Data-Driven Intelligent Systems

Lecture 24

Text Mining II/II

KNOWLEDGE
TECHNOLOGY

http://www.informatik.uni-hamburg.de/WTM/

Overview

- Information retrieval
  - Stop words; Stemming
  - Term-document matrix
  - Latent semantic indexing
  - Word2Vec
- Text classification
- Ontologies

Similarity-based Retrieval in Text Data

- Finds similar documents based on a set of common ***keywords***
- Answer should be based on the ***degree of relevance*** based on the nearness of the keywords, relative frequency of the keywords, etc.
- ***Stop list***
  - Set of words that are deemed ***irrelevant***, even though they may appear frequently
  - **E.g.**, `a, the, of, for, to, with,` etc.
  - Stop lists may vary when document set varies

Stop Words

- Many of the most frequently used words in English are almost worthless in retrieval and text mining – these words are called ***stop words***
  - the, of, and, to, ….
  - Typically up to about 400 to 500 such words
  - For an application or domain, specific stop words list may be constructed
- ***Why do we need to remove stop words***?
  - Reduce data file size
    - stop words account for 20-30% of total word counts
  - Improve efficiency
    - stop words have a large number of hits
    - stop words are not useful for searching or text mining

Stemming

- Several words are syntactic variants of each other since they share a common ***word stem***
- Techniques are used to find the root/stem of a word:
  - E.g.,

▪	user	engineering
▪	users	engineered
▪	used	engineer
▪	using	engineers
stem:	**use**	**engineer**

- This improves effectiveness of retrieval and text mining
  - match similar words
  - reduce indexing size
    - combing words with same roots may reduce indexing size as much as 40-50%.

Basic stemming algorithm example: Porter Algorithm

- remove ending
  - 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.”

Better than Stemmers: Lemmatizers

- Find the base of a word considering the intended meaning
  - E.g. ``saw´´

    → see (verb)

    → saw (noun)
- Requires context, e.g. by using a POS tagger
- Open area of research

- NLTK Python toolkit has both, stemmer and lemmatizer
  http//www.nltk.org

Overview

- Information retrieval
  - Stop words; Stemming
  - Term-document matrix
  - Latent semantic indexing
  - Word2Vec
- Text classification
- Ontologies

Term-Document Matrix

- Most common form of representation in text mining is the ***term-document*** matrix
  - Term: typically a single word, but could be a word phrase like “text mining”
  - Document: a generic term meaning a collection of text to be retrieved
  - Can be large - terms are often 50k or more, documents can be in the billions (www)
  - Can be binary or use counts

Term-Document Matrix Example (1)

Example: 10 documents: 6 terms

	Database	SQL	Index	Regression	Likelihood	linear
D1	24	21	9	0	0	3
D2	32	10	5	0	3	0
D3	12	16	5	0	0	0
D4	6	7	2	0	0	0
D5	43	31	20	0	3	0
D6	2	0	0	18	7	6
D7	0	0	1	32	12	0
D8	3	0	0	22	4	4
D9	1	0	0	34	27	25
D10	6	0	0	17	4	23

- Each document is just a vector of terms
  $D_i = (d_{i1}, d_{i2}, \ldots, d_{it})$
  - sometimes boolean

Distances in TD Matrices

- Given a term doc matrix representation
- Now we can define distances between documents D_i and D_j
- Elements of matrix can be {0,1} or term frequencies or normalized weights like TF-IDF
- Can use Euclidean distance or cosine similarity
- Cosine similarity proven to work well:

$$S_c(D_i, D_j) = \frac{\sum_{k=1}^{T} d_{ik} d_{jk}}{\sqrt{\sum_{k=1}^{T} d_{ik}^2 \sum_{k=1}^{T} d_{jk}^2}} = \frac{D_i \cdot D_j}{|D_i||D_j|}$$

$$= \cos(\theta)$$

angle between D_i and D_j

- if docs are the same, $S_c = 1$
- if nothing in common, $S_c = 0$

Term / Document Matrix Example (2)

Example: 10 documents: 6 terms

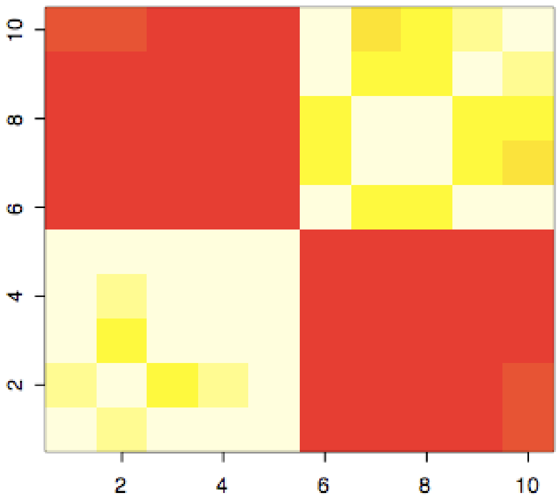
	Database	SQL	Index	Regression	Likelihood	linear
D1	24	21	9	0	0	3
D2	32	10	5	0	3	0
D3	12	16	5	0	0	0
D4	6	7	2	0	0	0
D5	43	31	20	0	3	0
D6	2	0	0	18	7	6
D7	0	0	1	32	12	0
D8	3	0	0	22	4	4
D9	1	0	0	34	27	25
D10	6	0	0	17	4	23

- Calculate cosine similarities / Euclidean distances
- What would you want these to look like?

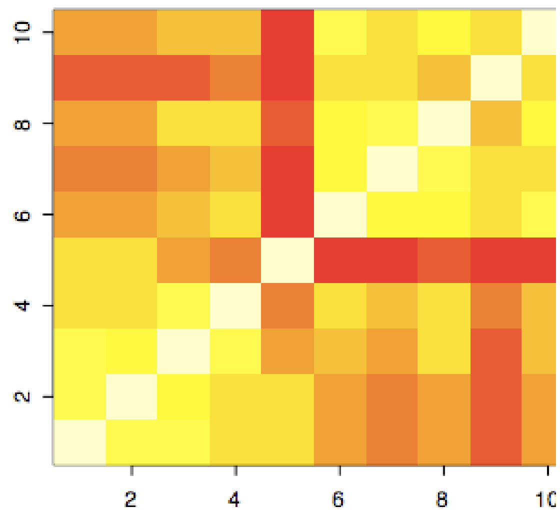
Visualisation of Document distance

- Images plot pairwise distances between documents

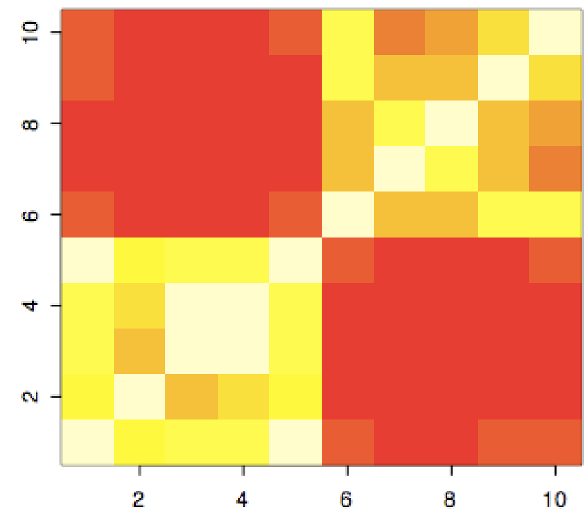
Cosine distance
1 – cosine similarity



Euclidean



scaled Euclidean



white: small distance
dark: large distance

R function: ‘image’

Overview

- Information retrieval
  - Stop words; Stemming
  - Term-document matrix
  - Latent semantic indexing
  - Word2Vec
- Text classification
- Ontologies

Towards Semantic Information Retrieval

- A query is a representation of the user's information needs
  - Normally a list of words.
- Once we have a TD matrix, queries can be represented as a vector in the same space
  - "Database Index" = (1,0,1,0,0,0)
- Query can be a simple question in natural language

- Calculate cosine similarity between query and documents
  - Returns a ranked vector of documents

Towards Semantic Information Retrieval

- Problem 1: similar queries can be posed in many ways
  - Synonymy: car *or* automobile; beetroot *or* beet; …
    → documents with either term are relevant
- Problem 2: semantics of query might remain unclear
  - Polysemy: crane (bird *or* construction equipment)
- Possibilities to address in particular problem 1:
  - Synonym lists or Thesauri ← imperfect and difficult to maintain
  - Latent Semantic Indexing (LSI), aka. Latent Semantic Analysis
    - tries to extract the latent ***semantics in the documents***
  - Word2Vec, vector representation of words
    - represents ***semantics of words***
- Search what I meant, not what I said!

Latent Semantic Indexing

- The TD matrix D has $N \times T$ entries, with T = #terms
- T is too large and should be reflected in k<<T typical "topics"
- Use singular value decomposition (SVD) to find k topics $\leftarrow$ generalization of principal component analysis (PCA)
  - Create a square matrix $D^T \cdot D$ of size $T \times T$
    - It reflects the *correlations between **terms*** over the documents
    - (like the affinity matrix A (normalized as L) in spectral clustering)
  - The first k principal components are k orthogonal basis vectors, which explain most variance in the data
  - Reduce data to an $N \times k$ matrix, with little information loss
    - (like the eigenvector matrix X in spectral clustering)
- Each *direction* is a linear combination of the input terms, and defines a *cluster* of "topics" in the data

Ex.: Eigenvectors to a block-diagonal Matrix

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$D^T \cdot D$, reflects correlations between terms

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \boxed{\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}} = \boxed{\begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix}}$$

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \boxed{\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}} = \boxed{\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}}$$

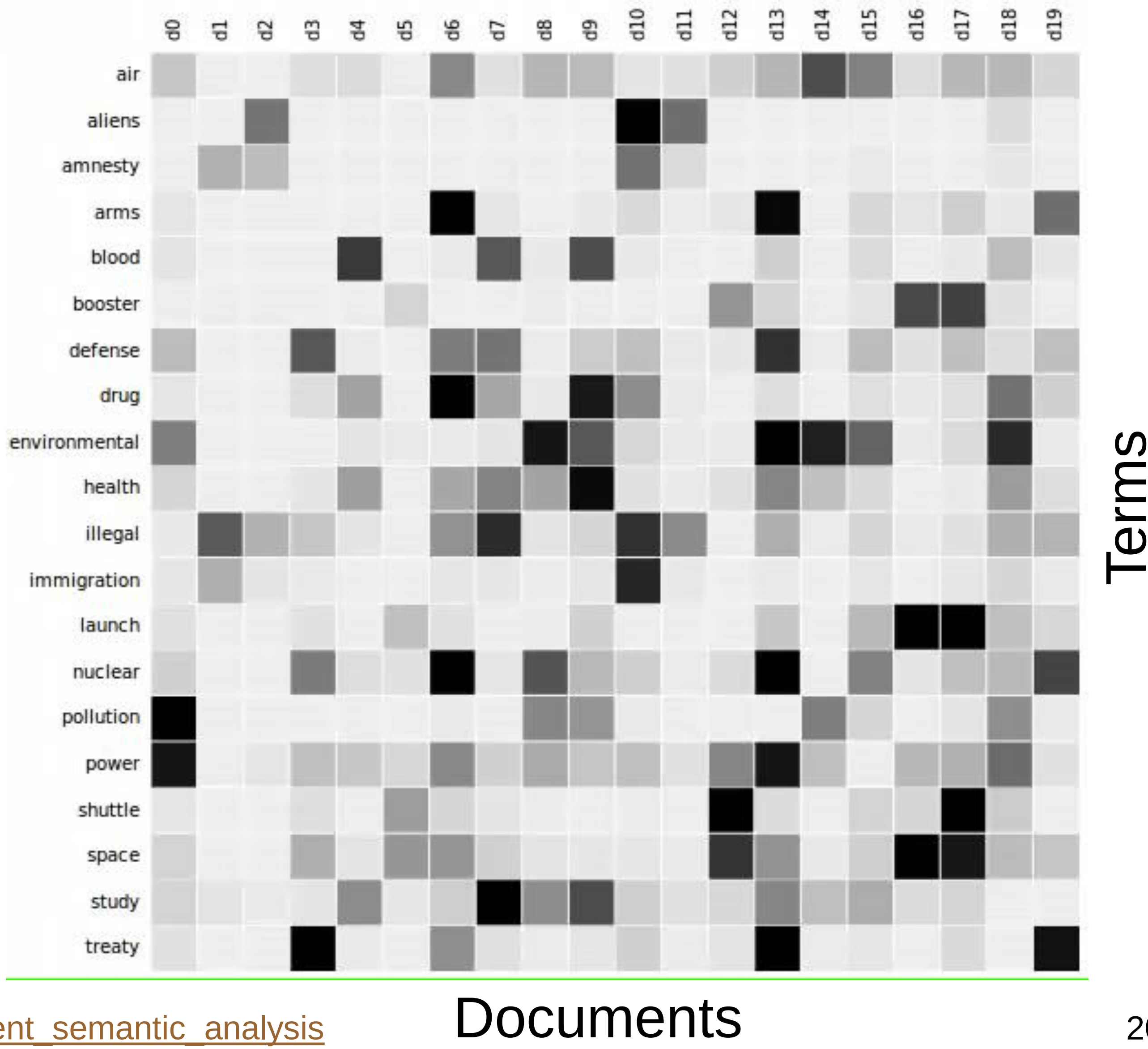
Eigenvectors
(get stretched, but not rotated, when multiplied with matrix)

$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right)$$

k ***eigenvectors/directions***, each being a linear combination of the input terms, forming “clusters” of terms into topics

LSI Example

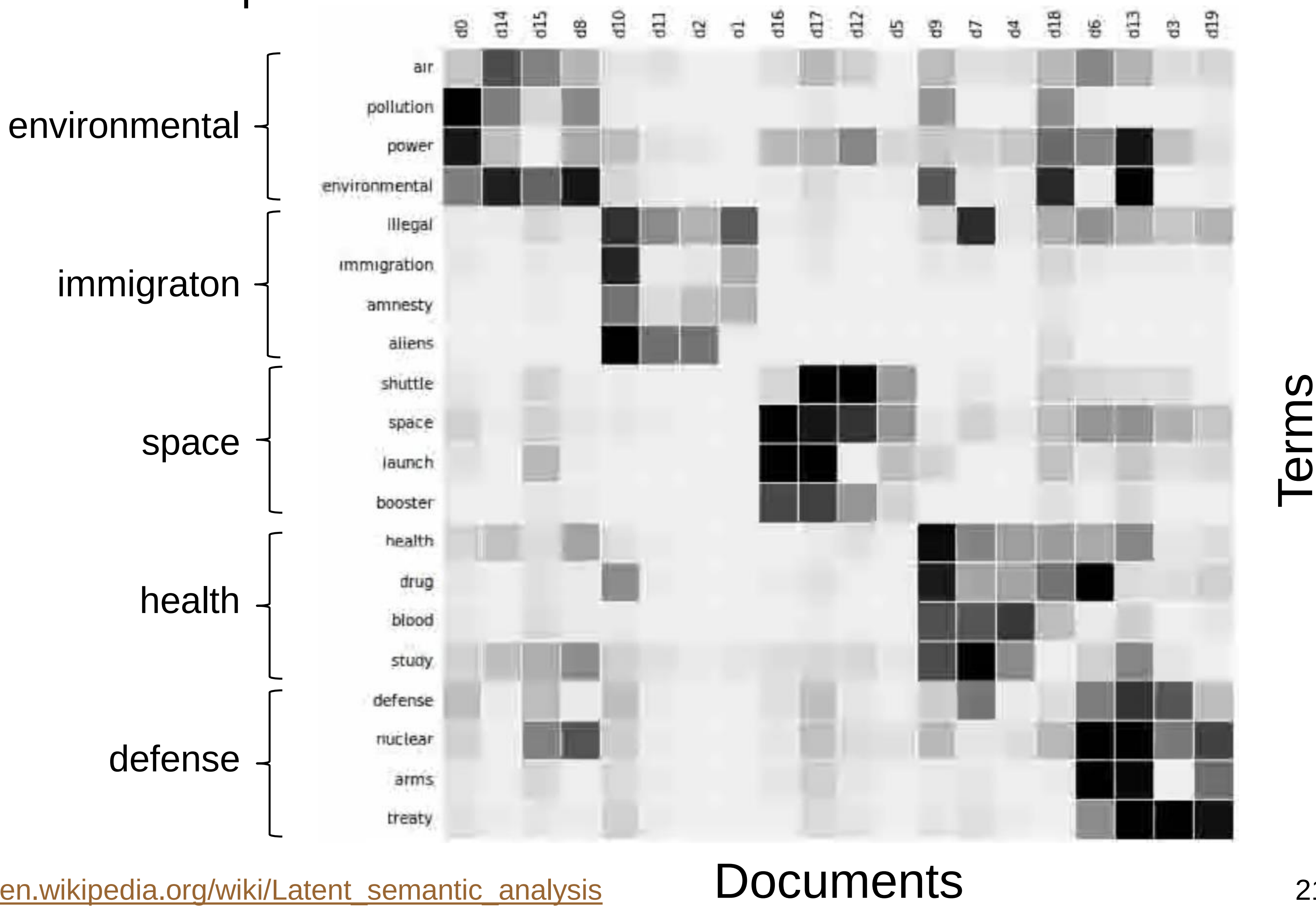
- T = 20 terms
- k= 5 PCs / Topics



https://en.wikipedia.org/wiki/Latent_semantic_analysis

LSI Example

- T = 20 terms
- k= 5 PCs / Topics

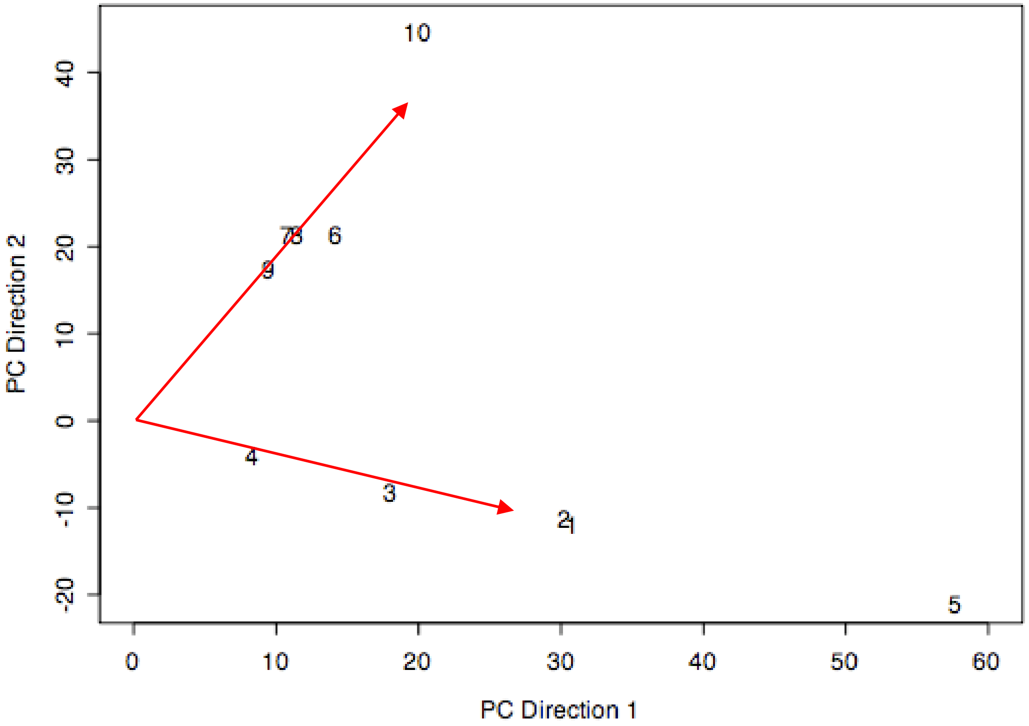


https://en.wikipedia.org/wiki/Latent_semantic_analysis

LSI Example

Previous example: 10 documents: 6 terms

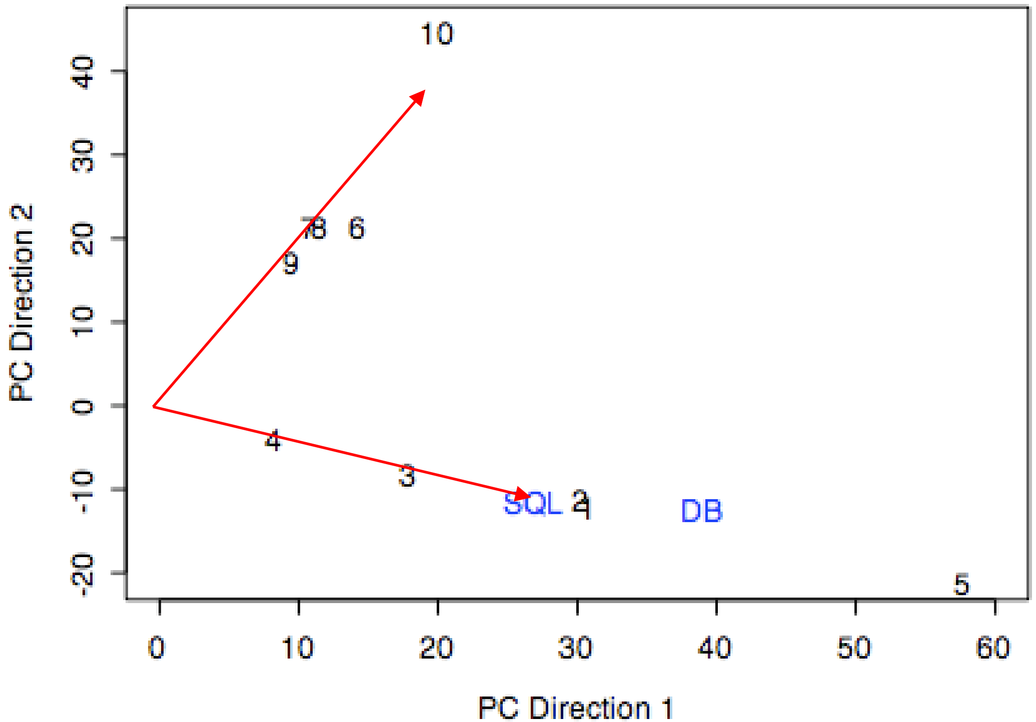
- Here, top 2 PC
  1. docs 1,2,3,4,5
  2. docs 6,7,8,9,10



LSI Example

Previous example: 10 documents: 6 terms

- Here, top 2 PC
  1. docs 1,2,3,4,5
  2. docs 6,7,8,9,10



- Two new documents, one with frequent term "SQL", another with frequent term "Databases"
- Even if they have no phrases directly in common, they are close in LSI space

Overview

- Information retrieval
  - Stop words; Stemming
  - Term-document matrix
  - Latent semantic indexing
  - Word2Vec
- Text classification
- Ontologies

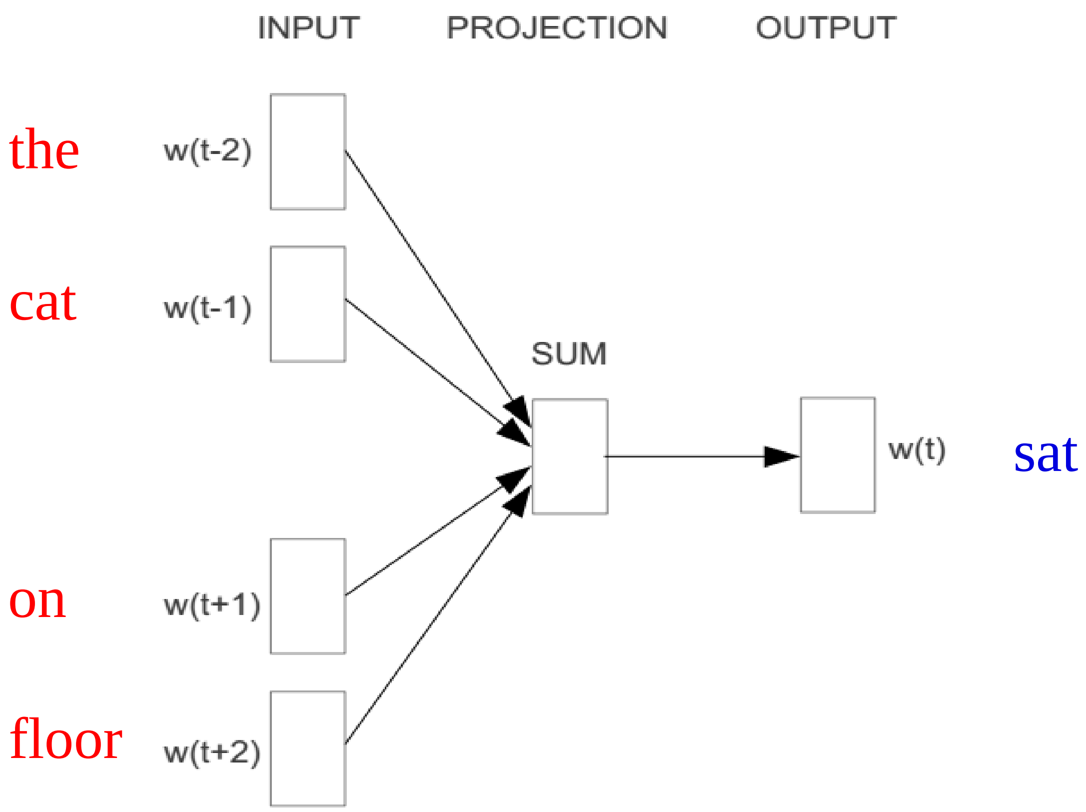
Word2vec

Another approach to capture the ***meaning*** of words

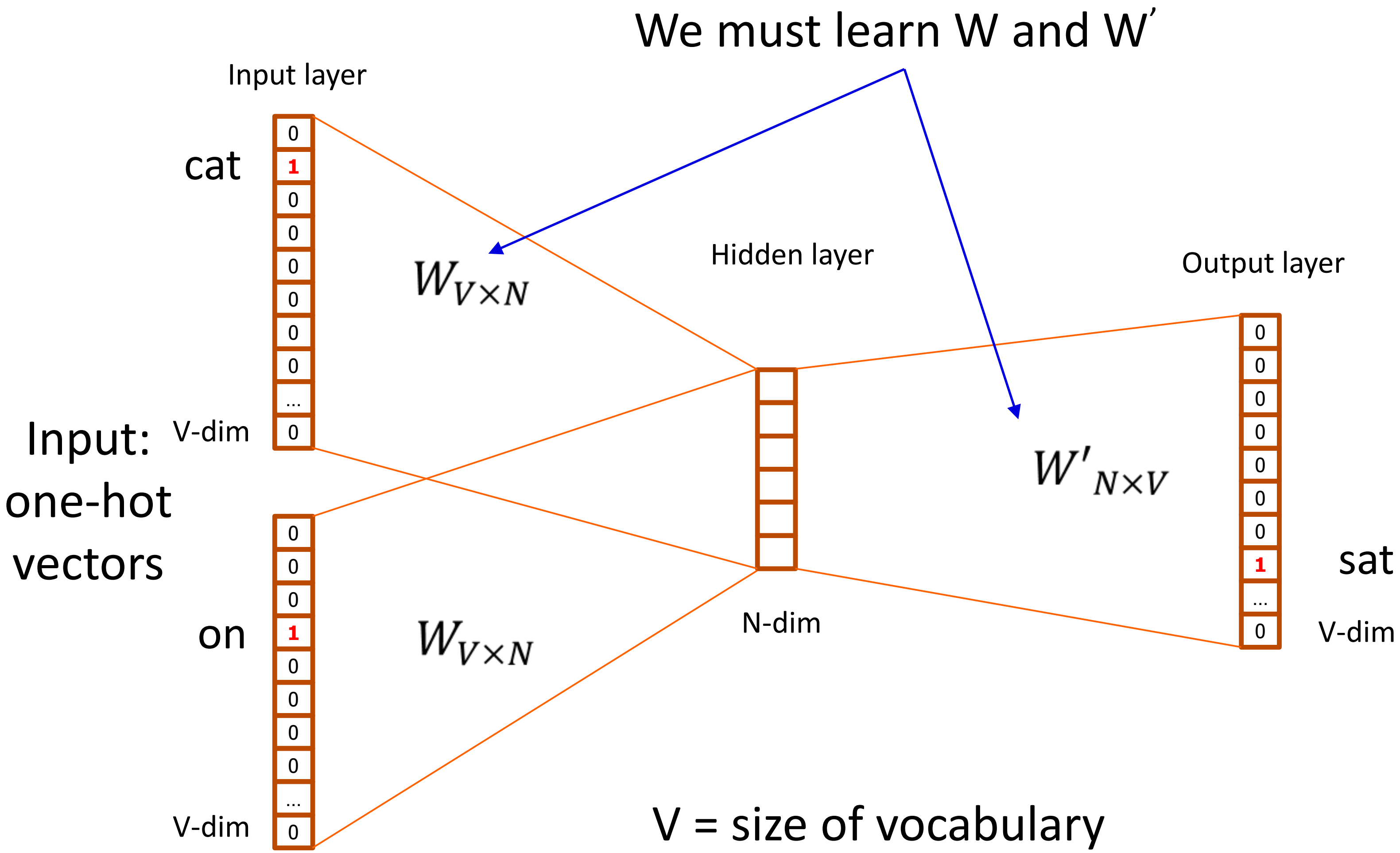
- Represent each word with a low-dimensional vector
- Word similarity = vector similarity
- Key ideas:
  - Predict surrounding words of every word
  - Using a multi-layer perceptron (MLP) with 1 hidden layer
  - Small number of *linear* hidden units → compression
  - *Large-scale training*
    - involving all domains / topics together

Word2vec – Continuous Bag-of-Words

- E.g. “The cat sat on floor”
  - window size = 2



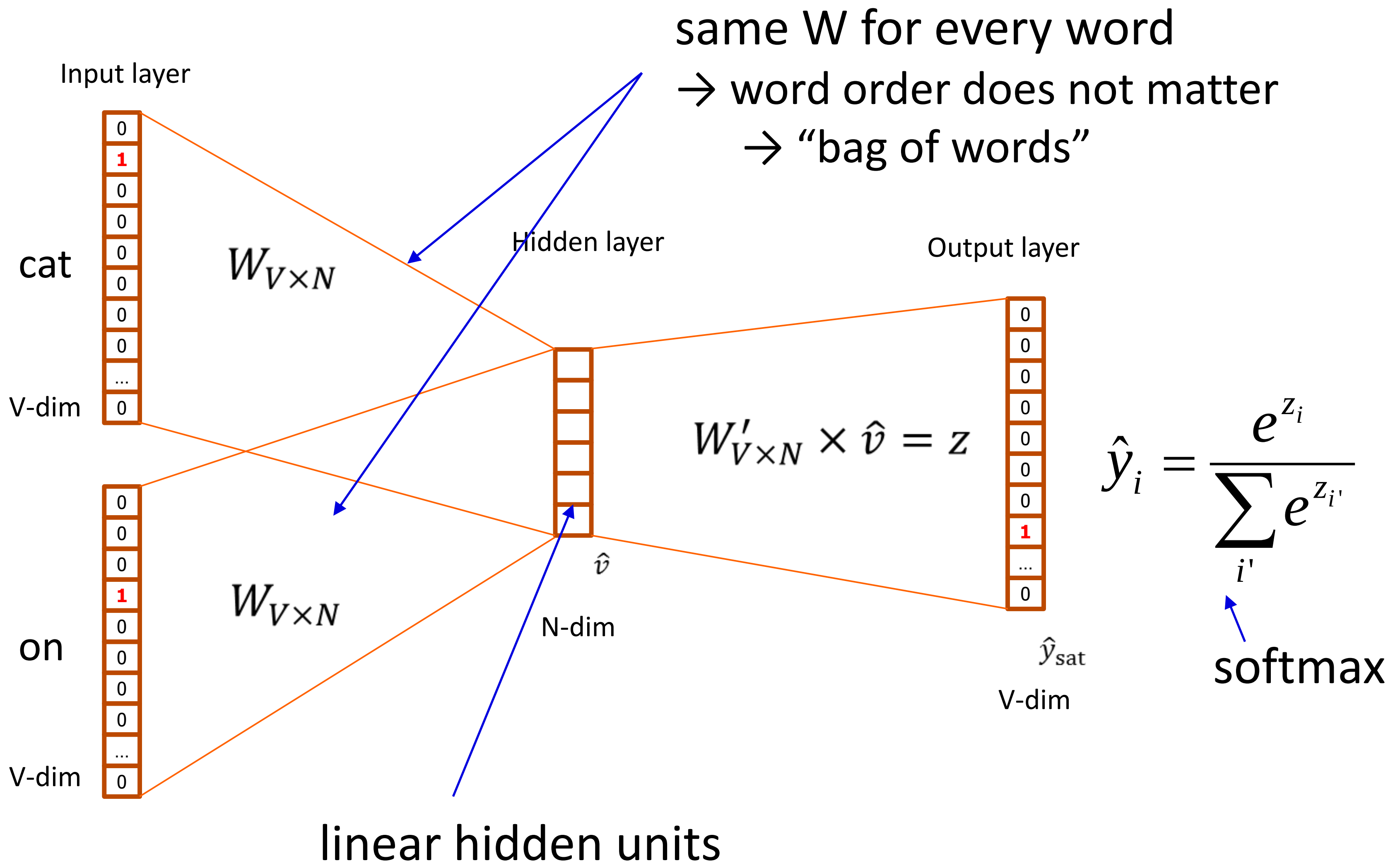
Word2vec – CBOW



V = size of vocabulary

N = size of word vector

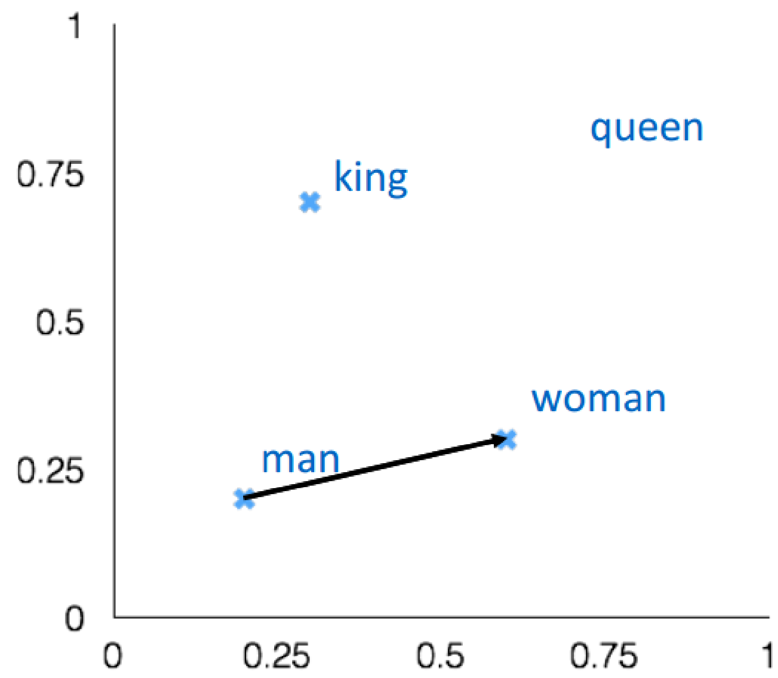
Word2vec – CBOW



Word2vec – Some Interesting Results

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]



- Word analogies – test for linear relationships (Mikolov, 2014)

Word2vec - links

- Implementations (google original and Python/gensim)
  - https://code.google.com/archive/p/word2vec/
  - https://rare-technologies.com/deep-learning-with-word2vec-and-gensim/

- Pretrained word vectors
  - Trained on 100 billion words from a Google News dataset
  - 3 million words, 300 features (* 4bytes/feature = 3.35 GB)
  - http://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/

Applications of Word Vectors

- Word Similarity
  - Synonyms (plane, aircraft)
  - Stemming, inflections/tense forms (thought -> think)
  - Clustering
- Machine translation
- POS tagging and named entity recognition
- Relation extraction
- Sentiment analysis (e.g. words nearby “happy” or “sad”)

Overview

- Information retrieval
  - Stop words; Stemming
  - Term-document matrix
  - Latent semantic indexing
  - Word2Vec
- Text classification
- Ontologies

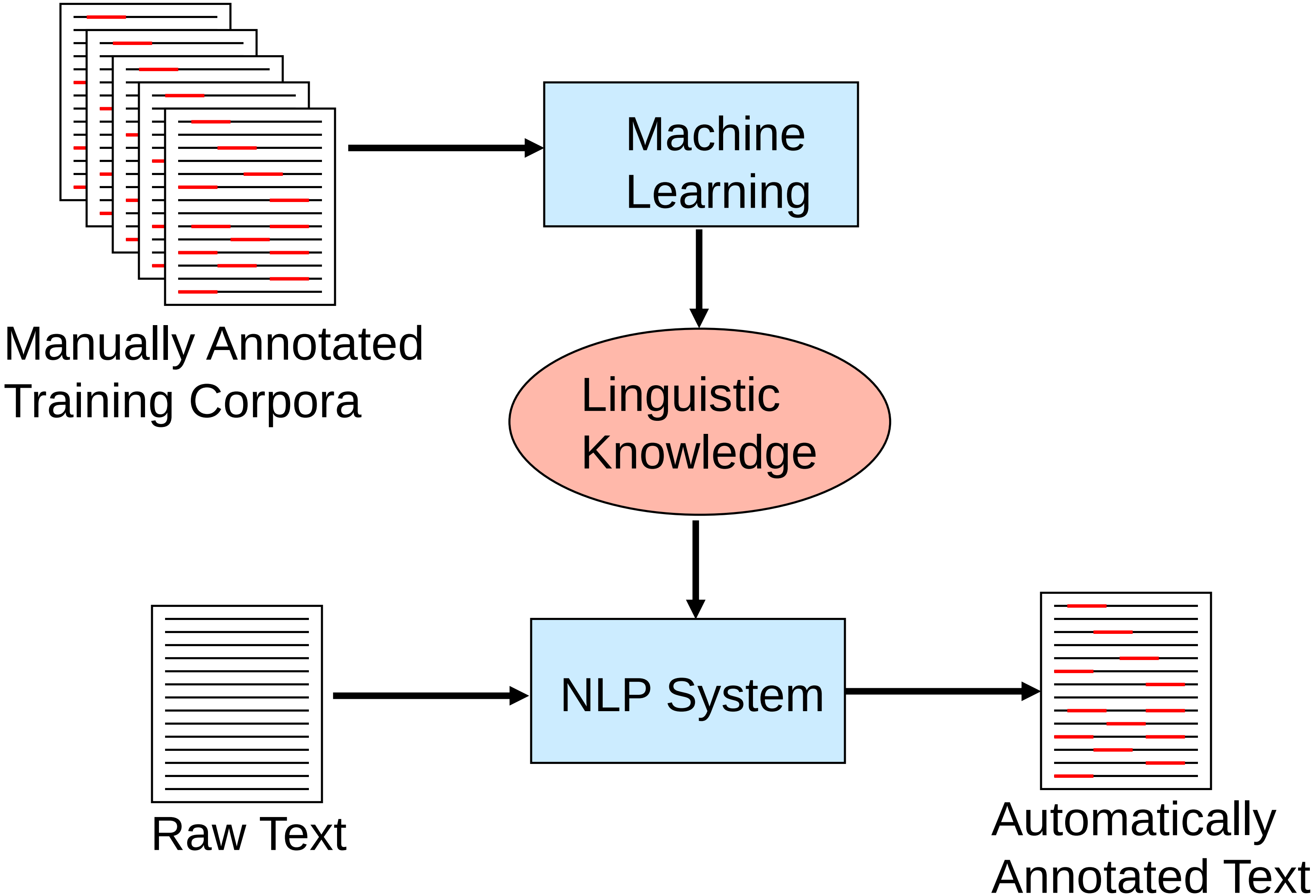
Manual Knowledge Acquisition

- Traditional ***rationalist*** approaches to language processing require human specialists to specify and formalize the required knowledge.
- ***Rules*** in language have numerous exceptions and irregularities.
  - "`All grammars leak.`" Edward Sapir (1921)
- Manually developed systems were expensive to develop and their abilities were limited and "brittle" (***not robust***).

Automatic Learning Approach

- Use machine learning methods to automatically acquire the required knowledge from appropriately annotated text corpora.
- ***``corpus based´´***, ***``statistical´´***, or ***``empirical´´*** approach
- Statistical learning methods widely used in NLP and Speech processing

Learning Approach



Advantages of the Learning Approach

- ***Large amounts*** of electronic text are now available
- Annotating/labeling corpora is ***easier*** and requires less expertise than manual knowledge engineering
- Learning algorithms today can handle large amounts of data and acquire accurate ***probabilistic knowledge***
- This knowledge allows ***robust*** processing
  - handles linguistic regularities as well as exceptions

Text Classification

- Motivation: automatic classification of text documents
  - Web pages, e-mails, corporate intranets, etc.
- Classification Process
  - Data preprocessing $\leftarrow$ *text-specific*
  - Define training- and validation set
  - Create the classification model / algorithm
  - Validate the model
  - Classify new text documents

Text Classification Algorithms

- K-Nearest Neighbors
- Decision Trees
- Boosting
- Naïve Bayes
- Support Vector Machines
- Neural Networks

Text Classification with deep CNN

- Preprocessing:
  - word2vec representation of words
  - bring all documents to same length (cutting or 0-padding)
- Model:
  - 1D-convolution .. pooling .. softmax on output (classes)
  - Observe training and validation error; then classify new docs
- Lift limitation of same-length texts:
  - Recurrent Neural Networks (hardly “understand” text)
  - Hierarchical Attention Networks
    (require a lot of training data)

https://medium.com/jatana/report-on-text-classification-using-cnn-rnn-han-f0e887214d5f

Text Classification with Tools

Text Classification Software

- Weka: http://www.cs.waikato.ac.nz/ml/weka/

- Natural Language Toolkit, NLTK
  https://www.nltk.org
- spaCy (written in Python and Cython)
  https://spacy.io

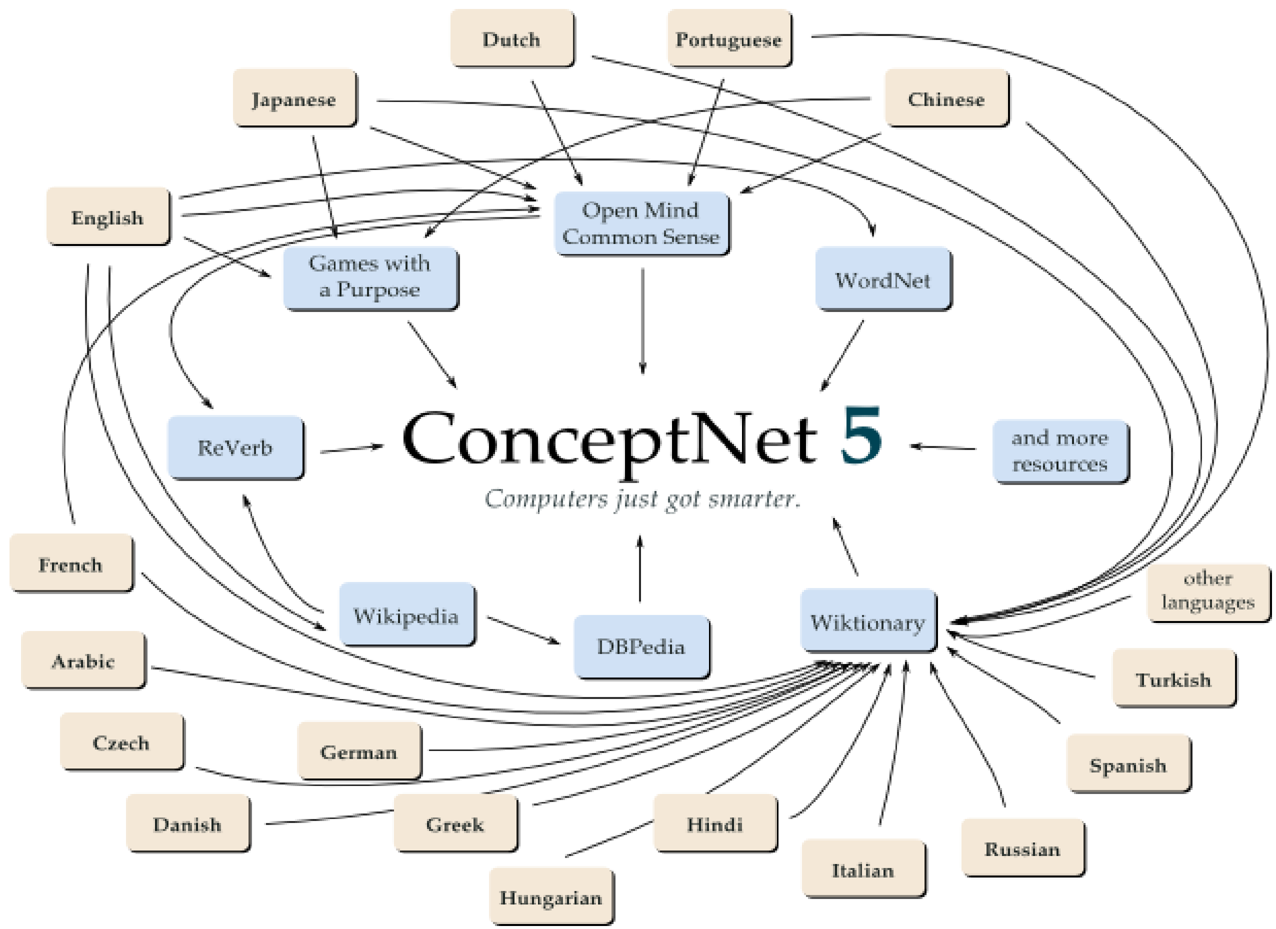
Overview

- Information retrieval
  - Stop words; Stemming
  - Term-document matrix
  - Latent semantic indexing
  - Word2Vec
- Text classification
- Ontologies

Ontologies

- An ***ontology*** defines types, properties and ***interrelationships*** of entities
- Can be represented as a ***semantic network***
- Early forms:
  - a ***thesaurus*** groups words according to similarity of meaning
  - a ***taxonomy*** is organized w.r.t. hierarchical relations
- Examples and extensions:
  - ConceptNet, WordNet, HowNet, Groningen Meaning Bank, BabelNet, CyC, Watson
  - DBpedia
    - data from Wikipedia documents
  - WikiData
    - metadata to supplement Wikipedia docs

ConceptNet –
an Ontology with Rich Semantic Relationships



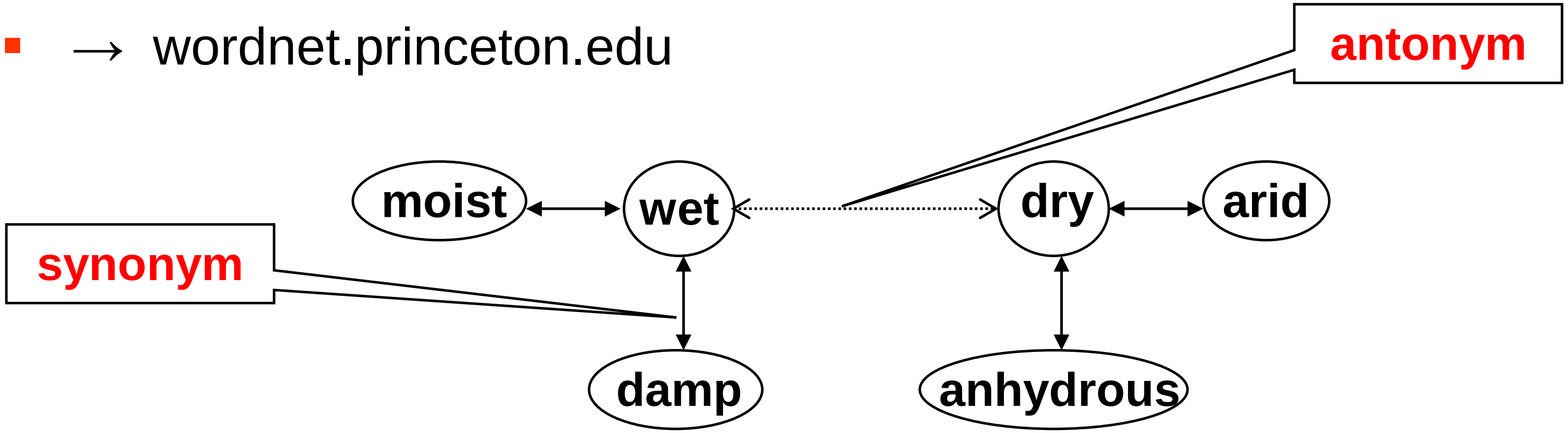
replace "toast" with any other word!

Play around:
http://conceptnet5.media.mit.edu/data/5.4/c/en/toast

WordNet Lexicon

An extensive ***lexical network*** for the English language

- Contains over ***138,838 words***.
- Several graphs, one for each ***part-of-speech***.
- ***Synsets*** (sets of cognitive synonyms), each defining a semantic sense.
- ***Relationship*** information (antonym, hyponym, …)
- *Encodes some of the lexicon that humans carry with them when interpreting text.*
- → wordnet.princeton.edu



WordNet Lexicon

- Nouns:
  - > 90,000 forms
  - 116,000 senses
  - Relations →

hypernym	breakfast -> meal
hyponym	meal -> lunch
has-member	faculty -> professor
member-of	copilot -> crew
has-part	table -> leg
part-of	course -> meal
antonym	leader -> follower

- Verbs
  - >10,000 forms
  - 20,000 senses

Hypernym	fly-> travel
Troponym	walk -> stroll
Entails	snore -> sleep
Antonym	increase -> decrease

BabelNet – a new Multilingual Ontology

BabelNet 1.1

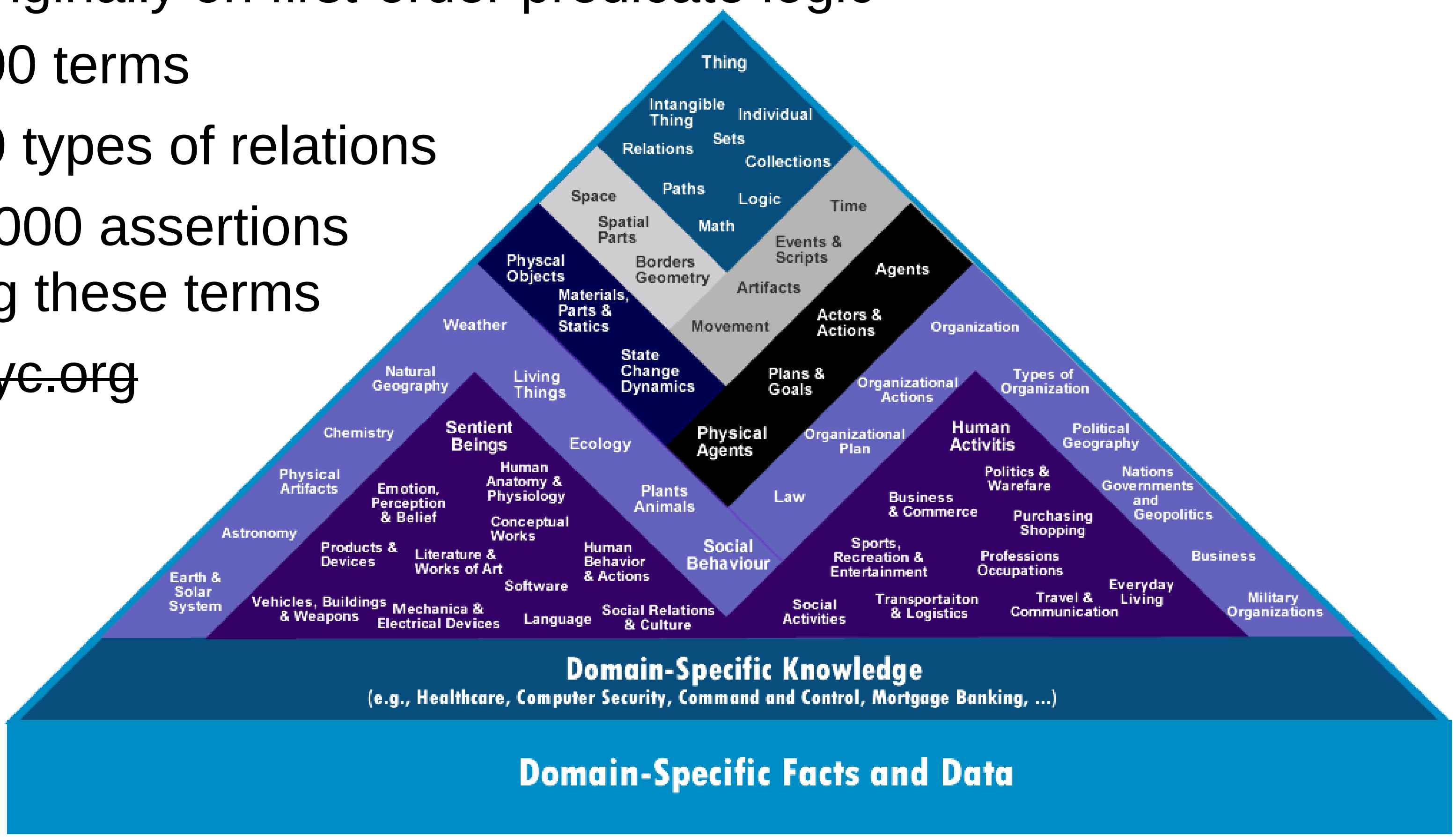
A very large multilingual ontology

A **very large multilingual ontology** with **5.5 millions** of concepts • A wide-coverage **"encyclopedic dictionary"** • Obtained from the automatic integration of **WordNet** and **Wikipedia** • Enriched with **automatic translations** of its concepts • Connected to the **Linguistic Linked Open Data** cloud!

http://www.babelnet.org/

CyC

- CyC: Knowledge base and ontology framework
- Q&A system in the 80s; can reason things never directly told
- Built originally on first-order predicate logic
- 500,000 terms
- 17,000 types of relations
- 7,000,000 assertions relating these terms
- ~~opencyc.org~~



Watson and the DeepQA Text Mining project

Watson: computer system to compete in real time with expert humans in the _Jeopardy_ Quiz

- Content acquisition: ***Domain analysis***, automatic corpus expansion, leveraging of the content
- Question analysis: Parsing, lexical answer type detection, ***semantic role labelling***, co-referencing, syntactic and semantic reasoning, decomposition
- Hypothesis generation: Get best candidates based on ***search*** and ***constraint satisfaction***
- Filtering, scoring and ranking: Machine learning and much more to estimate confidence.

Watson on Jeopardy!

Further reading:

- IBM's Watson/DeepQA: http://dl.acm.org/citation.cfm?id=2019525
- In the news: http://www.bbc.co.uk/news/technology-20159531

Summary

- Much available information is stored in text databases
  - Growing collections of documents from various sources
- Todays tools become increasingly essential
  - ***Classify*** documents
  - ***Compare*** documents, ***retrieve*** and ***rank*** importance
  - ***Find patterns*** and trends
- Information retrieval tools are for everybody`s taste
  - Simple count-based (bag of words)
  - Analysing grammar
  - Statistical learning methods

Finish with some fun?
(built by our team some time ago ….)