Introduction to Text Mining

Jelena Jovanovic

Email: jeljov@gmail.com

Web: http://jelenajovanovic.net

OVERVIEW

- Briefly about Text Mining: what, why, where?
- Text Mining workflow
  - Text pre-processing
  - Text transformation and feature engineering
    - Bag of Words & Vector Space Model (VSM)
    - Extensions of traditional VSM
    - Word embeddings
    - Further options for feature creation
  - Feature selection
  - Application of ML methods on transformed text
- Recommended links

What is Text Mining (TM)?

The use of computational methods and techniques to *automate extraction* of *high quality information* from text

→ high quality ≅ relevant, novel, interesting

A computational approach to the *discovery of new, previously unknown information and/or knowledge* through *automated extraction* of information from often *large amounts* of *unstructured text*

What is TM?

To achieve its goals, TM approaches combine methods from several related fields, including

- Statistical analysis,
- Machine learning,
- Computational linguistics,
- Information extraction,
- Graph analysis,
- Data visualization,
- …

WHY IS TM RELEVANT / USEFUL ?

Unstructured text is present in various (digital) forms, and in huge and ever increasing quantities:

- (e-)books,
- various kinds of business and administrative documents,
- news articles,
- blog posts,
- reviews,
- messages / posts on social networking and social media sites,
- lecture materials,
- student-authored content: essays, question answers, notes, …
- …

TM APPLICATION DOMAINS

- Document classification*
  - E.g., classifying text according to main theme, intent, sentiment, …
- Clustering of documents
  - E.g., to facilitate topic-based document search / exploration; to identify different cohorts of users / customers
- Document summarization
- Making predictions
  - E.g., predicting churn or stock market prices based on the analysis of posts on social networks
- Content-based recommender systems
  - recommendation of news articles, movies, books, articles, …

*The term *document* refers to any kind of unstructured text: forum or blog post, news article, tweet, review, …

CHALLENGES FACED BY TM: THE COMPLEXITY OF UNSTRUCTURED TEXT

In general, interpretation / comprehension of unstructured content (text, images, videos) is (often) easy for people, but very complex for computer programs

CHALLENGES FACED BY TM: THE COMPLEXITY OF UNSTRUCTURED TEXT

Difficulties with automated text comprehension are caused by the fact that human / natural language:

- is full of ambiguous terms and phrases
- often strongly relies on the context and background knowledge for defining and conveying meaning
- is full of fuzzy and probabilistic terms and phrases
- strongly based on commonsense knowledge and reasoning
- is influenced by and is influencing people's mutual interactions

CHALLENGES FACED BY TM: DIFFICULTIES IN APPLYING ML METHODS

The use of supervised machine learning (ML) methods (e.g. classification) for TM is often very expensive

- This is caused by the need to prepare high number of annotated (labelled) documents to be used as the training dataset
- Such a training set is essential for, e.g., document classification or extraction of entities, relations, and events from text

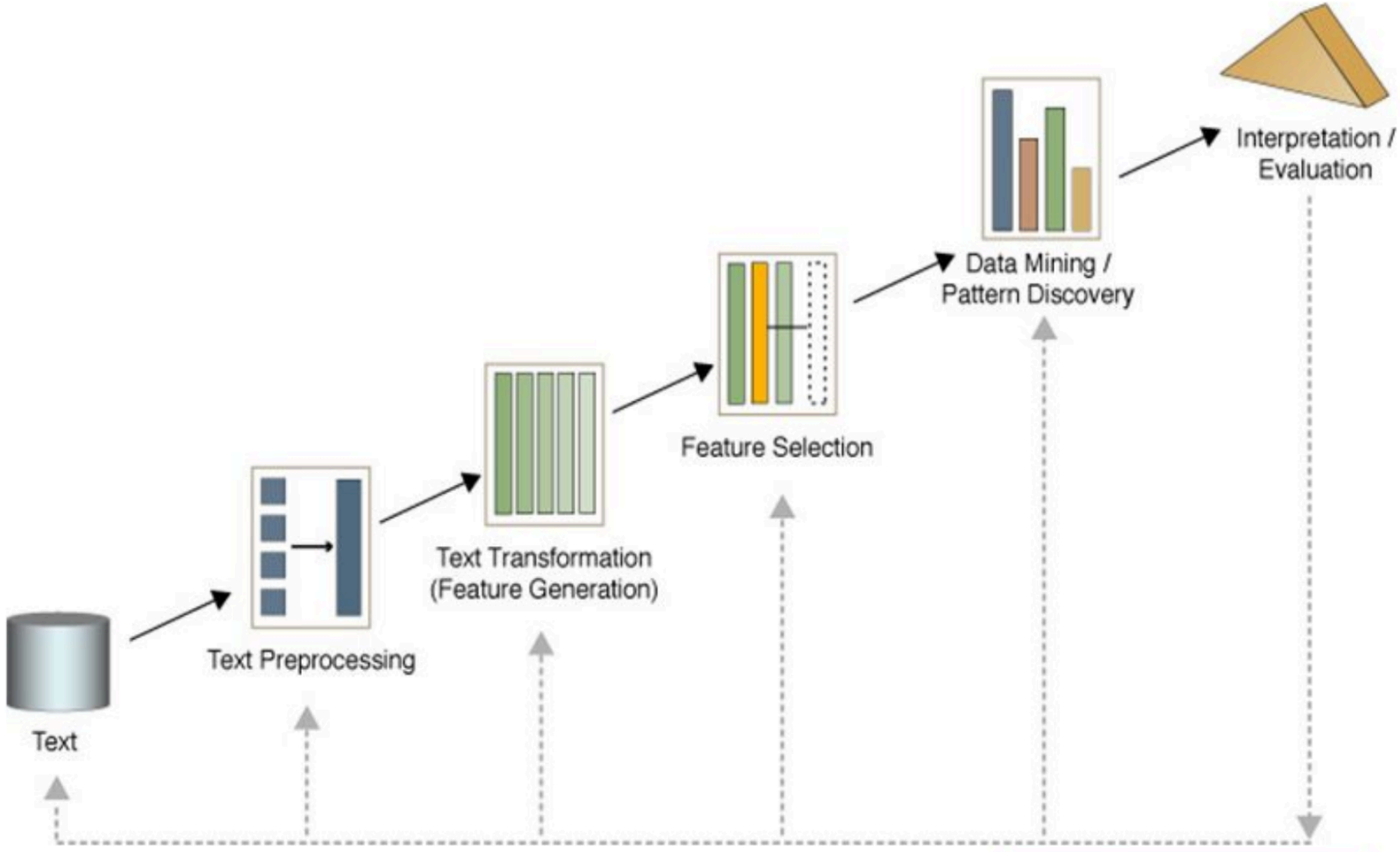
CHALLENGES FACED BY TM: DIFFICULTIES IN APPLYING ML METHODS

High-dimensionality of the attribute (feature) space and attributes' dependence on the corpus* terminology

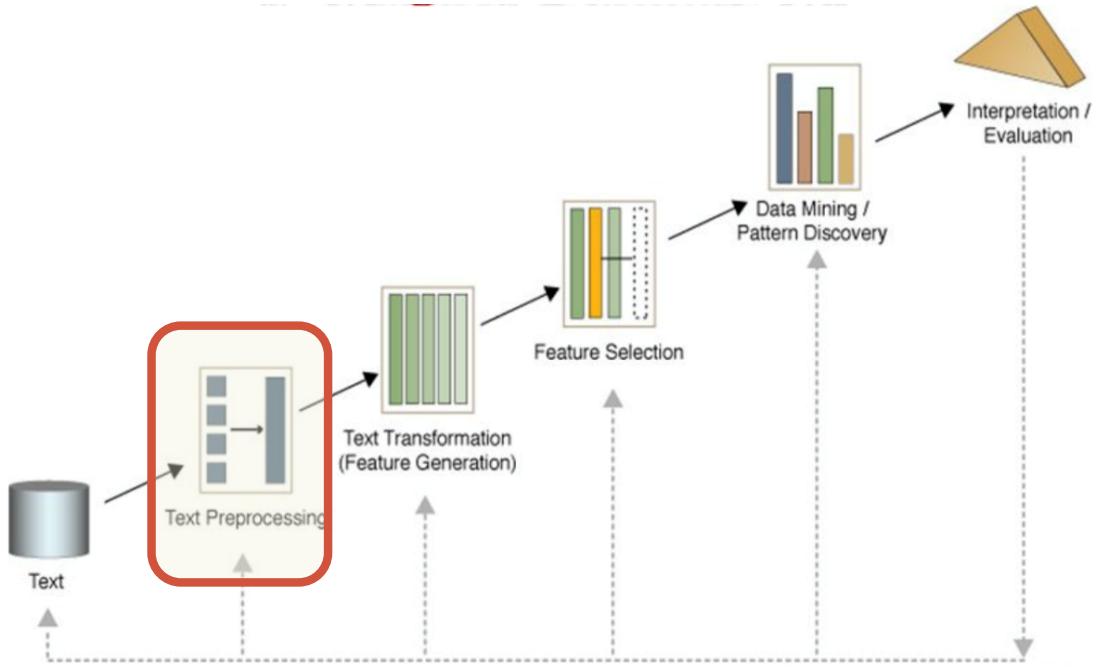
- Documents are often described with a large attribute set derived from the corpus being analyzed
  - E.g. terms (n-grams) from the corpus
- This makes the models susceptible to overfitting and stresses the importance of
  - collecting (sufficiently) representative training set
  - choosing an appropriate attribute selection method

*_corpus_ is a collection of documents that is the subject of analysis

MAIN STEPS OF THE TM WORKFLOW



Source: http://slideplayer.info/slide/10731087/



TEXT PREPROCESSING

TEXT PREPROCESSING

- The objective:

  to reduce the original set of words to a subset that is the most representative, hence, the most relevant, for the given corpus

- It is the essential first step for any TM task

TEXT PREPROCESSING

Typically, it consists of:

- Text normalization
- Removal of terms with overly high/low frequency of occurrence in the given corpus
- Removal of stop-words
- POS tagging
- Stemming / Lemmatization

TEXT NORMALIZATION

- Objective: transform various forms of the same term into a common, ‘normalized’ form
- For example:
  - Apple, apple, APPLE → apple
  - Intelligent Systems, Intelligent systems, Intelligent-systems → intelligent systems

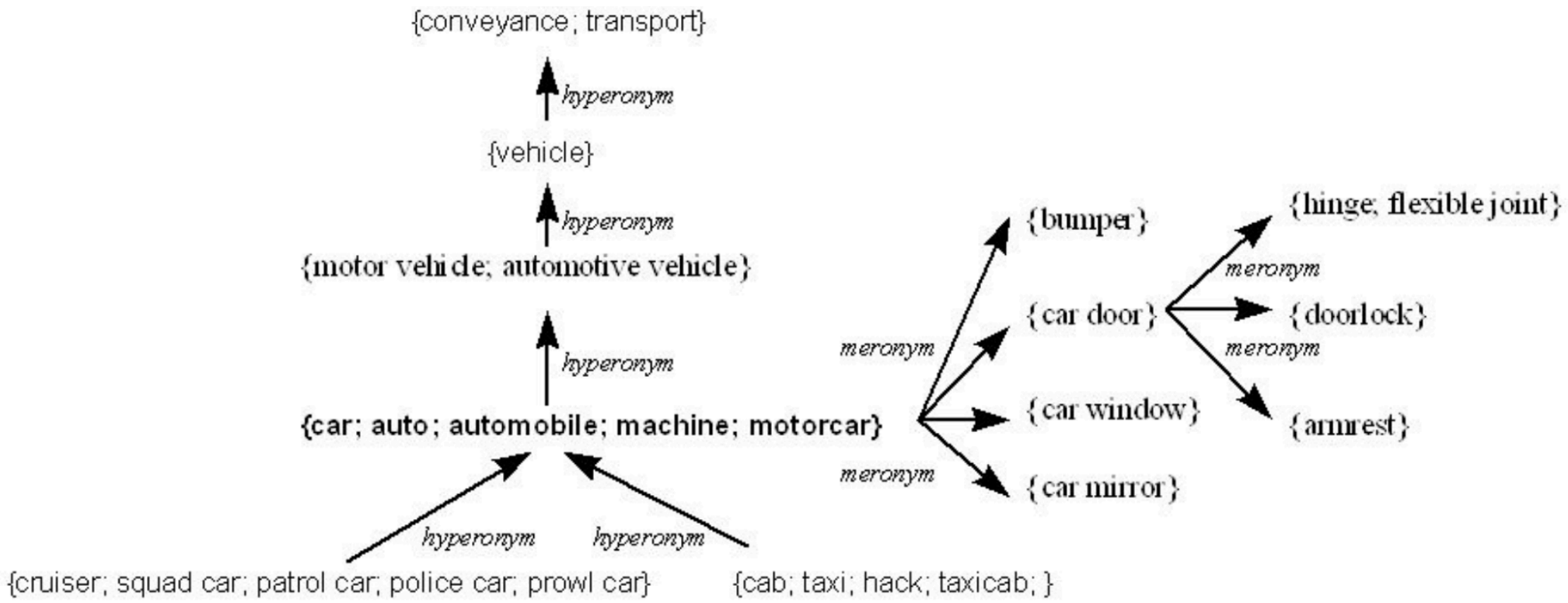
TEXT NORMALIZATION

How it is done:

- Using simple rules, such as:
  - Removing all punctuation marks (dots, dashes, commas,…)
  - Reducing all words to lower case
- Correction of spelling mistakes (misspellings)
  - E.g. matching words against one of the available misspelling corpora
- Using a morphological knowledge base, such as WordNet, to replace synonyms with common concepts
  - E.g., automobile, car → vehicle

WORDNET

An illustration of the WordNet structure



Source: http://slideplayer.com/slide/5196/

REMOVING TERMS WITH OVERLY HIGH / LOW FREQUENCY

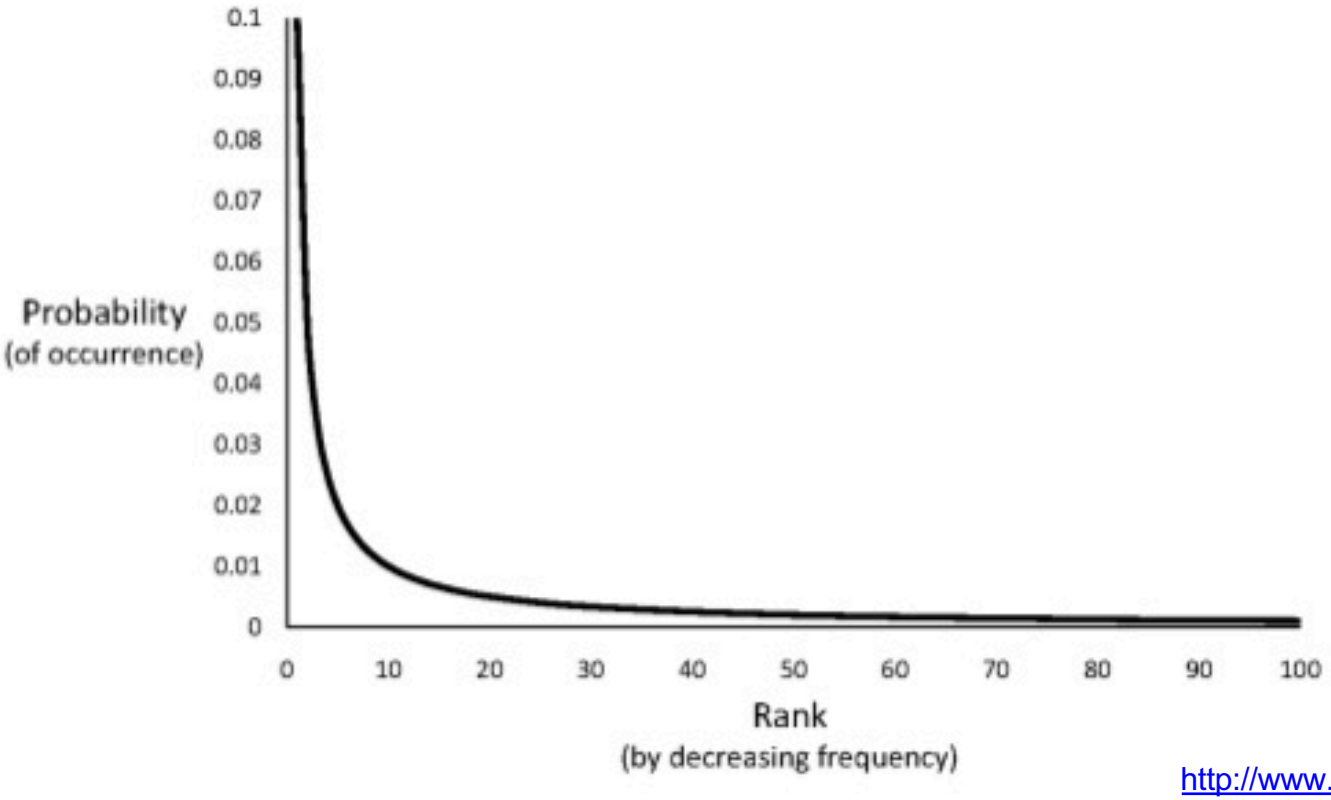
Empirical observations of numerous corpora found:

- Many low frequency words
- Only a few words with high frequency

TERM FREQUENCY AND ZIPF'S RULE

The empirical observations were formalized in the ***Zipf's rule***:

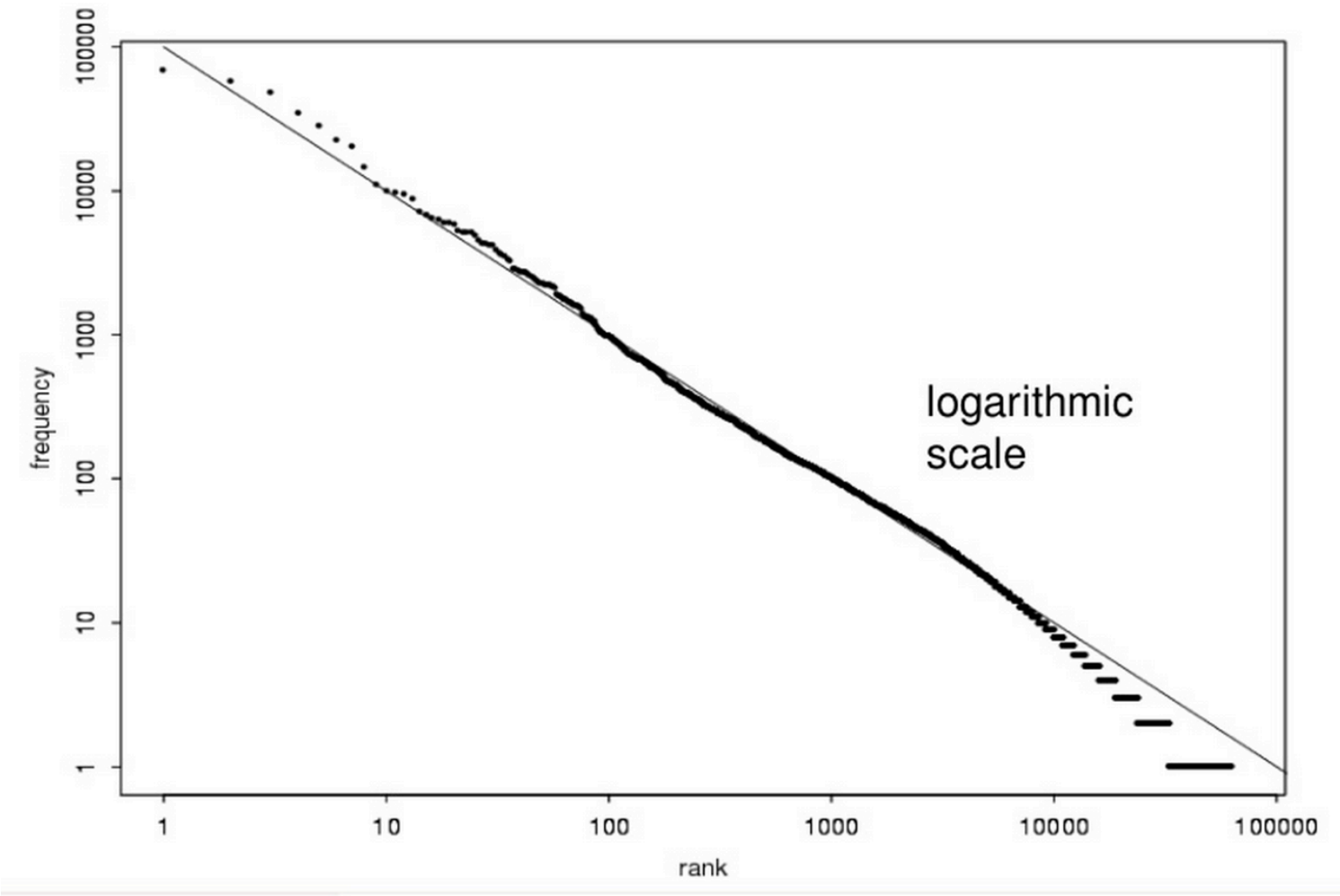
the frequency of a word in a given corpus is inversely proportional to its rank in the frequency table (for that corpus)



Word	Freq f	Rank r
the	3332	1
and	2972	2
a	1775	3
he	877	10
but	410	20
be	294	30
there	222	40
one	172	50
about	158	60
more	138	70
never	124	80
oh	116	90
two	104	100

Source:
http://www.slideshare.net/cowdung112/info-2402-irtchapter4

ILLUSTRATION OF THE ZIPF'S RULE



Word frequency in the Brown Corpus of American English text

source: http://nlp.stanford.edu/fsnlp/intro/fsnlp-slides-ch1.pdf

IMPLICATIONS OF THE ZIPF'S RULE

- Words in the upper part of the frequency table comprise a significant proportion of all the words in the corpus, but are semantically almost useless
  - Examples: the, a, an, we, do, to
- On the other hand, words towards the bottom of the frequency table are semantically rich, but are very rarely used and thus not representative of the corpus
  - Example: dextrosinistral, juxtapositional
- The rest of the words are those that represent the corpus the best and thus should be included in a model

STOP-WORDS

- An alternative or a complementary way to eliminating words that are (most probably) irrelevant for corpus analysis
- Stop-words are those words that (on their own) do not bear practically any information / meaning
- It is estimated that they represent 20-30% of words in any corpus
- There is no unique stop-words list
  - the stopwords R package offers easy access to frequently used lists
- Potential problems with stop-words removal:
  - the loss of the original meaning and text structure
  - examples: “this is not a good option” → “option”
    “to be or not to be” → null

POS TAGGING

Marking up words with Part of Speech (POS) tags, so that for each word, its POS (e.g., noun, pronoun, verb) is explicitly defined

An example:

"And now for something completely different"

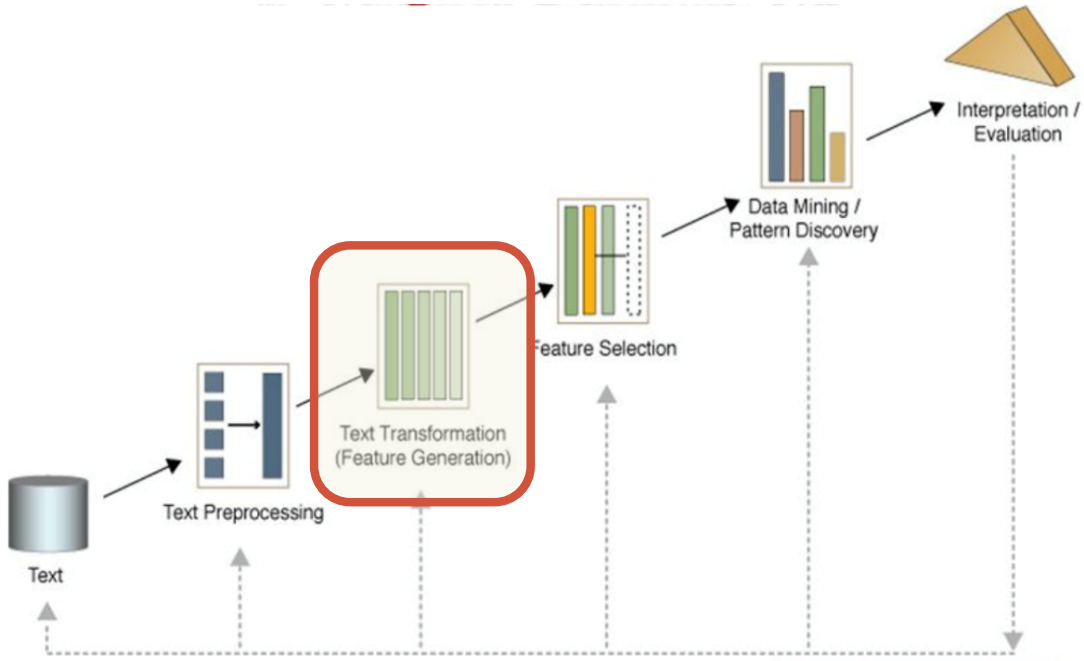
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'), ('completely', 'RB'), ('different', 'JJ')]

RB → Adverb; NN → Noun, singular or mass; IN → Preposition, …

The complete set of Penn Treebank POS tags, a standard POS tag set for English language, is available here

LEMMATIZATION AND STEMMING

- Two approaches to decreasing variability of words by reducing different forms of words to their basic / root form
- Stemming is a crude heuristic process that chops off the ends of words without considering linguistic features of the words
  - E.g., argue, argued, argues, arguing → argu
- Lemmatization relies on a vocabulary and morphological analysis of words, to return the base or dictionary form of a word, which is known as the lemma
  - E.g., argue, argued, argues, arguing → argue
    am, are, is → be



TEXT TRANSFORMATION (FEATURE CREATION)

TEXT TRANSFORMATION

It is the process of transforming unstructured textual content into a structured format that can be used as an input for:

- Statistical methods and techniques
  - E.g., topic modelling, latent semantic analysis
- Machine learning methods and techniques
  - E.g., classification, clustering
- Other pattern detection / information extraction methods
  - E.g., graph-based methods for keywords / topic extraction

Bag of Words (BoW) & Vector Space Model (VSM)

BAG OF WORDS MODEL

- Considers text as a simple set/bag of words
- Based on the two (unrealistic) assumptions:
  - words are mutually independent,
  - word order in text is irrelevant
- Despite these unrealistic assumptions and simplicity, it proved to be highly effective, and is often used in TM

BAG OF WORDS MODEL

- Unique words from the corpus are used for creating the corpus ‘dictionary’
- Then, each document from the corpus is represented as a vector of (dictionary) word frequencies

BAG OF WORDS MODEL

Example:

- Doc1: Text mining is to extract useful information from unstructured text.
- Doc2: Useful patterns are extracted from text.
- Doc3: Text mining is useful.

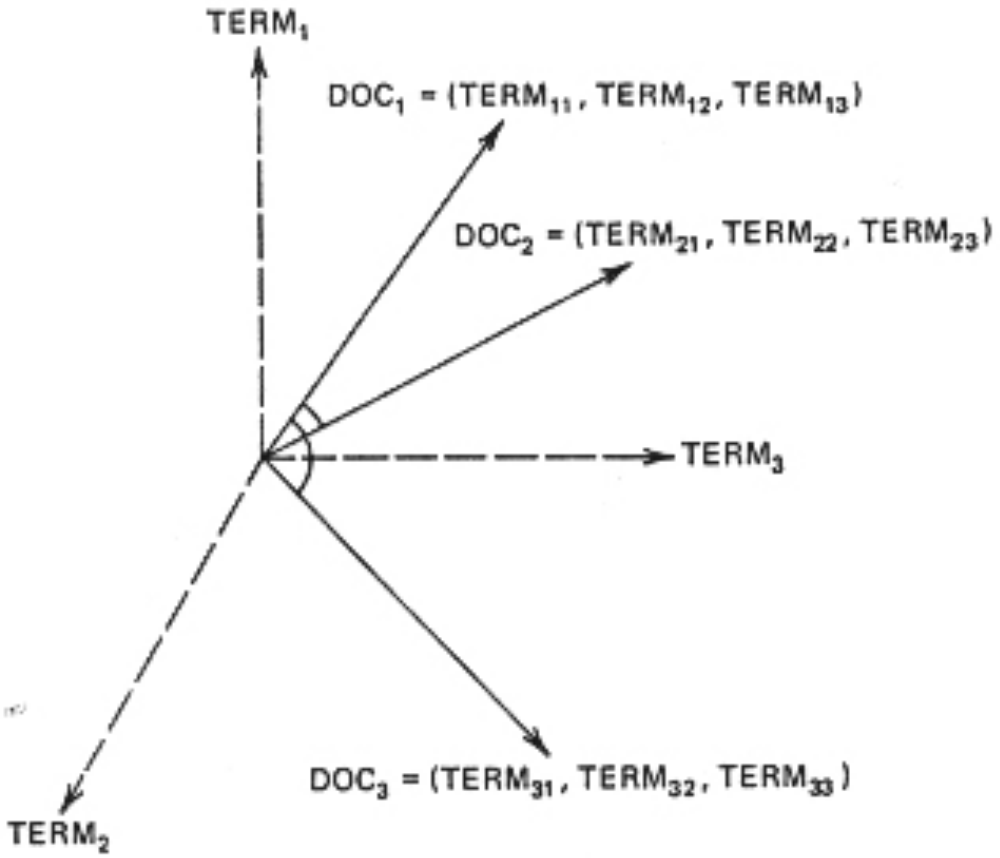
	text	mine	is	to	extract	use	inform	from	unstructur	are	pattern
Doc1	2	1	1	1	1	1	1	1	1	0	0
Doc2	1	0	0	0	1	1	0	1	0	1	1
Doc3	1	1	1	0	0	1	0	0	0	0	0

Note: words have been lower cased and stemmed

VECTOR SPACE MODEL

- Generalization of the Bag of Words model
- Documents from the corpus are represented as multi-dimensional vectors
  - Each unique term from the corpus represents one dimension of the vector space
  - The size of the vector space is determined by the number of unique terms in the corpus
  - *Term* can be a single word (unigram) or a sequence of words (bigrams, trigrams,…)
- Each term is associated with a weight that represents its relevance for a particular document

VECTOR SPACE MODEL (VSM)



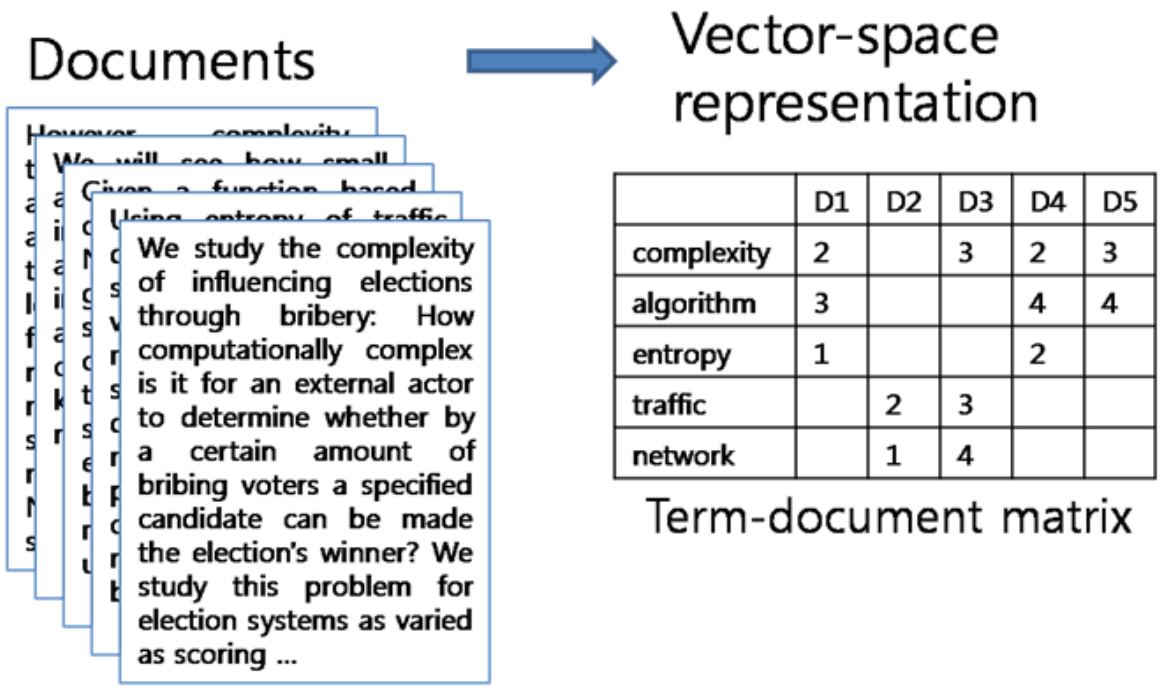
Documents that are similar, w.r.t. the terms they consist of, are placed closer (i.e., form a smaller angle) in the *n*-dimensional space of (the unique corpus) terms

Source:
http://www.cs.uni.edu/~okane/source/ISR/isr.html

VSM: Term Document Matrix

In VSM, corpus is represented in the form of *Term Document Matrix (TDM)*, an *m* x *n* matrix with following features:

- Rows (*i=1,m*) represent terms from the corpus
- Columns (*j=1,n*) represent documents from the corpus
- Cell *ij* stores the weight of the term *i* in the context of the document *j*



	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Image source:
http://mlg.postech.ac.kr/research/nmf

VSM: ESTIMATION OF TERM RELEVANCE

- There are various approaches for estimating term relevance, that is, for determining term weight
- Simple and frequently used approaches include:
  - Binary weights
  - Term Frequency (TF)
  - Inverse Document Frequency (IDF)
  - TF-IDF

Estimating term relevance: Binary weights

Weights take the value of 0 or 1, to reflect the presence (1) or absence (0) of a term in a particular document

Estimating term relevance: Term Frequency

- Term Frequency (TF) represents the number of occurrences of a term in a specific document
- The underlying assumption: the higher the term frequency in a document, the more important it is for that document

TF(t) = c(t,d)

c(t,d) - the number of occurrences of the term *t* in the document *d*

ESTIMATING TERM RELEVANCE: NORMALIZED TERM FREQUENCY

- TF values are often normalized in order to eliminate the effect of the document length
  - Words in longer documents tend to have higher TF values
- Typical approaches for normalizing TF values:
  - Divide each element of the TDM matrix (i.e. TF weight) with the norm of the corresponding word vector
  - Euclidean (L2) or Manhattan (L1) norm are often used

$$||x||_e = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Euclidean (L2) norm

$$||x||_1 = \sum_{i=1}^{n} |x_i|$$

Manhattan (L1) norm

Estimating term relevance: Inverse Document Frequency

- The underlying idea: assign higher weights to unusual terms, i.e., to terms that are not so common in the corpus
- IDF is computed at the corpus level, and thus describes corpus as a whole, not individual documents
- It is (often) computed in the following way:

$$IDF(t) = \log(\frac{N}{df(t)})$$

N – number of documents in the corpus

df(t) – number of documents with the term *t*

ESTIMATING TERM RELEVANCE: TF–IDF

- The underlying idea: value those terms that are not so common in the corpus (relatively high IDF), but still have reasonable frequency (relatively high TF)
- The most frequently used metric for computing term weights in a VSM
- There are different ways of combining TF and IDF metrics; the simplest and most often used one:

$$TF - IDF(t) = \mathrm{tf(t)} * \log(\frac{N}{df(t)})$$

ESTIMATING TERM RELEVANCE: TF–IDF

Alternative ways of computing TF-IDF weights

Term frequency	Document frequency	Normalization
n (natural): $\mathrm{tf}_{t,d}$	n (no): 1	n (none): 1
l (logarithm): $1+\log(\mathrm{tf}_{t,d})$	t (idf): $\log \frac{N}{df_t}$	c (cosine): $\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented): $0.5 + \frac{0.5\times \mathrm{tf}_{t,d}}{\max(\mathrm{tf}_{t,d})}$	p (prob idf): $\mathbf{max}\left(0, \log \frac{N-df_t}{df_t}\right)$	b (byte size): $1/CharLength^{\alpha}, \alpha < 1$
b (boolean): $\begin{cases} 1, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$		
L (log average): $\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$		

SMART notation - a mnemonic scheme for denoting TF-IDF weighting variants; e.g., one of the most often used TF-IDF variants is “ntc”

Image source: https://en.wikipedia.org/wiki/SMART_Information_Retrieval_System

VSM: Pros and cons

Advantages

- Intuitive
- Easy to implement
- Empirically proven as highly effective
- Particularly suitable for estimating document similarity, and, therefore, document clustering

VSM: Pros and cons

Drawbacks:

- Does not consider
  - words' position in the text,
  - word order,
  - words' co-occurrence in the overall corpus
- Large, sparse vectors; especially challenging in the case of very large corpora (w/ large vocabularies)
- Restricted to words and phrases (n-grams) as features

A GLANCE OVER OTHER OPTIONS FOR FEATURE CREATION

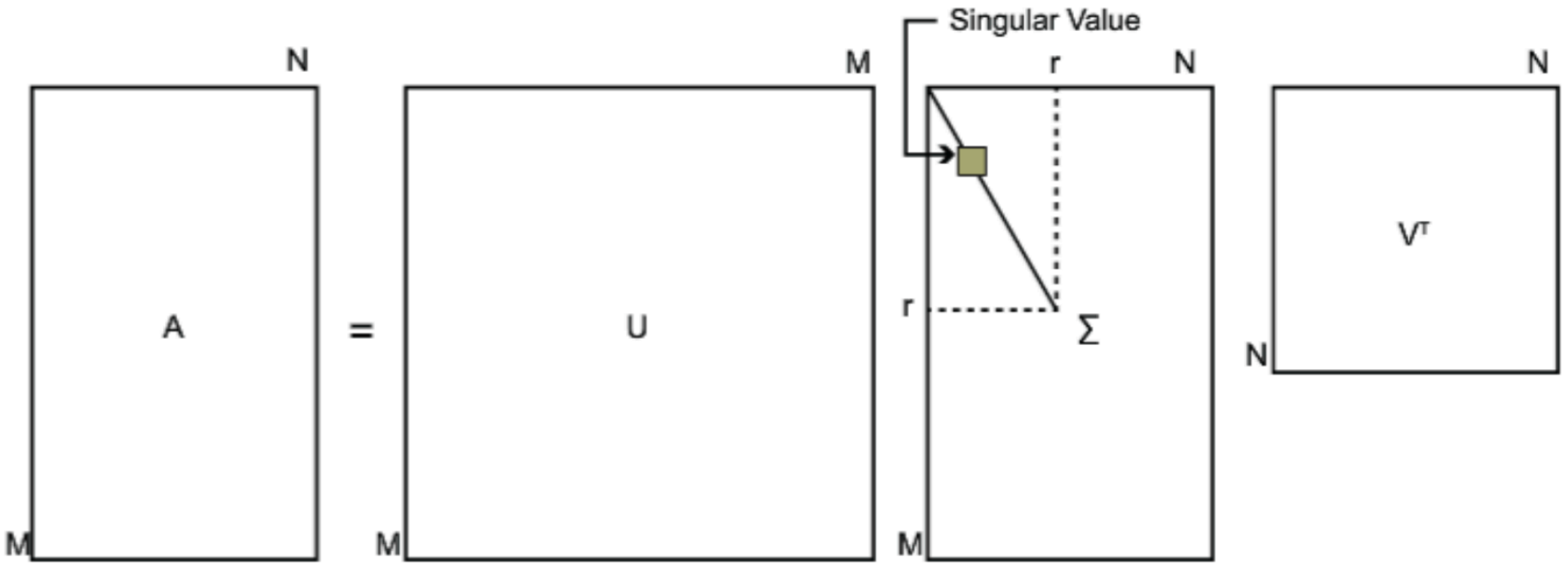
EXTENSIONS OF TRADITIONAL VSM

EXTENSIONS OF TRADITIONAL VSM

- Introduce additional kinds of features to VSM
- Instead of terms (n-grams), document vectors might include:
  - Dimensions obtained by reducing the feature (n-gram) space using a dimensionality reduction method such as *Singular Vector Decomposition* (SVD)
  - Entities detected in the text using an *entity linking* method
  - Topics detected in the text using a *topic modelling* method

SINGULAR VALUE DECOMPOSITION (SVD)

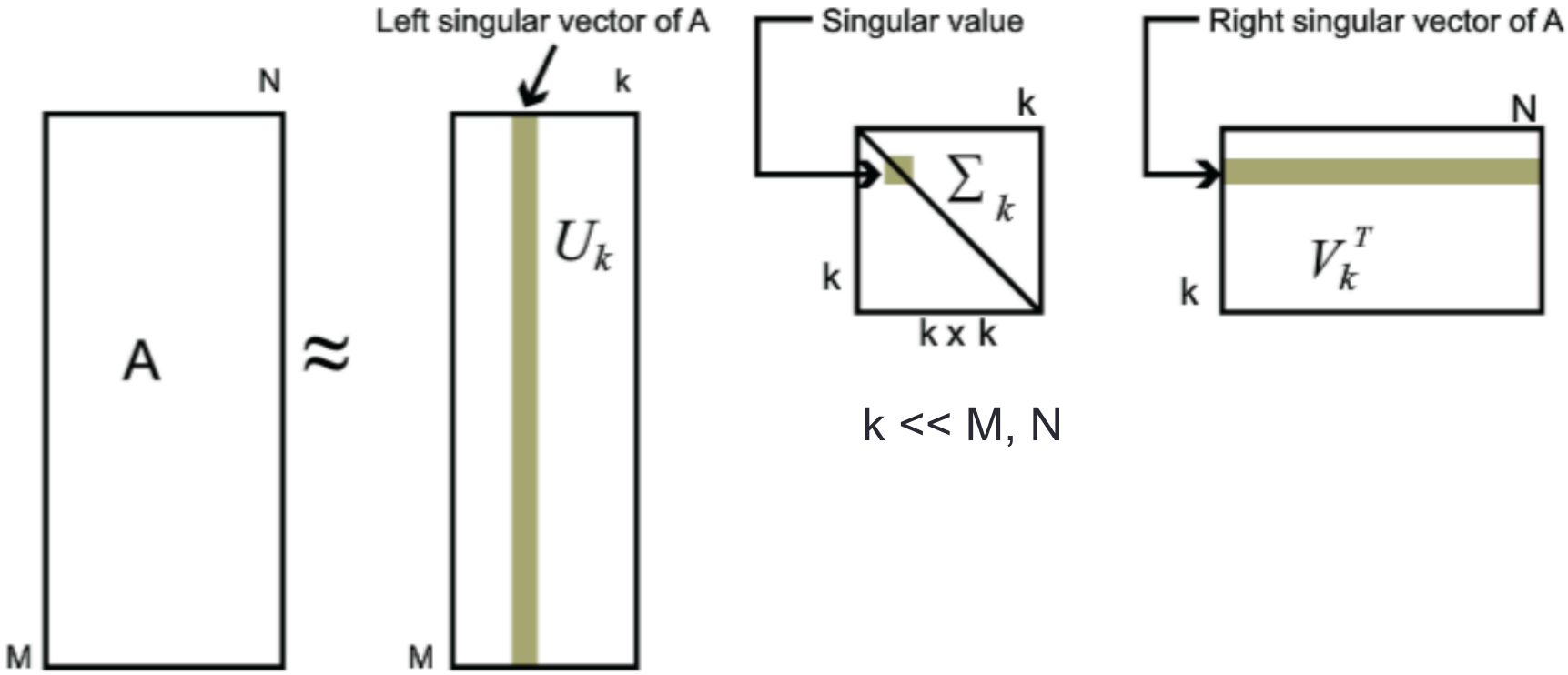
Matrix approximation through its rang reduction



In the context of a TM task,
A is the TDM matrix with M terms and N documents

Image source: https://rpubs.com/dsmilo/DATA643-Project3

SINGULAR VALUE DECOMPOSITION (SVD)



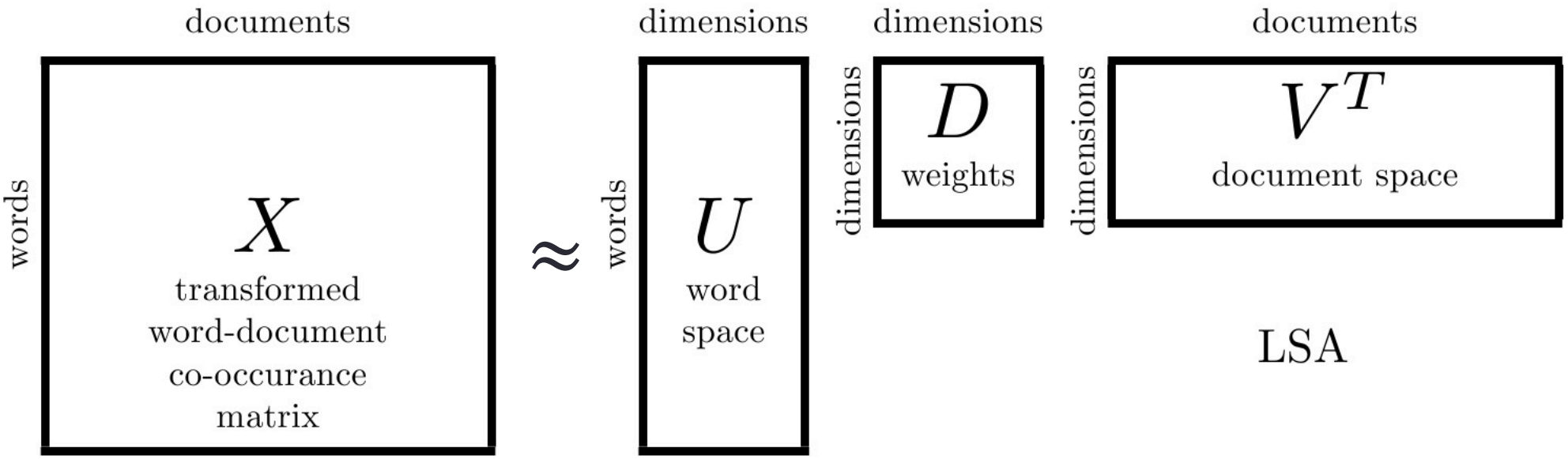
In the TM context,

- U_k (*left singular vector*) defines relations between terms and derived 'dimensions'
- V_k (*right singular vector*) defines relations between documents and 'dimensions'

Image source: https://rpubs.com/dsmilo/DATA643-Project3

LATENT SEMANTIC ANALYSIS (LSA)

LSA ≈ SVD applied to the TDM matrix to detect topics or concepts in the given corpus



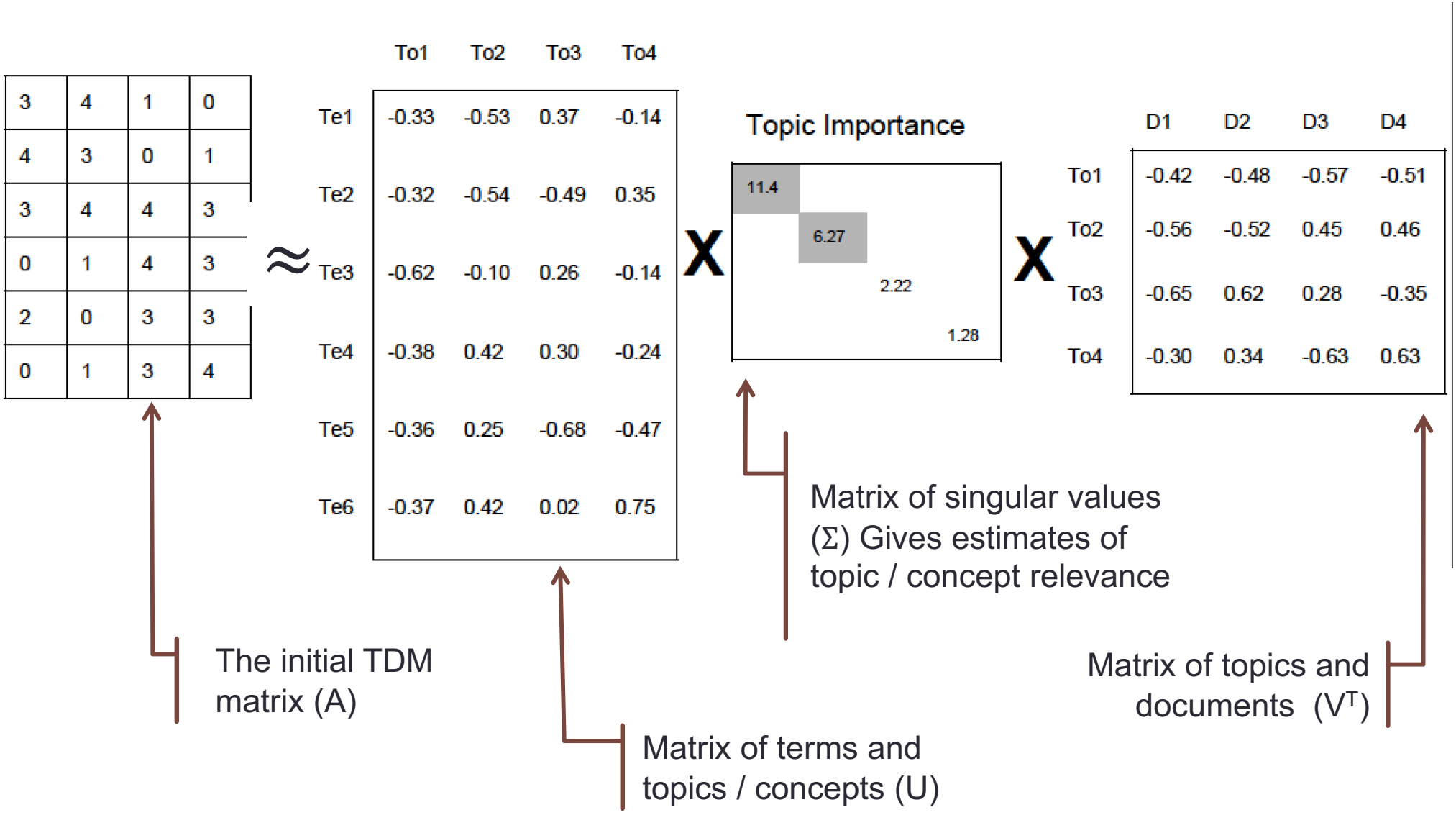
LATENT SEMANTIC ANALYSIS (LSA)

Example setup: let us suppose that we have extracted the following TDM from a tiny set of documents (D1-D4):

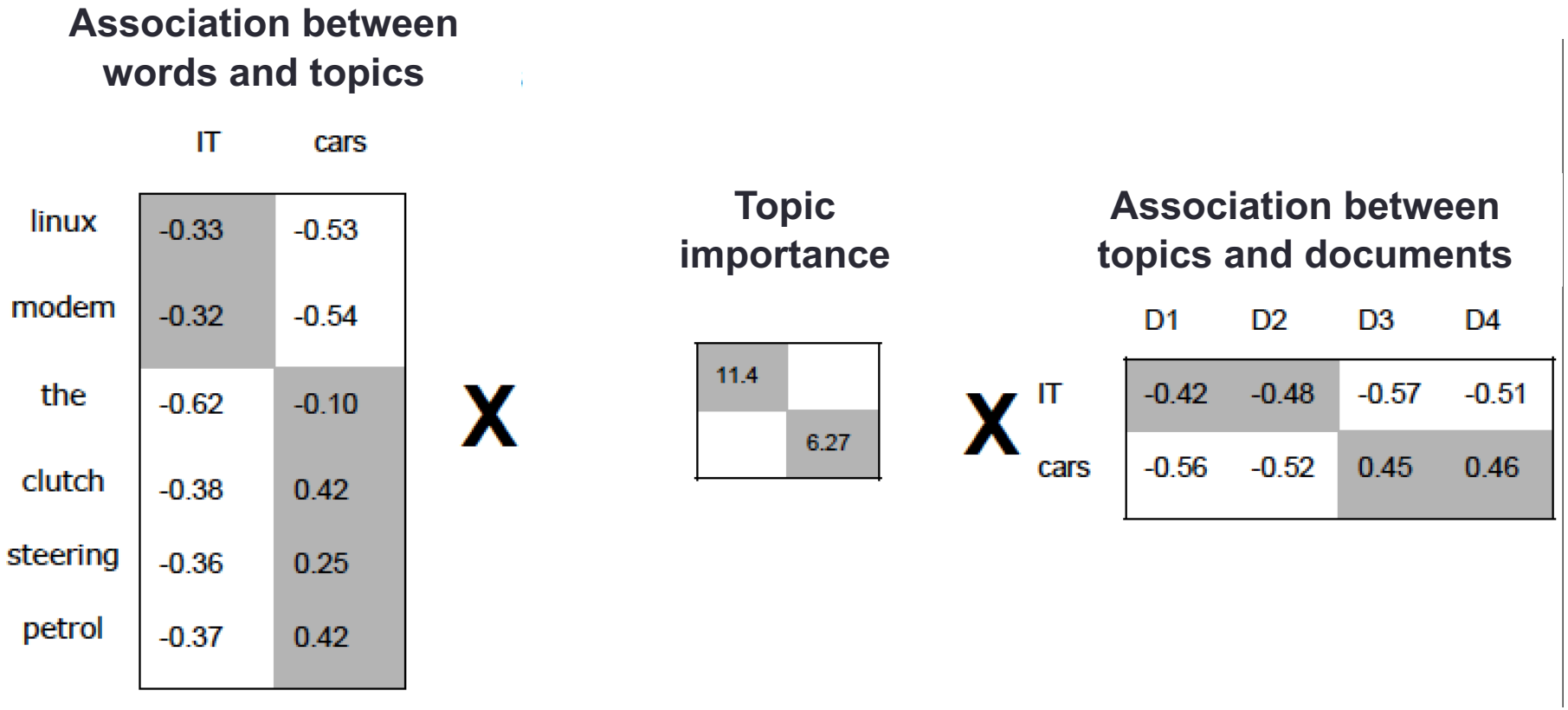
	D1	D2	D3	D4
linux	3	4	1	0
modem	4	3	0	1
the	3	4	4	3
clutch	0	1	4	3
steering	2	0	3	3
petrol	0	1	3	4

Example taken from: https://www.slideshare.net/vitomirkovanovic/topic-modeling-for-learning-analytics-researchers-lak15-tutorial

LATENT SEMANTIC ANALYSIS (LSA)



LATENT SEMANTIC ANALYSIS (LSA)



Removing concepts / themes with small singular values (noise), in order to capture more relevant concepts / themes (signal)

Named Entity Recognition (NER)

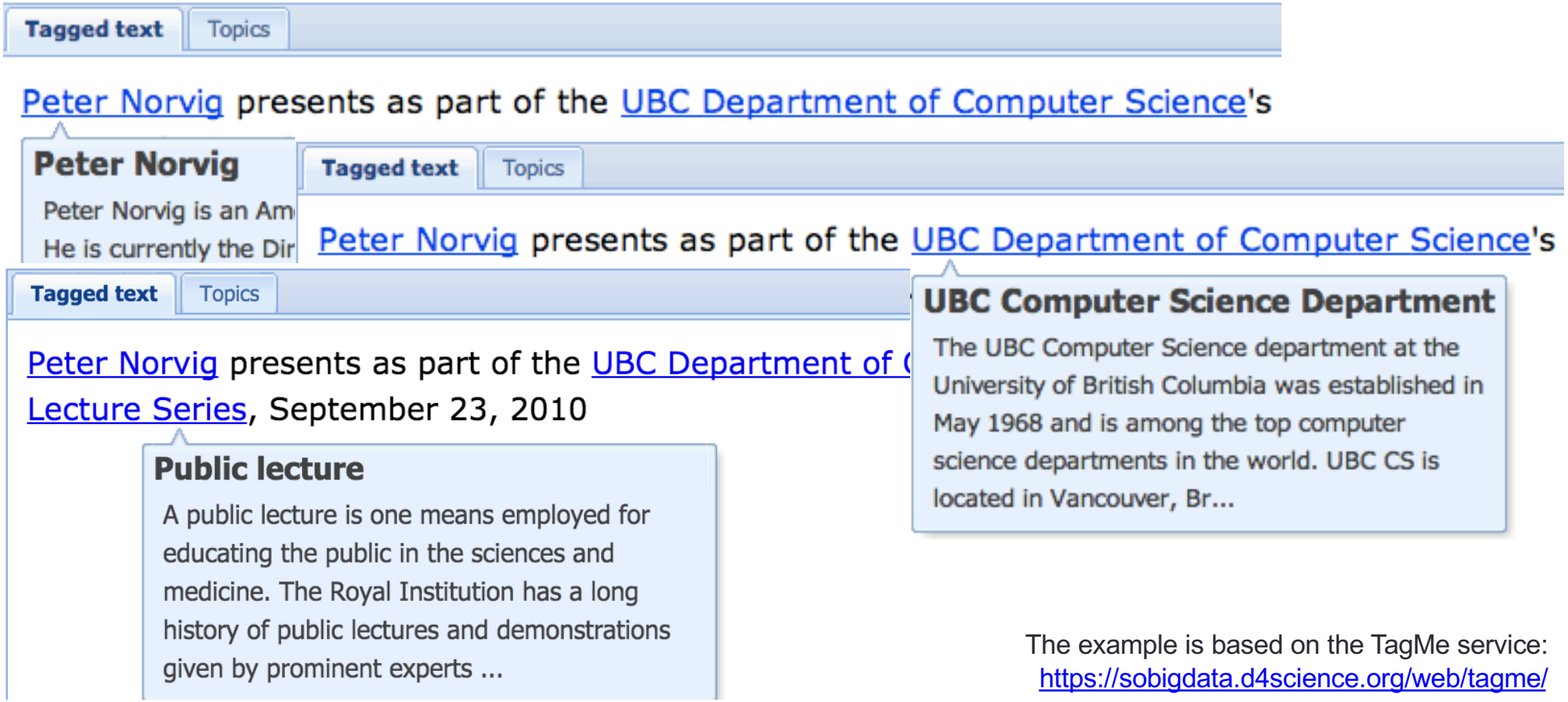
- Detection of entity mentions in the text and their association with the appropriate entity type
- Entities can be of different types: person, organisation, location, date, currency, and the like.
- Example:

> Peter Norvig presents as part of the UBC Department of Computer Science's Distinguished Lecture Series, September 23, 2010.

> Peter Norvig [PER] presents as part of the UBC Department of Computer Science's [ORG] Distinguished Lecture Series, September 23, 2010 [DATE].

ENTITY LINKING

- Entity linking = NER + Disambiguation
- Disambiguation = uniquely identifying an entity mention by linking it to an appropriate entity in a knowledge base (e.g. Wikipedia, Wikidata, DBpedia)



The example is based on the TagMe service:
https://sobigdata.d4science.org/web/tagme/

APPLYING ENTITY LINKING FOR FEATURE CREATION

- Entities detected in the text are used for building the corpus ‘dictionary’, which is then used for creating document vectors
- Text from the previous example would add the following entities to the corpus dictionary:
  - wikipedia:Peter_Norvig
  - wikipedia:Department_of_Computer_Science
  - wikipedia:Public_lecture

TOPIC MODELS

- Statistical models for mining topics of documents in the given corpus
- Topic is defined as a distribution of probabilities over the words of the given corpus
  - Each word ‘appears’ in each topic with a certain probability
- The modelling algorithm associates a topic distribution to each document in the corpus
  - Each document ‘belongs’ to each topic with a certain probability
- The number of topics has to be set in advance
  - It is an input parameter for the topic modelling algorithm

TOPIC MODELLING: ILLUSTRATION OF THE GENERATIVE MODEL

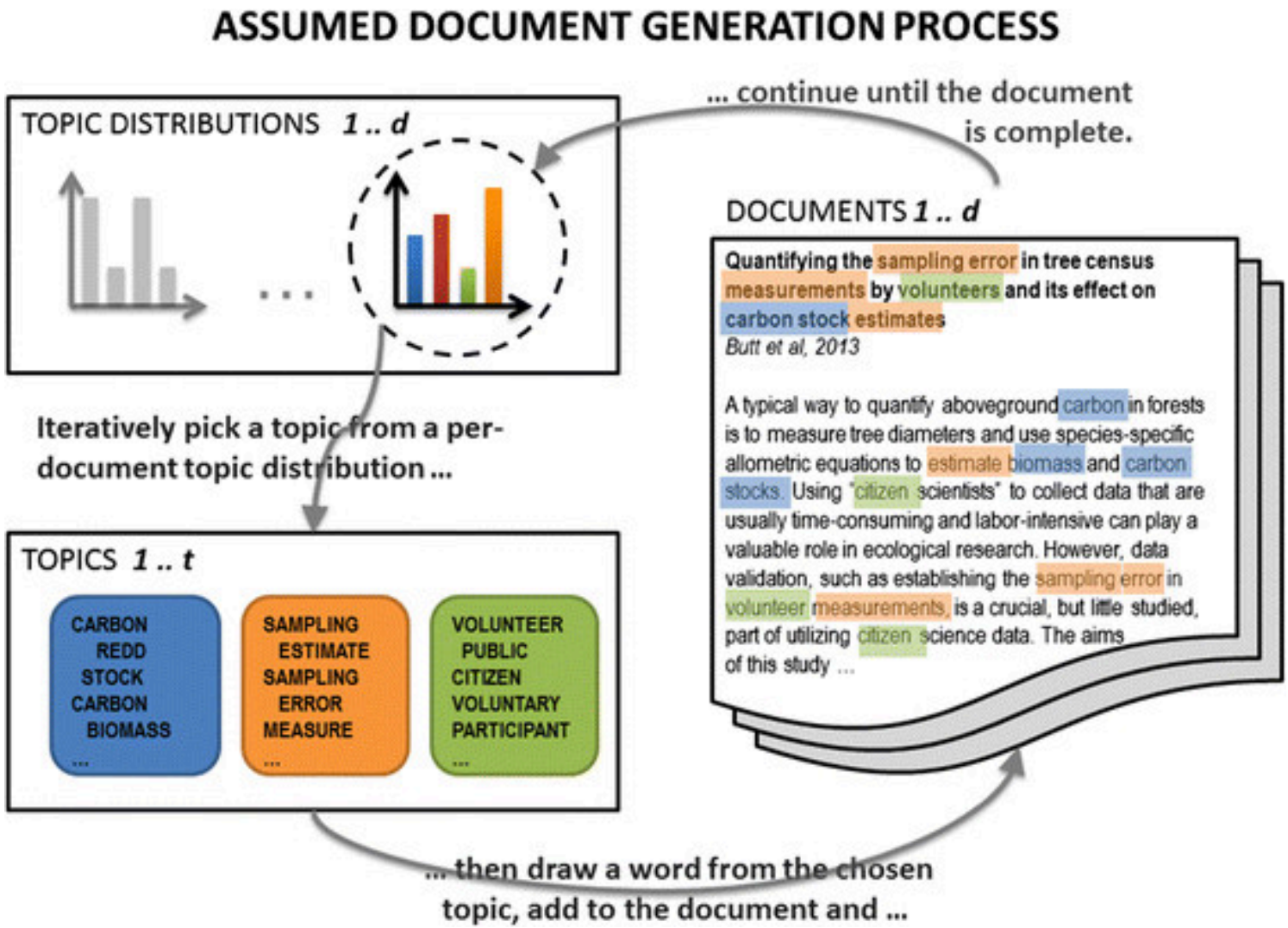
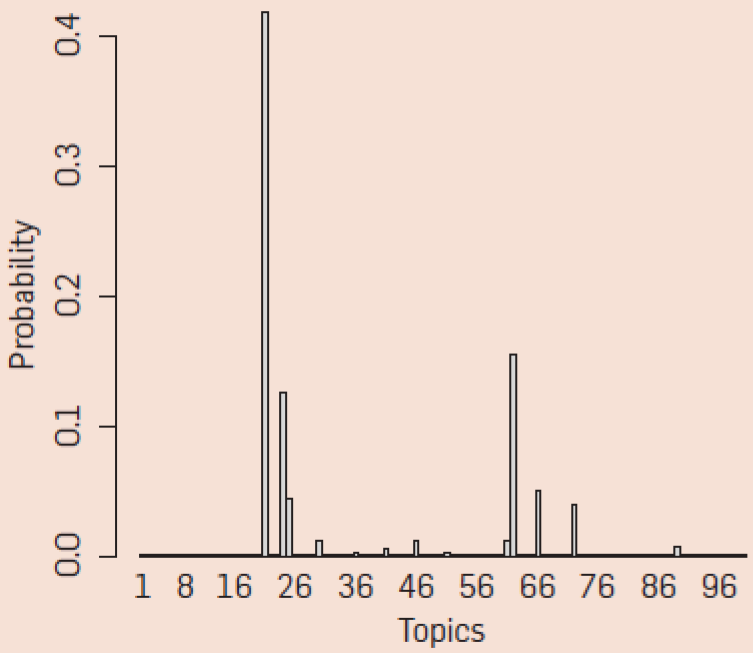


Image source:
https://www.researchgate.net/publication/264656298_Assessing_citizen_science_opportunities_in_forest_monitoring_using_probabilistic_topic_modelling

TOPIC MODELLING: ILLUSTRATION OF THE RESULTS



"Genetics"	"Evolution"	"Disease"	"Computers"
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

An illustration of the results obtained by fitting a 100-topic Latent Dirichlet Allocation (LDA) model to 17,000 articles from the journal Science

Source: Blei, David. 2012. Probabilistic topic models. Communications of the ACM 55(4):77–84.

APPLYING TOPIC MODELS FOR FEATURE CREATION

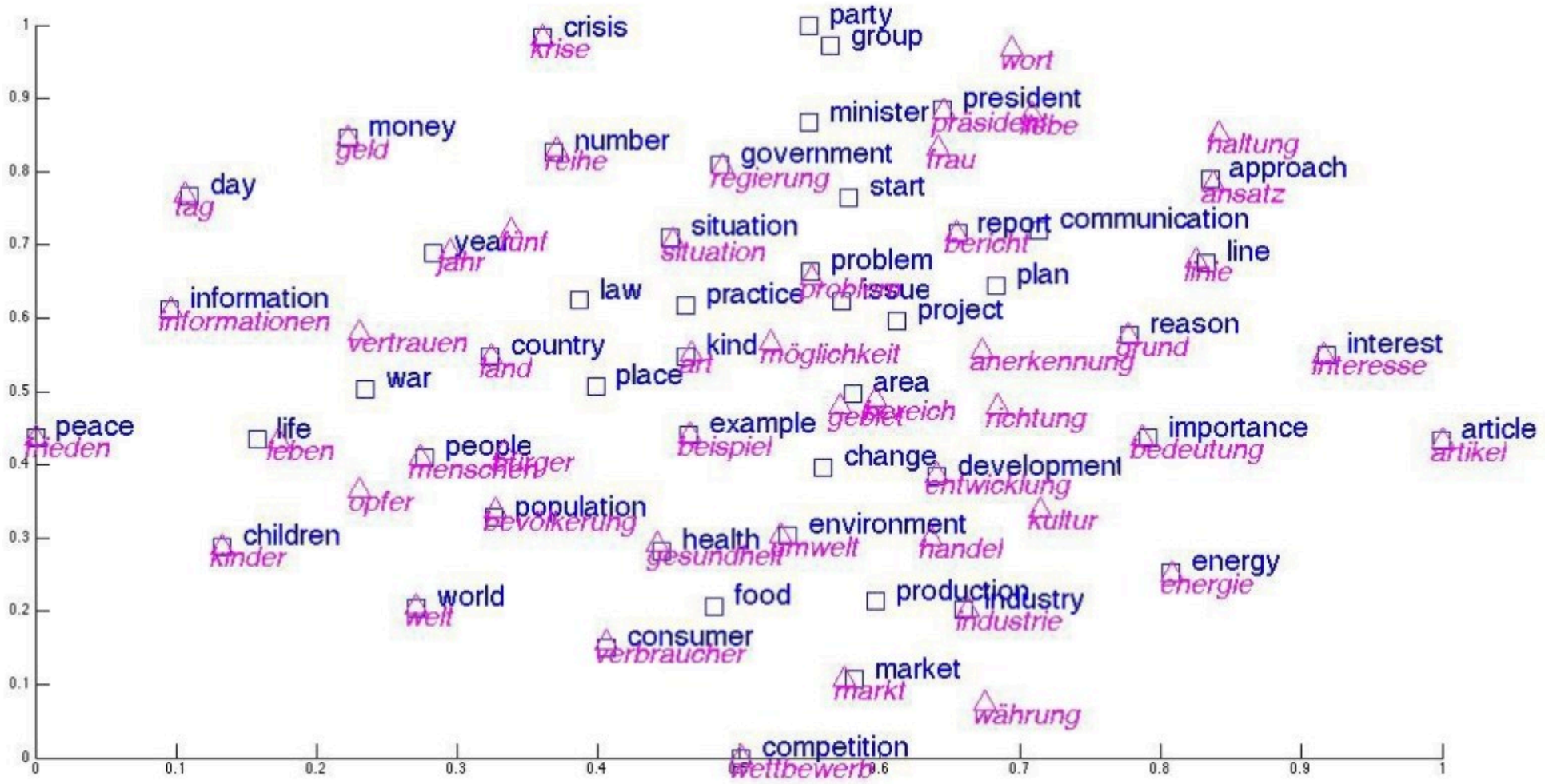
- Detected topics constitute the corpus ‘dictionary’, and thus serve as the basis for the creation of document vectors
- Each document vector is, in fact, a distribution of probabilities over topics for the given document
  - In other words, it is a vector of probabilities that the given document is on each of the topics from the ‘dictionary’

WORD EMBEDDINGS

WHAT IS WORD EMBEDDING?

- Currently, the most popular word representation model
- Vector representation of words, conceptually similar to the VSM we've examined
- Represents (embeds) words in a continuous vector space where semantically related words are mapped to nearby points
  - semantically related words are embedded near each other

WORD EMBEDDINGS: AN ILLUSTRATION



Most frequent German and English words from the WordSim353 datasets

Image source: https://nlp.stanford.edu/~lmthang/data/papers/naacl15_bivec.pdf

WHAT IS WORD EMBEDDING?

The overall idea behind word embeddings:

"*You shall know a word by the company it keeps*"

(John Rupert Firth)

- If two words happen to occur in similar contexts, they are likely to have similar meaning
- To put it differently, the meaning of a word can be captured, to some extent, by its use with other words
  - E.g., 'winter' and 'summer' are more likely to be used in the same context than the words 'winter' and 'chair'

WHAT IS WORD EMBEDDING?

To continue with the VSM analogy:

Word embedding is a kind of VSM where vectors…

… represent words

(not documents, as is the case of traditional VSMs)

… are dense and compact

(not sparse and large, as is the case of traditional VSMs)

… are relatively small - number of dimensions is much smaller than the number of items (words, queries, documents, …)

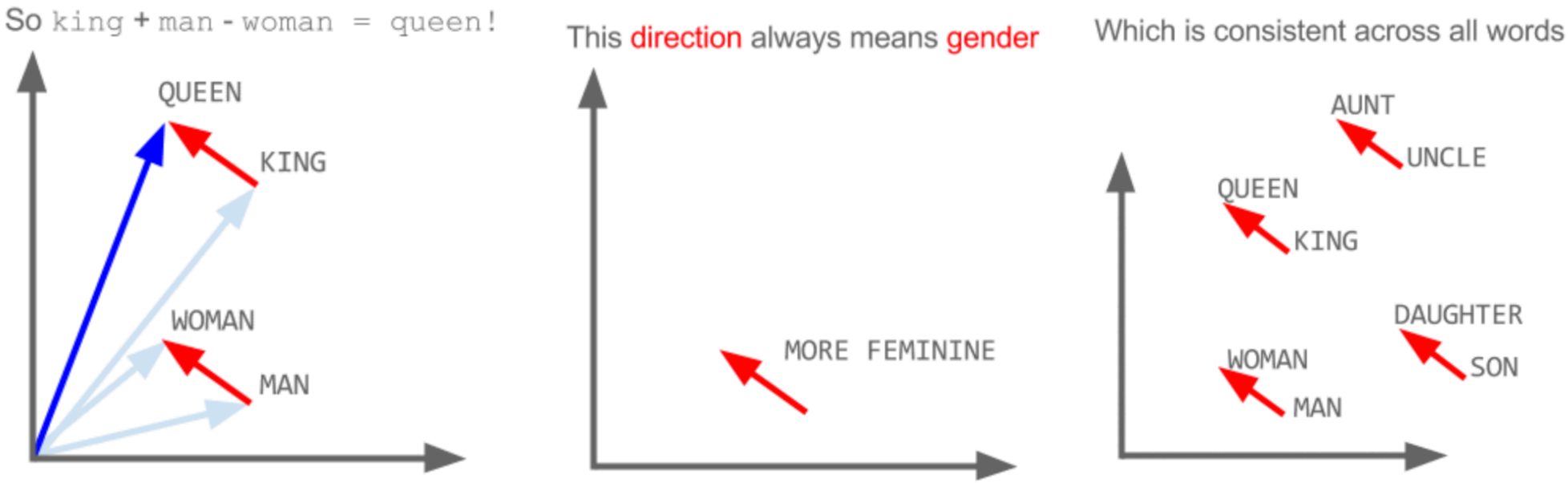
WHAT IS WORD EMBEDDING?

Well known for their remarkably good performance on word analogy tasks, such as

man is to *woman* as *king* is to _____?

good is to *best* as *smart* is to _____?

china is to *beijing* as *russia* is to _____?



See the whole animation: https://multithreaded.stitchfix.com/assets/images/blog/vectors.gif

MORE ON WORD VECTOR ARITHMETIC

Vector operations applied on word embeddings learned from a corpus of user comments posted on Web sites of 3 German newspapers

- Brexit – England + Griechenland / *Greece* =
  - Schuldenschnitt / *haircut* (0.36)
  - Grexit (0.34)
- Hitler + Putin =
  - Erdogan (0.57)
- König / *king* – Mann / *man* + Frau / *woman* =
  - Angela (0.59)

Source: https://www.youtube.com/watch?v=zFScws0mb7M

LEARNING WORD EMBEDDINGS

Highly demanding task, as it requires

- collecting a large-scale corpus
- pre-processing the corpus documents
- (initial) model building - a computationally demanding task
- examining the results
- tuning hyper-parameters - (another) computationally demanding and long process
- building the final model

PRE-TRAINED WORD EMBEDDING MODELS

An alternative is to use pre-trained word vectors

- Word2Vec: https://code.google.com/archive/p/word2vec/
- GloVe: http://nlp.stanford.edu/projects/glove/
- ConceptNet Numberbatch: https://github.com/commonsense/conceptnet-numberbatch
- A collection of pre-trained vectors based on different models and corpora

Note:

when deciding on a pretrained model, always consider the corpus it was built on and how well it matches your corpus

USING WORD EMBEDDINGS FOR TM TASKS

Simple methods for building a document representation:

- Average word vectors across all words in the given document
- Compute weighted average of word vectors
  - word vector for each word is weighted by the word's TF or TF-IDF value for the corresponding document
- Take min and max values of (weighted) word vectors and concatenate them

USING WORD EMBEDDINGS FOR TM TASKS

The listed simple methods

- result in a matrix with documents in rows and dimensions of the word vector model in columns
- such a matrix serves as the input for a TM algorithm

Pros and cons:

- Easy to apply, can be used with pre-trained word vectors
- Typically perform well (only) on short and topically coherent texts
- May lead to loss in precision in the captured meaning

USING WORD EMBEDDINGS FOR TM TASKS

More advanced methods include:

- Using *Word Mover Distance* (*WMD*) (Kusner et al., 2015) to compute similarity of documents
- Methods for learning vector representation for text of variable lengths (sentence, paragraph, document):
  - *Doc2Vec* (Le & Mikolov, 2014) - an extension of Word2Vec that uses paragraph vectors as part of the context during training
  - *Word Mover's Embedding* (Wu et al., 2018) – an unsupervised method that makes use of word embeddings, *WMD*, and *D2KE* (*Distances to Kernels and Embeddings*)

FURTHER OPTIONS FOR FEATURE CREATION

Further options for feature creation

There are numerous additional possibilities for feature creation

Important: when choosing features to include, always start from the objectives and particularities of the TM task at hand and the type of text in the corpus

Further options for feature creation

The following slides give examples of features that have been used in Learning Analytics research so far

Note: the examples are for illustration purposes only; they are by no means an exhaustive overview of the kinds of features that have been used in the field

Example 1: Automated detection of cognitive presence in forum posts (Kovanović et al., 2016)

- The task: classification of discussion forum posts into one of the four phases of cognitive presence
- Features:
  - LIWC (Linguistic Inquiry and Word Count) features
    - Counts of words that are indicative of different psychological processes (e.g., affective, cognitive, social, perceptual)
  - Coh-Metrix features
    - measures of text coherence, linguistic complexity, text readability, and lexical category use
  - Discussion context features
    - number of replies, message depth, cosine similarity to previous / next message, first / last (in the thread) indicator
  - Additional features: number of named entities; semantic cohesion of the message text (LSA-based sentence similarity)

Example 2: Automated scoring of open ended questions (Madnani et al., 2017)

- Task: predicting the score for an answer (of arbitrary-length) to open-ended question dealing with domain-specific concept(s)
- Features:
  - character n-grams (n=2-5)
    - to capture spelling and morphological variations
  - word n-grams (n=1,2)
    - to (approximately) capture the concepts mentioned in the answer
  - triples extracted from dependency parses
    - to (approximately) capture relationships between concepts
  - (answer) length bins
    - to (approximately) capture the level of detail in the answer

COMBINING VARIOUS KINDS OF FEATURES

Typical problem: I have (at least) two kinds of features that I want to use to build a classification model:

- text data that has been represented as a sparse bag of words,
- more traditional dense features
  - e.g., in the case of detecting relevant forum posts, that can be the length of a post, lexical diversity and/or cohesion of the post, reputation of the sender,...

Based on a discussion thread from the Data Science forum:
https://datascience.stackexchange.com/questions/987/text-categorization-combining-different-kind-of-features

COMBINING VARIOUS KINDS OF FEATURES

Potential solutions (in increasing order of 'goodness'):

- Add the few dense features to your sparse features matrix so that all the features are in a single sparse matrix, which is then used to train your model.
- Perform dimensionality reduction (such as SVD or PCA) on the sparse features to make them denser, and combine the features into a single dense matrix to train your model.
- Create a model using only sparse features and then combine its predictions (probabilities if it's classification), as a single dense feature, with your other dense features to create another (*stacked*) model.
  - Note: use only cross-validation predictions as features to train the 2nd (stacked) model, otherwise you'll risk overfitting.

Based on a discussion thread from the Data Science forum:
https://datascience.stackexchange.com/questions/987/text-categorization-combining-different-kind-of-features

EXAMPLE: COMBINING VARIOUS KINDS OF FEATURES FOR AUTOMATED SHORT ANSWER SCORING

Two approaches to short answer scoring:

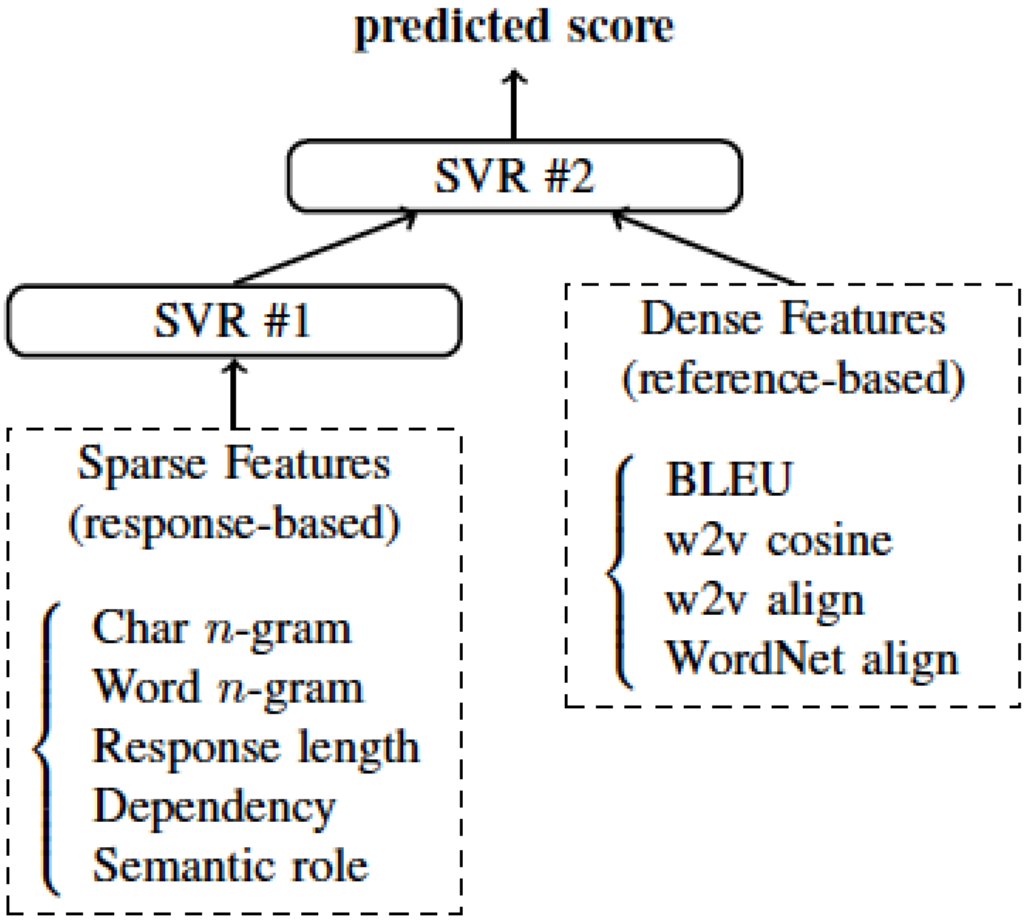
a) Response based – extracts large number of linguistic features from student responses (e.g., word and character n-grams)

b) Reference based – computes similarity scores between student responses and reference texts (e.g., exemplars for each score level)
   - E.g. cosine similarity between averaged word2vec representation of content words in the response and reference texts

The challenge: how to combine a large set of sparse features (a) and a small set of dense continuous features (b)?

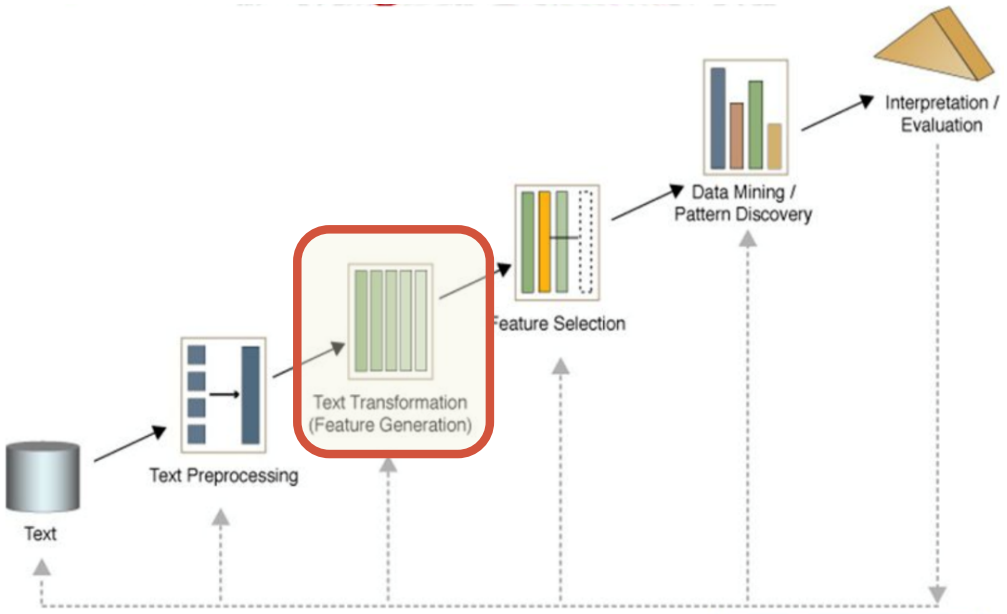
EXAMPLE: COMBINING VARIOUS KINDS OF FEATURES FOR AUTOMATED SHORT ANSWER SCORING

The stacking model presented by Sakaguchi et al. (2015)



"…we train a lower-layer model to aggregate the sparse response-based features into a single "response-based prediction" feature, and then train an upper-layer SVR model that includes that feature along with all of the reference-based features."

SVR = Singular Vector Regression



FEATURE SELECTION

FEATURE SELECTION

- Numerous methods are available
- Primarily aimed for the selection of linguistic features, that is, terms (n-grams) from the corpus
- Typically, feature ranking (filtering) methods are used; common to all of them:
  - assume feature independence,
  - assign scores to the features,
  - select features with high scores

FEATURE SELECTION: RANKING (FILTERING) METHODS

Frequently used measures for feature ranking:

- Document frequency and collection (total term) frequency
- Information gain (for supervised TM tasks only)
- Mutual information (for supervised TM tasks only)
- Chi-square (for supervised TM tasks only)

Feature selection
Information Gain (IG)

IG of the term *t* measures the reduction of entropy (*E*), that is, uncertainty of prediction, that can be achieved by knowing if the term *t* is in the given document or not

$$IG(t) = E(c) - (P_r(t)E(c|t) + P_r(\bar{t})E(c|\bar{t}))$$

$P_r(t)$ - probability of term t occurring in the document

$P_r(\bar{t})$ - probability of term t *not* occurring in the document

$E(c)$ - entropy of class c

$E(c|t)$ - entropy of class c given that term t is in the document

FEATURE SELECTION
MUTUAL INFORMATION (MI)

MI measures how much information term ***t*** contains about class ***c***

- If the distribution of the term ***t*** in the class ***c*** is the same as in the whole corpus, MI = 0
- If the term ***t*** is present in a document if and only if the document is in the class ***c***, MI reaches its maximum

$$MI(t,c) = log\frac{P_r(t \wedge c)}{P_r(t) \times P_r(c)}$$

Or

$$MI(t,c) = logP_r(t|c) - logP_r(t)$$

Drawback: it is biased towards words with low frequency of occurrence in the given corpus

FEATURE SELECTION
CHI SQUARE (CHI)

- Tests the (in)dependence of term ***t*** and class ***c***
- High scores indicate that the occurrence of the term *t* and class *c* are dependent, and the term should be selected as a feature
- The computation is based on the two-way contingency table of term ***t*** and class ***c***:

	c	$\bar{c}$
t	A	B
$\bar{t}$	C	D

$$CHI(t,c) = \frac{N \times (A \times D - C \times B)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)}$$

Drawback: not reliable for low frequency terms (as table cells of such terms would be sparingly populated)

FEATURE SELECTION: FEATURE RANKING (FR) METHODS

Advantages:

- simple and computationally efficient

Drawbacks:

- Do not take the associations of features (words) into account
  → the selected features, while individually relevant, when taken together, may turn to be redundant and thus can deteriorate a classifier's performance

FEATURE SELECTION: FEATURE SUBSET SELECTION (FSS) METHODS

An alternative to feature ranking methods

- E.g., selection through recursive feature elimination

Advantages:

- consider relations between features

Drawbacks:

- computationally expensive for high-dimensional text data
- the result is often tied to a particular classification algorithm / model

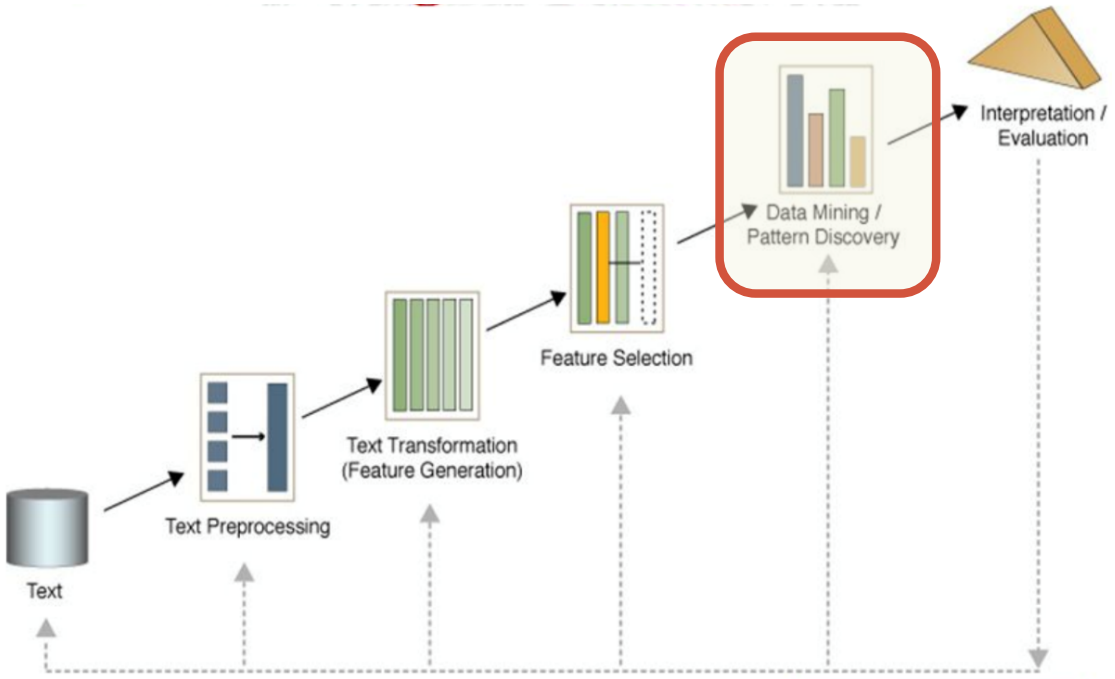
FEATURE SELECTION: A MIXED-METHOD APPROACH

- To combine the advantages of FR and FSS methods, a two-stage approach is recommended:
  1) Employ a FR method to do the initial reduction of the feature set
  2) Follow up with a FSS method to eliminate redundant features
- See e.g. (Javed et al., 2015)

FEATURE SELECTION

Comprehensive experimental comparisons of feature selection techniques for:

- Text classification: Yang & Pedersen, 1997; Forman, 2003
- Text clustering: Liu et al., 2003



DATA MINING / PATTERN DISCOVERY

DATA MINING, PATTERN DETECTION

Structured representation of textual data, such as DTM, can be used as the input for various machine learning algorithms

- Classification algorithms for
  - Sentiment analysis
  - Categorizing documents into a set of pre-defined categories / labels
  - Named entity recognition and entity linking
  - …
- Clustering algorithms for
  - Detecting topics / themes of documents in a corpus
  - Topic-wise grouping of documents in a corpus
  - …

BINARY CLASSIFICATION EXAMPLE: OPINION MINING (SENTIMENT ANALYSIS)

- Corpus: movie reviews from the Large Movie Review Dataset; labels (classes) were assigned based on the movie rating (1-4: negative, 7-10: positive)
- Features: unigrams, bigrams, trigrams, and their various combinations
- ML method: logistical regression
  - for each observation, the method predicts a value in the [0,1] range; value > 0.5 is considered positive

Stakić, M. (2016). Sentiment analysis in movie reviews. BSc thesis, University of Belgrade, Belgrade.

BINARY CLASSIFICATION EXAMPLE: OPINION MINING (SENTIMENT ANALYSIS)

Illustration of the results:

It's the movie equivalent of that rare sort of novel where you find yourself checking to see how many pages are left and hoping there are more, not fewer. (film Pulp Fiction, 1994)

Value predicted by the classifier: **0.9516**

As someone who's watched more bad movies than you can imagine, I'm mostly immune to the so-bad-it's-good aesthetic, though I can see how, viewed in a theater at midnight after a few drinks, this might conjure up its own hilariously demented reality. (film The Room, 2003)

Value predicted by the classifier: **0.0111**

MULTICLASS CLASSIFICATION EXAMPLE: AUTOMATED ANALYSIS OF STUDENTS' SELF-REFLECTIONS

- Presented in (Kovanović et al., 2018)
- Corpus:
  - students' self-reflections on video recordings of their own musical performances, collected from 4 undergrad. courses
  - sentence segments were manually labelled as Observation, Motive, or Goal
- Features:
  - top 100 unigrams, bigrams, and trigrams
  - Linguistic Inquiry and Word Count (LIWC) features
  - Coh-Metrix features
  - Context feature (e.g. first_in_sentence)
- ML method: Random forest

AUTOMATED ANALYSIS OF STUDENTS' SELF-REFLECTIONS: TOP 15 FEATURES

Feature	Description
liwc.see	Perceptual processes: seeing (e.g., view, saw, seen)
cm.SMCAUSr	Situation model: ratio of casual particles to causal verbs
cm.DRPVAL	Syntactic pattern density: agentless passive voice density, incidence
liwc.focuspast	Time orientation: focus towards past (e.g., ago, did, talked)
cm.WRDNOUN	Word information: noun incidence
liwc.ingest	Biological processes: ingestion (e.g., dish, eat, pizza)
cm.CNCCaus	Connectives: causal connectives, incidence
trust.ensemble	Frequency of "trust ensemble" bigram
cm.SMINTEr	Intentional cohesion: ratio of intentional particles to intentional actions/events
liwc.Period	Punctuation: use of full stop
cm.DRNP	Syntactic pattern density: incidence score of noun phrases
chamber.music	Frequency of "chamber music" bigram
liwc.AllPunc	Punctuation: all (e.g., periods, commas, question marks)
liwc.cause	Cognitive processes: causality (e.g., because, effect)
cm.SMCAUSv	Situational model: incidence score of causal verbs

Feature importance is estimated using the *Mean Decrease in Gini index (MDG)*.

MDG assesses how useful a feature is for separating data items among different classes

CLUSTERING EXAMPLE: THE KEYGRAPH METHOD

- Topic detection and topic-wise document clustering, proposed in (Sayyadi & Raschid, 2013)
- Key facts about KeyGraph method
  - Transforms a document corpus into a keywords co-occurrence graph
  - Uses an off-the shelf community detection algorithm to group highly co-occurring keywords into “communities” (clusters)
  - The detected keyword communities tend to be good proxies for corpus topics, and allow for topical grouping of documents

KEYGRAPH: ILLUSTRATION OF THE RESULTS

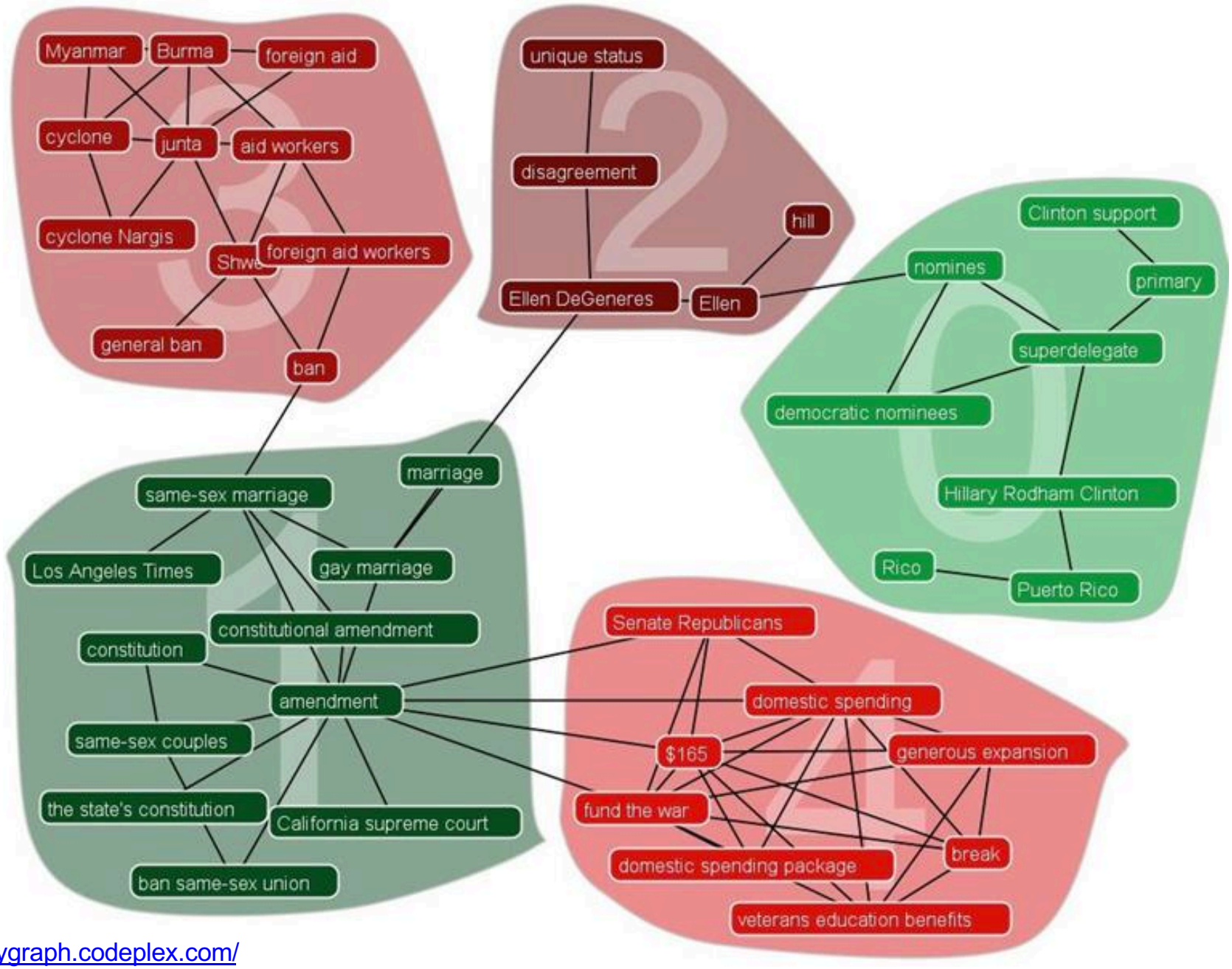


Image source: http://keygraph.codeplex.com/

AN EXAMPLE OF A COMBINED USE OF DIFFERENT TM METHODS: SUMMARIZATION OF GROUP CHAT MESSAGES

Presented in (Stanojević, 2017)

Objective:

- generate a summary of a discussion thread by grouping and summarizing topically related posts or post segments

Method:

- Text representation using the Vector Space Model, with TF-IDF metric for estimation of term relevance
- Computation of message similarity using cosine similarity metric
- Clustering of messages, based on the estimated similarity, using the Affinity Propagation algorithm
- Summarization of each message cluster using the TextRank algorithm

An example of a combined use of different TM methods: Topic Analysis of Discourse in a MOOC

Presented in (Joksimovic et al., 2015)

Objectives:

- identify main themes / topics - as groups of concepts - emerging from learners' posts on social media used in a MOOC
- analyse how the identified themes evolved over the course and if they might have been influenced by the recommended readings

An example of a combined use of different TM methods: Topic Analysis of Discourse in a MOOC

Method:

- Semantic annotation of social media posts and recommended readings
  - detecting concept mentions in the text and associating them with the corresponding concepts from a knowledge base (Wikipedia)
- Creation of concept co-occurrence graphs
  - undirected weighted graphs for each week of the course and each media
- Clustering of concepts into topics/themes
  - a community detection algorithm applied to the largest connected component in each graph
- Computation of graph metrics to get an insight into detected themes
  - graph and cluster connectedness measures to estimate topical coherence
  - centrality measures to estimate the importance of individual concepts

An example of a combined use of different TM methods: Topic Analysis of Discourse in a MOOC

Method (cont.):

- Computing similarity of posts in each two consecutive weeks
  - representing posts as vectors of detected concepts and computing cosine similarity of those vectors
- Computing similarity of posts and course materials
  - cosine similarity between vector representation of posts and vector representation of recommended readings
  - the readings recommended in week *k* were compared to posts in each succeeding week (*k+1, k+2*,…)

RELEVANT / RECOMMENDED LINKS

R PACKAGES FOR TEXT ANALYSIS

- quanteda
- tidytext
- udpipe
- text2vec
- wordVectors
- fastText
- a complete list of CRAN packages for natural language processing

Note: in addition to the packages focused on text analysis, one needs to use many other R packages to implement a complete TM workflow

RECOMMENDATIONS

Books

- J. Silge & D. Robinson. *Text Mining with R – A Tidy Approach*. O'Reilly, 2017. E-book publicly available at: http://tidytextmining.com/
- G.S. Ingersoll, T.S. Morton, A.L. Farris. *Taming Text*. Manning Pub., 2013. (with examples in Java)
- S. Bird, E. Klein, & E. Loper. *Natural Language Processing with Python.* O'Reilly, 2009. E-book publicly available at: http://www.nltk.org/book/

RECOMMENDATIONS

Online courses and tutorials

- Introduction to Text Analytics with R
  - a series of YouTube videos that will gradually, through a practical example, walk you through all the phases of the TM process
- Text as Data course
  - provides “an overview of popular techniques for collecting, processing, and analyzing text-based data”
- Text Mining @ LASI’18
  - materials (documented R scripts and slides) from the TM tutorial and workshop held at last year’s LASI

RECOMMENDATIONS

A few introductory articles on word embeddings in R

- Getting Started with Word Embeddings in R
  - a tutorial for building word embedding text representation using the wordVectors R package
- Word Embeddings and Document Vectors: Part 2. Classification
  - comparison of classifiers based on traditional document vectors vs word vectors; examines different classification algorithms, different word embeddings, and different options for text pre-processing
- An analysis of tweets from followers of the Austrian alt-right movement
  - an interesting example that shows how to use a GloVe model and text2vec R package to find semantically related words; also illustrates how to collect data from Twitter using the twitteR package

RECOMMENDATIONS

Various text analytics resources

- WordNet:
  - R wordnet
  - A nice example of using WordNet, plus some additional text preprocessing resources
- Regular expressions tutorial
- Word embeddings
  - Word2vec: paper, pre-trained models
  - GloVe (Global Vectors for Word Representation): paper, pre-trained models
  - NLPL word embeddings repository

RECOMMENDATIONS

Public (open) data sources

- Project Gutenberg
  - GutenbergR – R package for easy access to data from the Gutenberg collection
- NLP datasets - Large collection of data sets (corpora) for NLP / Text Mining research
- Data for Everyone - not all data sets are text based, but there are such
- Kaggle datasets with the “linguistic” tag – some of these will be speech data, but majority are text-based

REFERENCES

- Forman, G. (2003). An Extensive Empirical Study of Feature Selection Metrics for Text Classification. J. Mach. Learn. Res., 3, 1289–1305. http://www.jmlr.org/papers/volume3/forman03a/forman03a_full.pdf
- Javed, K., Maruf, S., & Babri, H. A. (2015). A two-stage Markov blanket based feature selection algorithm for text classification. *Neurocomputing*, *157*, 91–104. https://doi.org/10.1016/j.neucom.2015.01.031
- Joksimović, S., Kovanović, V., Jovanović, J., Zouaq, A., Gašević, D., & Hatala, M. (2015). What Do cMOOC Participants Talk About in Social Media?: A Topic Analysis of Discourse in a cMOOC. In *Proc. of the 5th Int'l Conf. on Learning Analytics And Knowledge* (pp. 156–165). New York, NY, USA: ACM. DOI: 10.1145/2723576.2723609
- Kovanović et al (2016). Towards Automated Content Analysis of Discussion Transcripts: A Cognitive Presence Case. In *Proc. of the Sixth Int'l Conf. on Learning Analytics & Knowledge* (pp. 15–24). DOI: 10.1145/2883851.2883950
- Kovanović et al. (2018). Understand Students' Self-reflections Through Learning Analytics. In *Proc. of the 8th Int'l Conf. on Learning Analytics and Knowledge* (pp. 389–398). New York, NY, USA: ACM. DOI: 10.1145/3170358.3170374
- Kusner, M. J., Sun, Y., Kolkin, N. I., & Weinberger, K. Q. (2015). From Word Embeddings to Document Distances. In Proc. of the 32nd Int'l Conf. on Machine Learning - Vol. 37 (pp. 957–966). http://proceedings.mlr.press/v37/kusnerb15.pdf
- Le, Q., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents. In Proc. of the 31st Int'l Conf. on Machine Learning - Vol. 32 (pp. 1188 – 1196). https://arxiv.org/abs/1405.4053

REFERENCES

- Liu, T., Liu, S., Chen, Z., & Ma, W.-Y. (2003). An Evaluation on Feature Selection for Text Clustering. In Proc. of the 20th Int'l Conf. on Machine Learning (pp. 488–495). Washington, DC, USA: AAAI Press. Retrieved from http://dl.acm.org/citation.cfm?id=3041838.3041900
- Madnani, N., Loukina, A., & Cahill, A. (2017). A Large Scale Quantitative Exploration of Modeling Strategies for Content Scoring. *Proc. of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, 457–467. DOI: 10.18653/v1/W17-5052
- Sakaguchi, K., Heilman, M., & Madnani, N. (2015). Effective Feature Integration for Automated Short Answer Scoring. In Proceedings of NAACL: Human Language Technologies (pp. 1049–1054). URL: http://www.aclweb.org/anthology/N15-1111
- Sayyadi, H., & Raschid, L. (2013). A Graph Analytical Approach for Topic Detection. *ACM Trans. Internet Technol.*, *13*(2), 4:1–4:23. DOI: 10.1145/2542214.2542215
- Stanojević, S. (2017). Summarisation of group chat messages, MSc in Software engineering, U. of Belgrade, Belgrade
- Wu et al. (2018). Word Mover's Embedding: From Word2Vec to Document Embedding. *Proc. of the 2018 Conf. on Empirical Methods in Natural Language Processing*, 4524–4534.
- Yang, Y. & Pedersen, J.O. (1997). A Comparative Study on Feature Selection in Text Categorization. In *Proc. of the 14th Int'l Conf. on Machine Learning* (pp. 412-420). http://courses.ischool.berkeley.edu/i256/f06/papers/yang97comparative.pdf

Introduction to Text Mining

Jelena Jovanovic

Email: jeljov@gmail.com

Web: http://jelenajovanovic.net