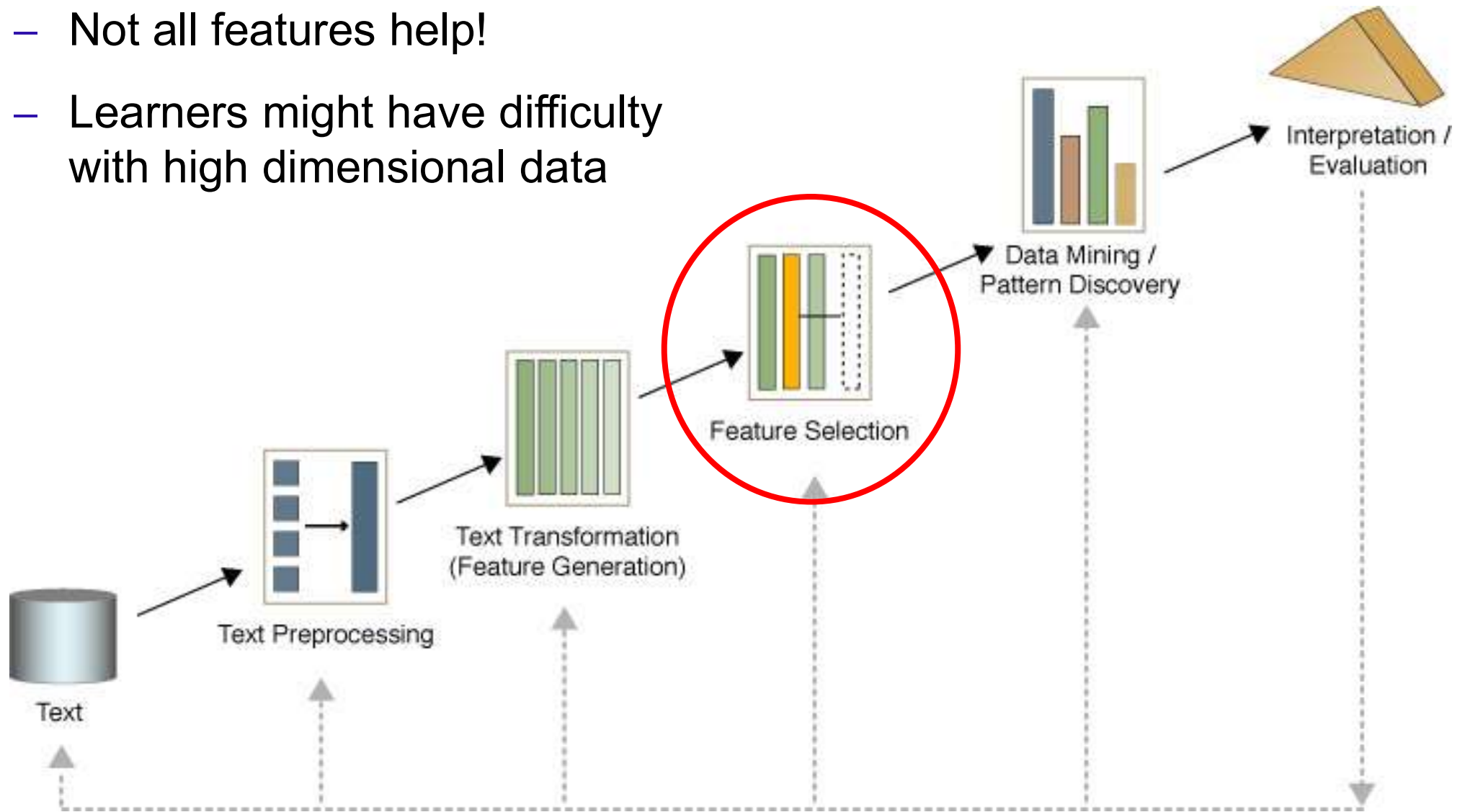
- Well known word embedding methods
 - **Word2Vec** (Google)
 - **GloVe** (Stanford NLP Group)
 - **fastText** (Facebook AI Research)
- pretrained embeddings can be downloaded
 - GloVe: trained on Common Crawl, Wikipedia, and Tweets
 - fastText: embeddings for 294 languages
- using Embeddings
 - Python: Gensim offers Word2Vec implementation
 - RapidMiner: Word2Vec extension on marketplace
- embeddings are used as features in downstream tasks
 - together with traditional symbolic methods as part of neural architectures

GloVe 50 embedding of the word **the** pretrained on Common Crawl

```
the
0.418 0.24968 -0.41242 0.1217
0.34527 -0.044457 -0.49688 -
0.17862 -0.00066023 -0.6566
0.27843 -0.14767 -0.55677 0.14658
-0.0095095 0.011658 0.10204 -
0.12792 -0.8443 -0.12181 -
0.016801 -0.33279 -0.1552 -
0.23131 -0.19181 -1.8823 -0.76746
0.099051 -0.42125 -0.19526 4.0071
-0.18594 -0.52287 -0.31681
0.00059213 0.0074449 0.17778 -
0.15897 0.012041 -0.054223 -
0.29871 -0.15749 -0.34758 -
0.045637 -0.44251 0.18785
0.0027849 -0.18411 -0.11514 -
0.78581
```

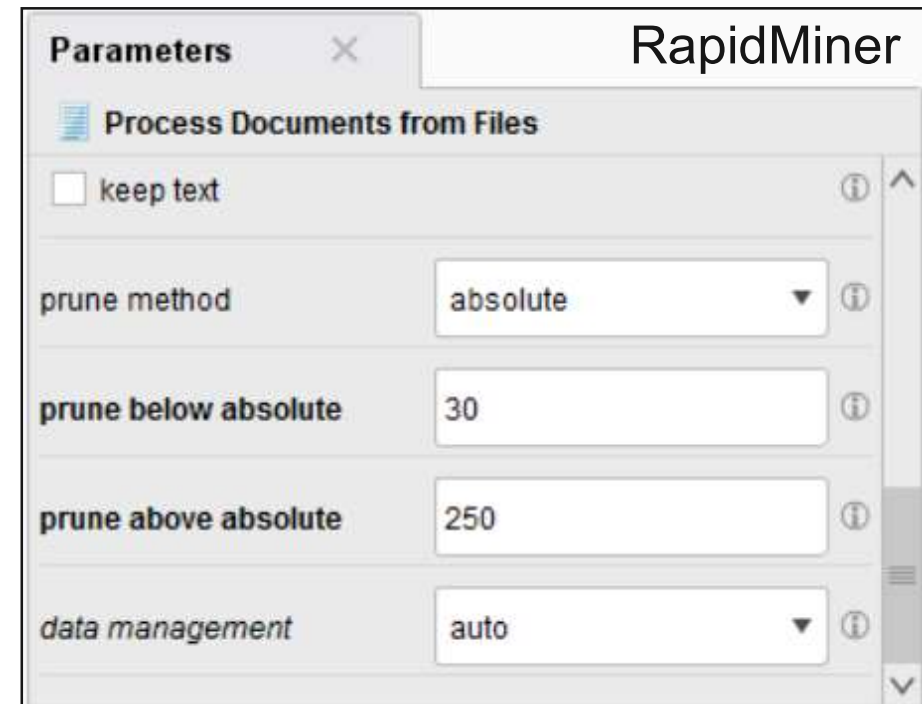
4. Feature Selection

- Not all features help!
- Learners might have difficulty with high dimensional data



Pruning Document Vectors in RapidMiner and Python

- Prune 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

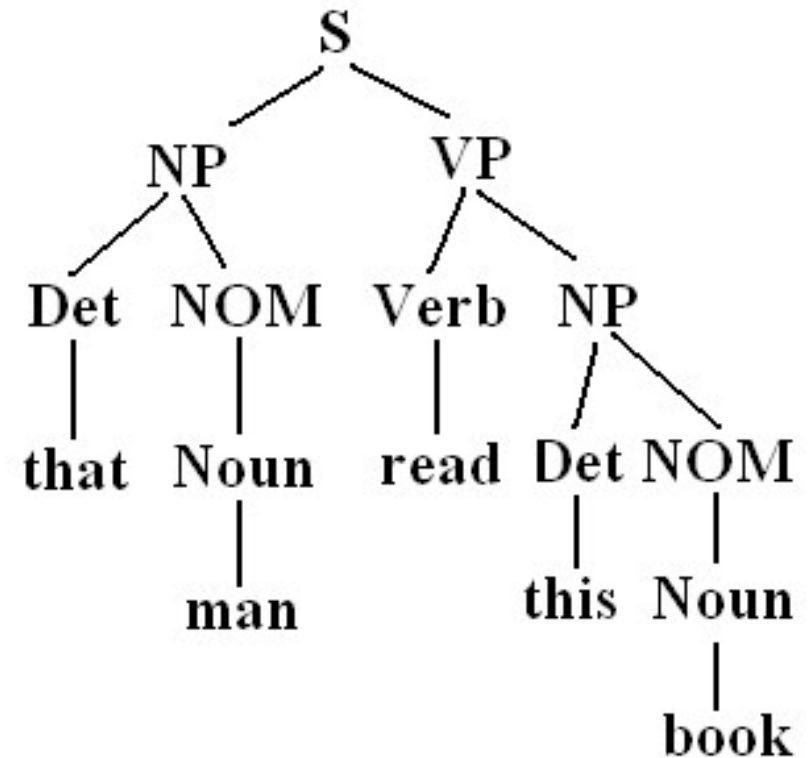


Python

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

Filter Tokens by POS Tags

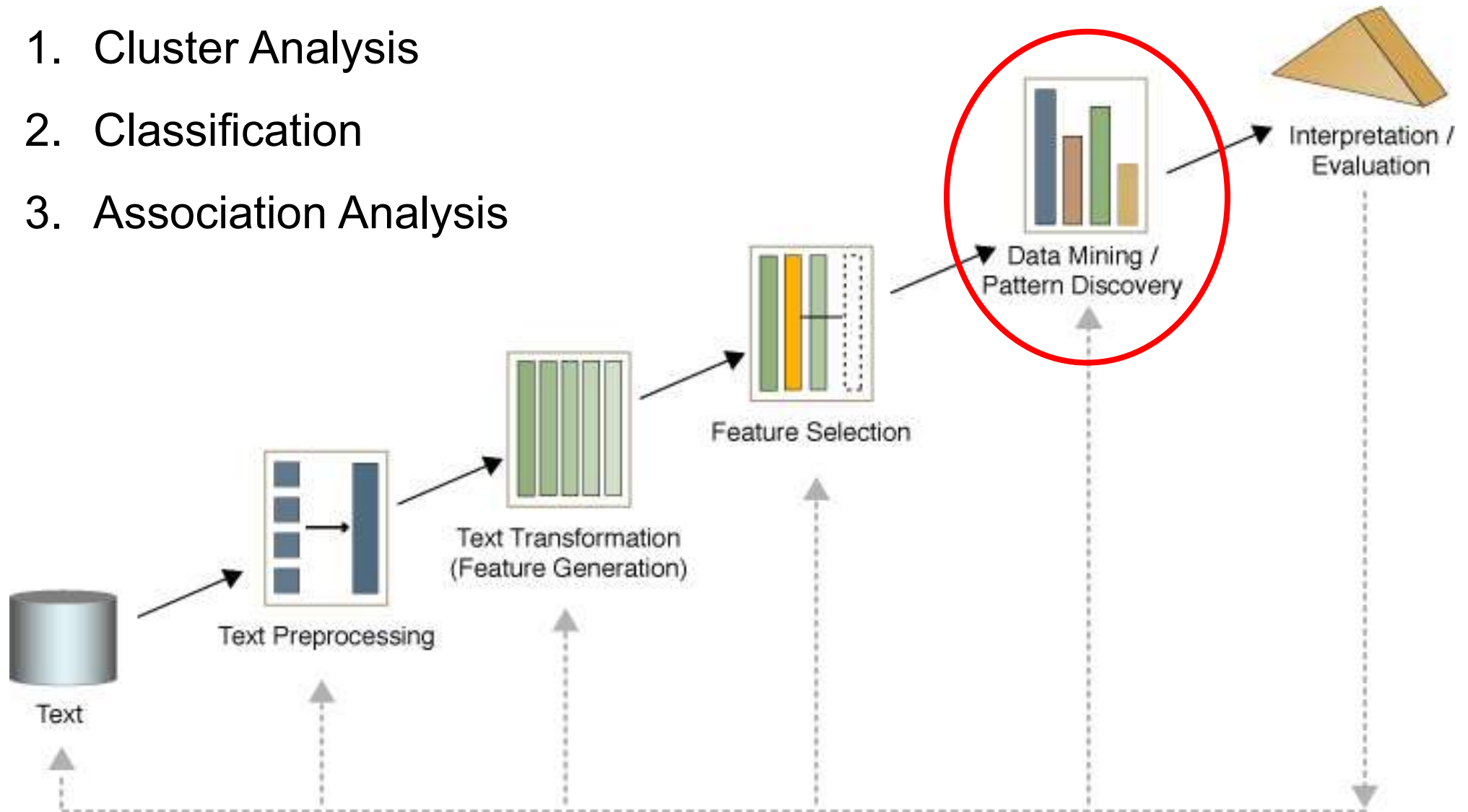
- POS tagging may be helpful for feature selection
- sometimes you want to focus on certain classes of words:
 - Adjectives (JJ.) for sentiment analysis
 - good, bad, great
 - Nouns (N.) for text clustering
 - red and blue cars are similar
 - red and blue trousers are similar
- Rapidminer supports
 - PENN tag system for English
 - STTS tag system for German
 - filtering conditions are expressed as regular expressions
- Python: NLTK library supports PENN tag system for English



5. Solving Downstream Tasks

Methods:

1. Cluster Analysis
2. Classification
3. Association Analysis



5.1 Document Clustering

Goal

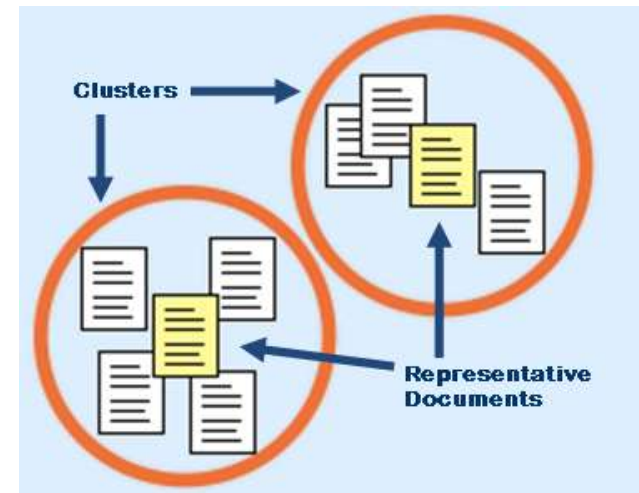
- Given a **set of documents** and a **similarity measure** among documents find clusters such that
 - documents in one cluster are more similar to one another
 - documents in separate clusters are less similar to one another
- using some clustering algorithm

Applications

- Topical clustering of news stories
- Topical clustering of social media posts
- Grouping of document versions

Question

- Which similarity measures are a good choice for comparing document vectors?



Jaccard Coefficient

- The **Jaccard coefficient** is a popular similarity measure for vectors consisting of asymmetric binary attributes

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

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

- used together with **binary term occurrence vectors** (one-hot vectors)
 - 1 represents occurrence of specific word
 - 0 represents absence of specific word
 - most values are 0 as only small subset of the vocabulary is used in a document

Example: Jaccard Coefficient

- Example 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 binary term occurrence vectors

	Saturn	is	the	gas	planet	with	rings	Jupiter	largest	Roman	god	of	sowing
d1	1	1	1	1	1	1	1	0	0	0	0	0	0
d2	0	1	1	1	1	0	0	1	1	0	0	0	0
d3	1	1	1	0	0	0	0	0	0	1	1	1	1

- Jaccard similarities between the documents

- $\text{sim}(d1,d2) = 0.44$

- $\text{sim}(d1,d3) = 0.27$

- $\text{sim}(d2,d3) = 0.18$

Cosine Similarity

- Popular similarity measure for comparing weighted document vectors such as **term-frequency** or **TF-IDF vectors**

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

where \bullet indicates vector dot product and $\|d\|$ is the length of vector d

$$a \cdot b = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$$

$$\|a\| = \sqrt{\sum_{i=1}^N a_i^2}$$

- Example

$$d_1 = 3 \ 2 \ 0 \ 5 \ 0 \ 0 \ 0 \ 2 \ 0 \ 0$$

$$d_2 = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 2$$

$$d_1 \bullet d_2 = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5$$

$$\|d_1\| = (3*3 + 2*2 + 0*0 + 5*5 + 0*0 + 0*0 + 0*0 + 2*2 + 0*0 + 0*0)^{0.5} = (42)^{0.5} = 6.481$$

$$\|d_2\| = (1*1 + 0*0 + 0*0 + 0*0 + 0*0 + 0*0 + 0*0 + 1*1 + 0*0 + 2*2)^{0.5} = (6)^{0.5} = 2.245$$

$$\cos(d_1, d_2) = 0.3150$$

Example: 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:

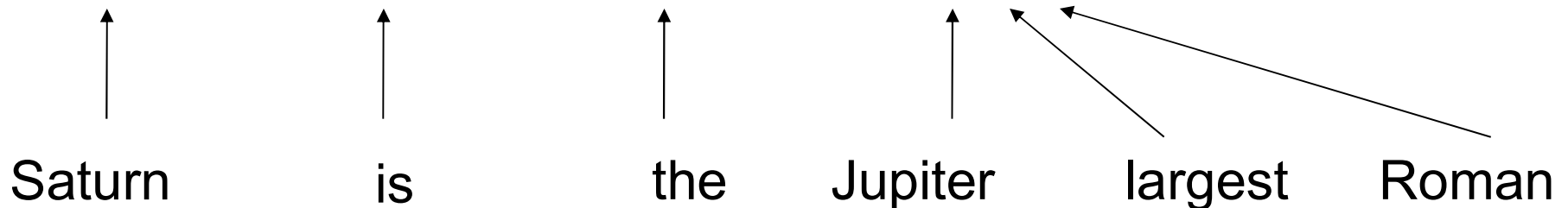
- “Saturn is the gas planet with rings.”
- “Jupiter is the largest gas planet.”
- “Saturn is the Roman god of sowing.”

$$w_{ij} = tf_{ij} \times idf_i.$$

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

- First document as TF-IDF vector:

$(1/7 * \log(3/2), 1/7 * \log(3/3), 1/7 * \log(3/3), \dots, 0, 0, 0, \dots)$



Example: 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 TF-IDF vectors

	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

- Cosine similarities between the documents
 - $\cos(d1,d2) = 0.13$
 - $\cos(d1,d3) = 0.05$
 - $\cos(d2,d3) = 0.00$

Embedding-based Similarity

1. Translate documents into embedding vectors

- using for example doc2vec or
- average embeddings of all words in the document

2. Calculate similarity of document embedding vectors

- using dot product or cosine similarity
- apply nearest neighbor search

Similarity of two words

Given two words, this demo gives the similarity value between 1 and -1.

Restaurant Eatery Show similarity

0.72302246

Given two words, this demo gives the similarity value between 1 and -1.

Restaurant Resource Show similarity

0.14613703

– Useful libraries

- Gensim: offers doc2vec and word2vec implementations
- Faiss: offers scalable, approximate nearest neighbor search methods (used for information retrieval)

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

5.2 Document Classification

- 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
- **Applications**
 - topical classification of news stories or web pages
 - SPAM detection
 - sentiment analysis, hate speech detection
- **Classification methods** commonly used for text
 1. naive bayes
 2. support vector machines (SVMs)
 3. recurrent neural networks (RNNs), e.g. long short-term memory (LSTMs)
 4. Transformers like BERT
 5. but KNN or random forests may also work

Example Application: Sentiment Analysis



- Given: A text
- Goal: Assign a class of sentiment to the text
 - e.g., positive, (neutral,) negative
 - e.g., sad, happy, angry, surprised
- Can be implemented as supervised classification task
 - requires training data
 - i.e., pairs like <text; sentiment label>

Example Application: Sentiment Analysis

- Labeling data for sentiment analysis
 - is expensive, like all data labeling tasks
- Reviews from the Web may be used as labeled data (distant supervision)

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

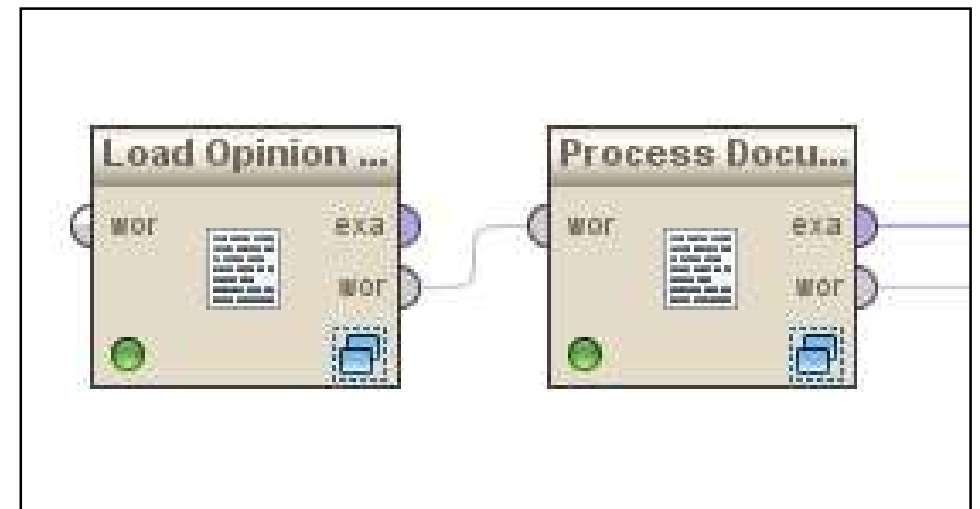
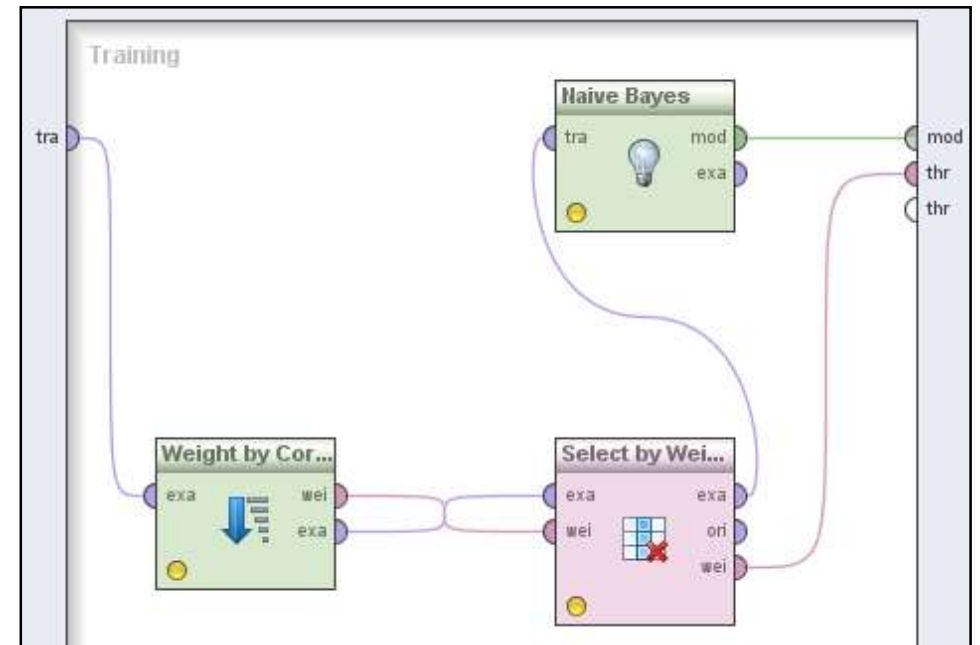
- There exist various large corpora of reviews for public download
 - Amazon Product Data by Julian McAuley: 142 million reviews from Amazon
 - WebDataCommons: 70 million reviews from 50,000 websites that use RDFa or Microdata markup

Preprocessing for Sentiment Analysis

- Recap – we started our processing with: Simple Syntactic Analysis
 - text cleanup (remove punctuation, HTML tags, ...)
 - normalize case
 - ...
- However, reasonable features for sentiment analysis might include
 - punctuation “!”, “?”, “?!”
 - smileys encoded using punctuation: e.g. ;-) :- (
 - use of visual markup, where available (red color, bold face, ...)
 - amount of capitalization (“SCREAMING”)
- Practical Approach
 - Replace smileys or visual markup with sentiment words in preprocessing
 - 😊 → great, COOL → cool cool

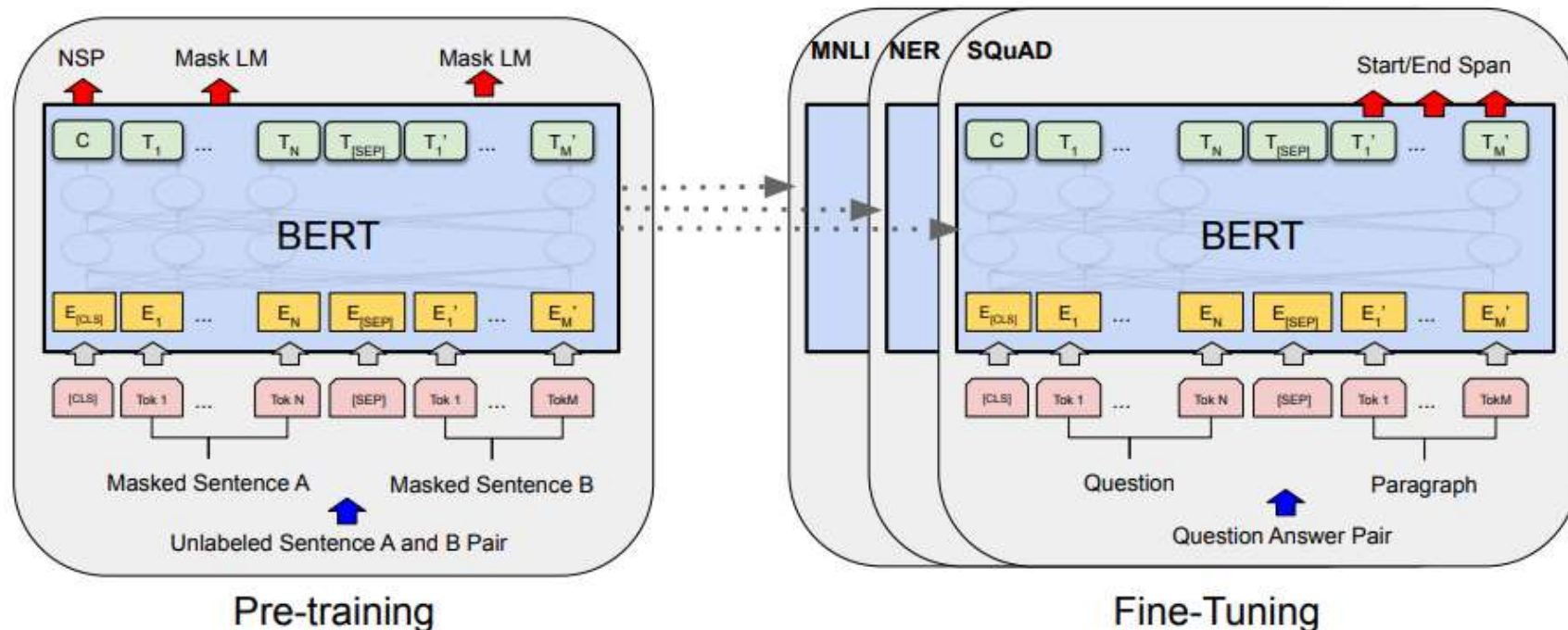
Sentiment Analysis Tricks

- Finding selective words
 - weight words according to their correlation with label
 - select top-k words with highest correlation
- Sentiment lexicons
 - use external dictionary of opinion words
 - Bing Liu's lexicon
<http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar>
 - AFINN lexicon
<https://github.com/fnielsen/afinn>
 - restrict RapidMiner word list to these words or combine them with your learned sentiment words



5.3 Pre-Trained Language Models

- introduce **pre-training**, **fine-tuning** paradigm
 - pre-trained on large text corpora
 - fine-tune model for specific downstream tasks
 - BERT model size: 60 to 345 million parameters
- outperform previous models on most NLP tasks

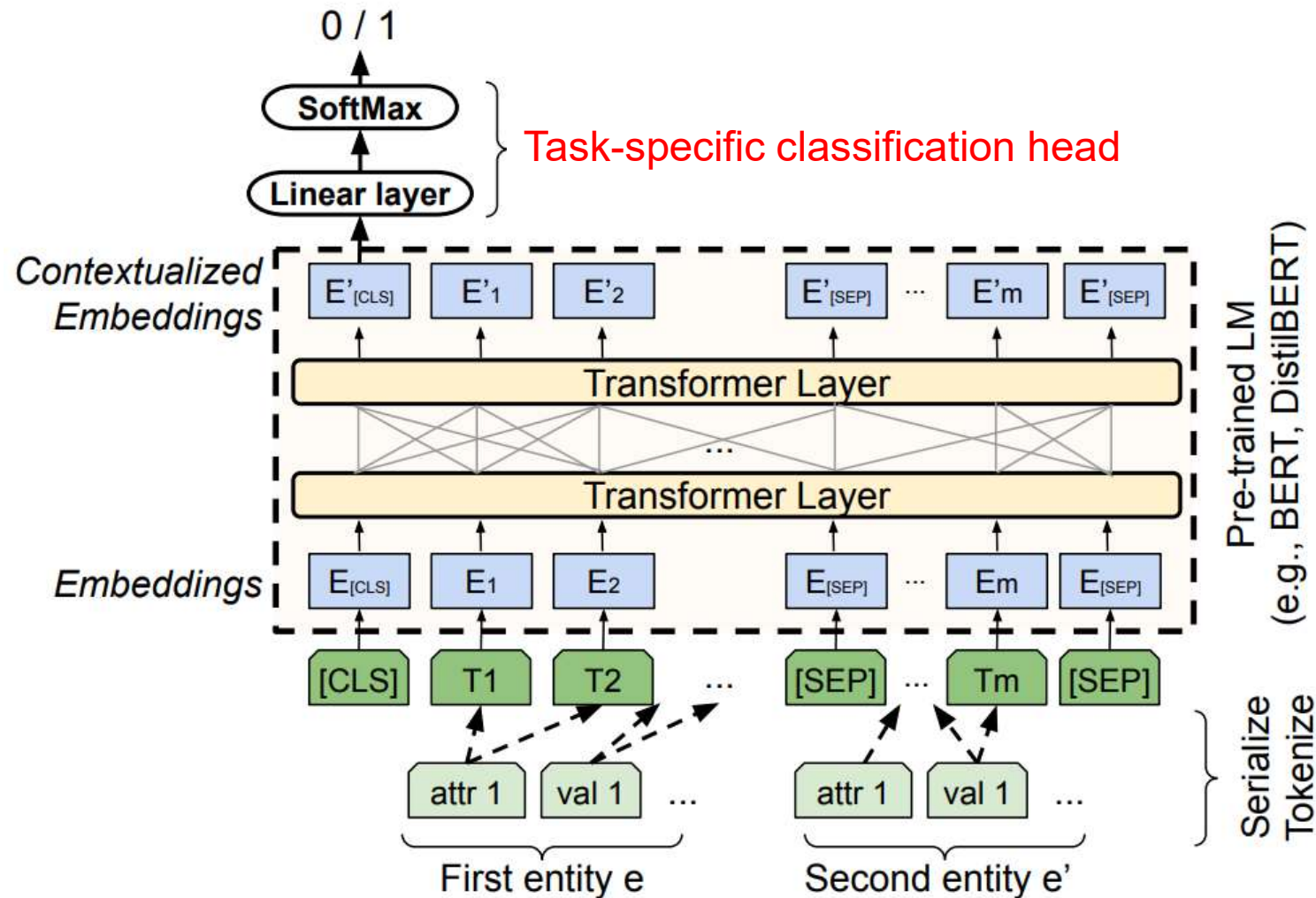


Example: BERT for Entity Matching

- DITTO applies BERT, DistilBERT and RoBERTa for determining if two records describe the same real-word entity.
 - Classes: Match / Non-Match
- Entity serialization for BERT
 - Pair of entity descriptions (records) are turned into a single sequence
 - [CLS] Entity Description 1 [SEP] Entity Description 2 [SEP]
 - Entity Description = [COL] attr₁ [VAL] val₁ . . . [COL] attr_k [VAL] val_k
 - Serialization Example =
[CLS][COL] Name [VAL] Franz Müller [COL] Birthdate [VAL] 20/08/2002[SEP]

Yuliang, et al: **Deep entity matching with pre-trained language models**. PVLDB 2021.

DITTO: Architecture



- $[CLS]$ token summarizes the pair of entities
- linear layer on top of $[CLS]$ token for matching decision

DITTO: Evaluation

Type	Dataset	DITTO F1	DeepMatcher F1	Magellan F1
Structured	iTunes-Amazon	97.0	88.5 -8.5	91.2 -5.8
	DBLP-ACM	99.0	98.4 -0.6	98.4 -0.6
	DBLP-Scholar	95.6	92.3 -3.3	94.7 -0.9
	Walmart-Amazon	86.8	66.9 -19.9	71.9 -14.9
Textual	Abt-Buy	89.3	62.8 -26.5	43.6 -45.7
	Amazon-Google	75.6	69.3 -6.3	49.1 -26.5
	WDC Computer - Large	91.7	89.5 -3.2	64.5 -27.2
	WDC Computer - Small	80.8	70.5 -10.3	57.6 -23.2

- constant improvement for structured data
- large performance gain for textual data

Potential Reasons for the Performance Gain

- Serialization allows to pay attention to all attributes
 - no strict separation between attributes
- WordPiece tokenizer breaks unknown terms into pieces
 - no problems with out of vocabulary terms
- Transfer learning from pre-training texts
 - different surface forms are already close in embedding space
- Contextualization of the embeddings
 - potentially more suited for capturing differing semantics
- using Transformers for your projects
 - Hugging Face Library: code for fine-tuning Transformers
 - Hugging Face Model Hub: download pre-trained models



Hugging Face

Summary

- Main challenge in text mining: Preprocessing and vectorization
- There are lots of alternative techniques
 - Thus, you need to experiment in order to find out which work well for your use case
 - focus has shifted from bag-of-words approaches to embeddings and Transformers for end-to-end learning
- Currently lots of ground-breaking innovation triggered by generative transformer models
- **Courses on Text Mining**
 - IE 661: Text Analytics (Strohmaier)
 - IE 696: Advanced Methods in Text Analytics (Ponzetto)
 - IE663: Information Retrieval and Web Search (Ponzetto)



References

– Excellent Online Course

- Stanford CS224N: Natural Language Processing with Deep Learning
 - videos: <https://www.youtube.com/playlist?list=PLoROMvodv4rOSH4v6133s9LFPRHjEmbmJ>
 - slides: <https://web.stanford.edu/class/cs224n/>

– Textbook

- Jurafsky & Martin: Speech and Language Processing. 2nd edition. Pearson International Edition.
- Draft version of 3rd edition available online <https://web.stanford.edu/~jurafsky/slp3/>

– Useful tools

- Stanford CoreNLP: collection of neural as well as traditional NLP tools
- Gensim: Python implementation of word2vec
- HuggingFace: Transformer library and model hub

