Classification of biomedical texts

Crash course text processing & representation

(Slides partially taken from Ulf Leser)

Mario Sänger (WBI, HU Berlin)

saengema@informatik.hu-berlin.de

We are hiring!

Studentische Hilfskraft, Forschung und Lehre

In der Arbeitsgruppe "Wissensmanagement in der Bioinformatik" am Institut für Informatik der Humboldt-Universität zu Berlin ist ab 1.6.2020 eine studentische Hilfskraftstelle (40h/Monat, 2 Jahre) zu besetzen. Der/die Stelleninhaber*in unterstützt uns in der Lehre (als Korrektor*in bzw. Tutor*in) und arbeitet an Forschungsprojekten am Lehrstuhl mit. Diese beschäftigen sich mit angewandtem Maschinellem Lernen, biomedizinischen Text Minings, Informationsintegration, der skalierbaren verteilten Datenanalyse, und Bioinformatik für individualisierte Medizin.

Aufgaben

- Erstellung von Softwareprototypen
- Mitarbeit an Forschungsprojekten im Umfeld der biomedizinischen Datenanalyse
- Unterstützung in der Lehre

Voraussetzungen

- Studium der Informatik oder eines angrenzenden Fachs
- Vertiefte Erfahrung im Programmieren
- Erfahrung in der statistischen Datenanalyse und/oder der Bioinformatik
- Grosses Interesse an der angewandten Forschung
- Ein hohes Maß an Eigenmotivation und Kommunikationsfähigkeit / Teamfähigkeit
- Gutes Englisch

https://www.informatik.hu-berlin.de/de/forschung/gebiete/wbi/jobs/shk_haushalt_2004

Outline

- Text pre-processing
  - Definition and logical view
  - Text pre-processing steps

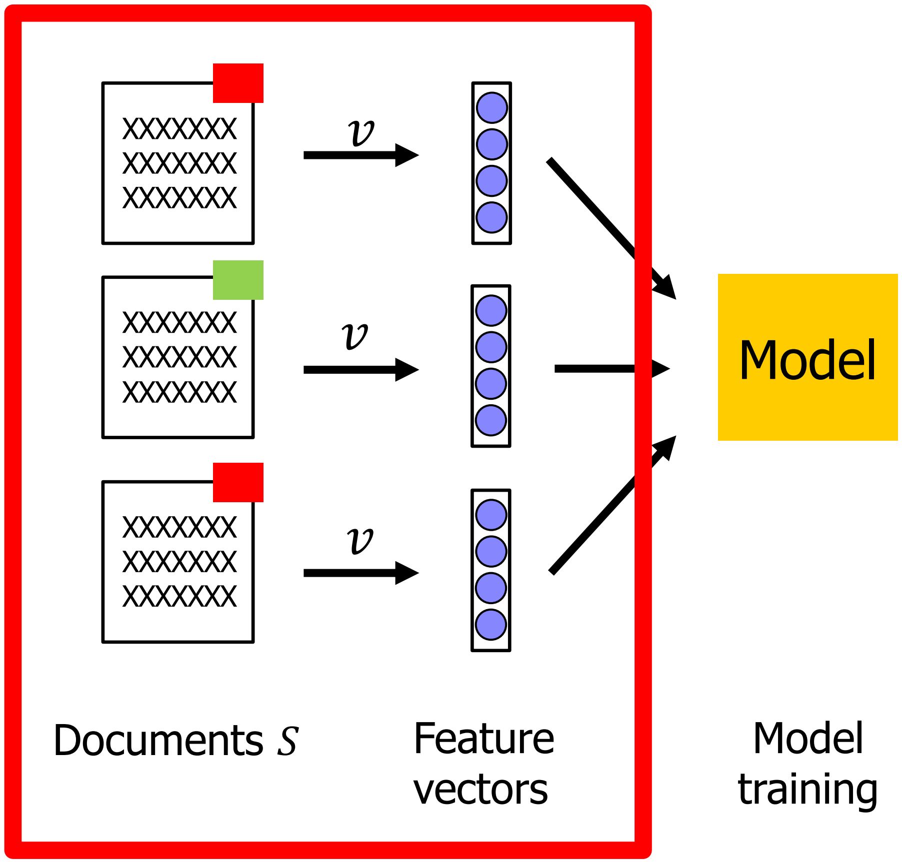
- Text representation
  - Bag of Words (BOW)
  - TF-IDF Weighting
  - Word embeddings

- Feature engineering
  - Feature selection

Text pre-processing

Supervised text classification

- Given a set D of documents and a set of classes C. A classifier is a function $f: D \rightarrow C$



- Problems
  - Finding enough training data
  - Finding the best pre-processing (tokenization, case, POS tag set …)
  - Finding the best features
  - Finding a good classifier (~ assigning as many docs as possible to their correct class)

Definitions

- A document as a sequence of sentences
- A sentence is a sequence of tokens
- A token is the smallest unit of text (words, numbers, …)
- A concept is the mental representation of a "thing"
- A term is a token or a set of tokens representing a concept
  - "San" is a token, but not a term
  - "San Francisco" has two tokens but is only one term
  - Dictionaries usually contain terms, not tokens

Definition

- A homonym is a term representing multiple concepts
- A synonym is a term representing a concept which may also be represented by other terms
- A syn-set is a set of synonyms representing the same concept

- “Word” can denote either a token or a term
  - We (and many other) will mostly make no difference between token and terms

Text retrieval

- We will not cover the details of text retrieval process and expected the textual content of our documents as given

- Text retrieval and conversion includes ...
  - .... download or crawl the data source
  - .... transform PDF, XML, HTML, ... into ASCII / Unicode
  - .... handling formatting instructions, special characters, formulas, tables, footnotes, images, ...
  - .... find the net content (no ads, navigations bars,...) in web documents
  - ....

Tokenization

- Fundamental elements of text processing systems: token
- Simple approach: search for „ „ (blanks)
  - “A state-of-the-art Z-9 Firebird was purchased on 3/12/1995.”
  - “[Bis[1,2-cyclohexanedionedioximato(1-)-O]- [1,2-cyclohexanedione dioximato(2-)-O]methyl-borato(2-)-N,N0,N00,N000,N0000,N00000)-chlorotechnetium) belongs to a family of …“
- Typical approach (but many (domain-specific) variations)
  - Treat hyphens / parentheses as blanks
  - Remove “.” (after sentence splitting)
- Recent approaches: split words into subwords [3]
  - E.g. “playing” -> ”play” and “#ing”
  - Explicitly models morphological information (prefixes, suffixes,...)

Sentence splitting

- Most linguistic analysis works on sentence level
- Sentences group together entities and statements
- Naive approach: reg-exp search for "[.?!;] " (with blank!)
  - Abbreviations:
    - "C. Elegans is a worm which …"; "This does not hold for the U.S."
  - Errors (due to previous normalization steps)
    - "is not clear.Conclusions.We reported on …"
  - Proper names:
    - ".NET is a technique for …"

- State-of-the-art tokenizer and splitter use rule- or classification-based approaches [1,2]

Case – A Difficult Case

- Should all text be converted to lower case letters?
- Advantages
  - Decreases number of words
  - Word-based similarity gets simpler
- Disadvantages
  - No abbreviations
  - Loss of important hints for sentence splitting
  - Loss of important hints for NER, RE, …
  - Loss of semantic info (proper names, Essen versus essen,…)
- Different impact in different languages (German / English)

Word stems and root forms

- We could also opt to treat different forms of the "same" word equivalently

playing → play
plays → play
played → play

common root form

am, are, is → be

Car, cars, car's, cars' → *car*

Stemming

- Reduce inflected words to their word stem
  - In general: use (heuristic) rules to prune affixes from inflected word forms
    - If word ends with "*ing*" – remove "*ing*"
    - If word ends with "*ed*" – remove "*ed*"
    - If word ends with "ly" – remove "ly"
    - …
- Word stems doesn't have to be valid words
  - Example: *argue*, *argued*, *argues*, *arguing*, *argus* -> argu
- Algorithms: PorterStemmer, LancasterStemmer

Lemmatization

- Reduces the inflected words properly ensuring that the root word belongs to the language
  - Lemma: is the canonical form, dictionary form, or citation form of a set of words
  - Depends on correctly identifying the intended part of speech and meaning of a word (unlike stemming)

- Examples
  - "*better*" has "*good*" as it's lemma
  - "*meeting*" can be either the base form of a noun or a form of a verb ("*to meet*") depending on the context
    - "in our last meeting" vs. "We are meeting again tomorrow"

Stop words

- Words that are so frequent that their removal (hopefully) does not change the meaning of a document
  - English: Top-2: 10% of all tokens; Top6: 20%; Top-50: 50%
  - English (top-10; LOB corpus): the, of, and, to, a, in, that, is, was, it
  - German(top-100): aber, als, am, an, auch, auf, aus, bei, bin, …
- Consequences
  - Removing top-100 stop words reduces ~40% of all tokens
  - Hopefully increases precision due to less spurious hits
- Variations
  - Remove top 10, 100, 1000, … words
  - Language-specific, domain-specific, corpus-specific

Example

The children of obese and overweight parents have an increased risk of obesity. Subjects with two obese parents are fatter in childhood and also show a stronger pattern of tracking from childhood to adulthood. As the prevalence of parental obesity increases in the general population the extent of child to adult tracking of BMI is likely to strengthen.

$\downarrow$ 100 stop words

children obese overweight parents increased risk obesity. Subjects obese parents fatter childhood show stronger pattern tracking childhood adulthood. prevalence parental obesity increases general population extent child adult tracking BMI likely strengthen.

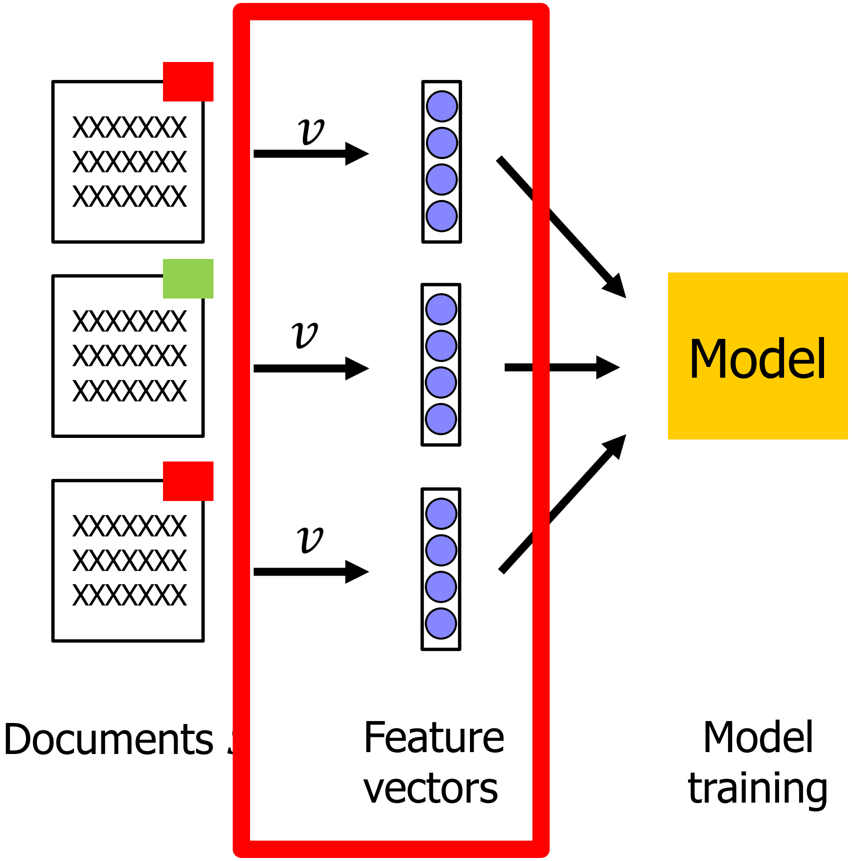
$\downarrow$ 10 000 stop words

obese overweight obesity obese fatter adulthood prevalence parental obesity BMI

Text representation

Supervised text classification

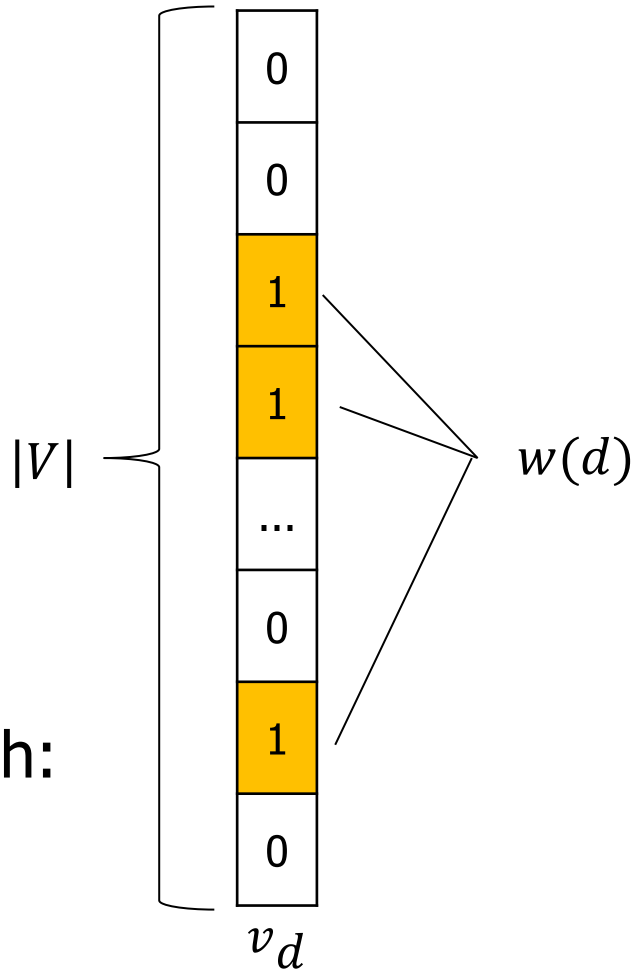
- Given a set D of documents and a set of classes C. A classifier is a function $f: D \rightarrow C$



- Problems
  - Finding enough training data
  - Finding the best pre-processing (tokenization, case, POS tag set …)
  - Finding the best features
  - Finding a good classifier (~ assigning as many docs as possible to their correct class)

Bag of Words (BoW)

- Let D be the set of all normalized documents
  - $d \in D$ is a document
- Let V be the set of all distinct tokens in D
  - V is also called the vocabulary
- Let w be the function that maps a given d to its set of distinct tokens in V (its bag-of-words)
- Let v_d by a vector of size $|V|$ for d with:
  - $v_d[i] = 0 \ iff \ t_i \notin w(d)$
  - $v_d[i] = 1 \ iff \ t_i \in w(d)$



BoW example

- Assume stop word removal, stemming and binary weights
  - Stop words: “the”, “in”, “is”, “a”, “other”, “and”

	documents	cat	sat	hat	eat	food	dog	like
1	the cat sat in the hat							
2	the cat is eating cat food							
3	the dog likes a cat							
4	the dog and the cat eat and eat							
5	the dog eats cat and dog food							

BoW example

- Assume stop word removal, stemming and binary weights
  - Stop words: “the”, “in”, “is”, “a”, “other”, “and”

	documents	cat	sat	hat	eat	food	dog	like
1	the cat sat in the hat	1	1	1				
2	the cat is eating cat food							
3	the dog likes a cat							
4	the dog and the cat eat and eat							
5	the dog eats cat and dog food							

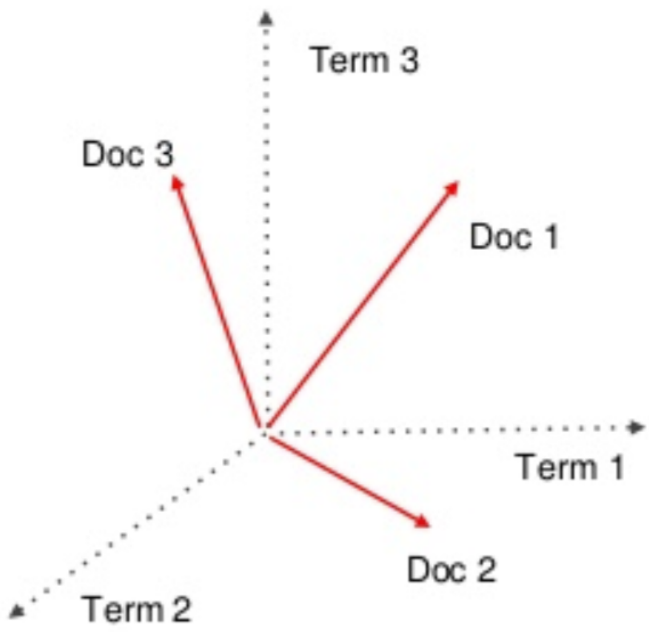
BoW example

- Assume stop word removal, stemming and binary weights
  - Stop words: “the”, “in”, “is”, “a”, “other”, “and”

	documents	**cat**	**sat**	**hat**	**eat**	**food**	**dog**	**like**
1	the cat sat in the hat	1	1	1				
2	the cat is eating cat food	1			1	1		
3	the dog likes a cat	1					1	1
4	the dog and the cat eat and eat	1			1		1	
5	the dog eats cat and dog food	1			1	1	1	

Vector space model

- Each term is one dimension
  - Different suggestions for determining co-ordinates, i.e., term weights
- The closest docs are the most similar ones
  - Rationale: vectors correspond to themes which are loosely related to sets of tokens / terms
  - Distance between vectors ~ distance between themes
  - Different suggestions for defining distance



The angle between two vectors

- Recall: The scalar product between two vectors v and w of equal dimension is defined as

$$v \circ w = |v| * |w| * \cos(v, w)$$

- This gives us the angle

$$\cos(v, w) = \frac{v \circ w}{|v| * |w|}$$

  - With

$$|v| = \sqrt{\sum_i v[i]^2} \qquad v \circ w = \sum_i v[i] * w[i]$$

Example: document similarity

1	1	1	1				
2	1			1	1		
3	1					1	1
4	1			1		1	
5	1			1	1	1	

$$sim(v_1, v_2) = \frac{\sum_i v_1[i] * v_2[i]}{\sqrt{\sum_i v_1[i]^2} * \sqrt{\sum_i v_2[i]^2}}$$

- Similarity between d_1 and d_2

$$sim(d_1, d_2) = \frac{(1*1+1*0+1*0+0*1+0*1+0*0+0*0)}{\sqrt{(1^2+1^2+1^2)} * \sqrt{(1^2+1^2+1^2)}} = \frac{1}{\sqrt{3} * \sqrt{3}} = \frac{1}{3}$$

- Similarity between d_4 and d_5

$$sim(d_4, d_5) = \frac{(1+1+1+1)}{(\sqrt{3} * \sqrt{4})} = \frac{4}{(\sqrt{3} * \sqrt{4})} \quad \sim 1.154$$

TF weighting

- Let D be a document collection, V be the set of all terms in D, $d \in D$ and $\mathrm{t} \in V$
  - The term frequency tf_{dt} is the frequency of t in d

	documents	cat	sat	hat	eat	food	dog	like
1	the cat sat in the hat	1	1	1				
2	the cat eats cat food	2			1	1		
3	the dog likes a cat	1					1	1
4	the dog and the cat eat and eat	1			2		1	
5	the dog eats cat and dog food	1			1	1	2	

Example: document similarity

1	1	1	1				
2	2			1	1		
3	1					1	1
4	1			2		1	
5	1			1	1	2	

$$sim(v_1, v_2) = \frac{\sum_i v_1[i] * v_2[i]}{\sqrt{\sum_i v_1[i]^2} * \sqrt{\sum_i v_2[i]^2}}$$

- Similarity between d_1 and d_2

$$sim(d_1, d_2) = \frac{(1 * 2)}{\sqrt{(1^2+1^2+1^2)} * \sqrt{(2^2+1^2+1^2)}} = \frac{2}{\sqrt{3} * \sqrt{6}} \quad \sim 0.4714$$

- Similarity between d_4 and d_5

$$sim(d_4, d_5) = \frac{(1 + 2 + 1 + 2)}{(\sqrt{6} * \sqrt{7})} = \frac{6}{(\sqrt{6} * \sqrt{7})} \quad \sim 0.9258$$

TF-IDF weighting

- Let D be a document collection, V be the set of all terms in D, $d \in D$ and $\mathrm{t} \in V$
- The term frequency tf_{dt} is the frequency of t in d
- The document frequency df_t is the frequency of docs in D containing t
  - This should rather be called "corpus frequency"
  - May also be defined as the frequency of occurrences of t in D
  - Both definitions are valid and both are used
- The inverse document frequency is defined as $idf_t = |D|/df_t$
  - In practice, one usually uses $idf_t = \log(|D|/df_t)$

TF-IDF weighting (in short)

- TF-IDF weight for a token t in document d is defined as:

$$v_d[t] = tf_{dt} * idf_t$$

- Give tokens in a document d high weights which are ...
  - frequent in d and
  - infrequent in D

- IDF deals with the consequences of Zipf's law
  - The few very frequent (and unspecific) terms get lower scores
  - The many infrequent (and specific) terms get higher scores

Shortcomings

- No treatment of synonyms (query expansion, …)
- No treatment of homonyms
  - Different senses = different dimensions
  - We would need to disambiguate terms into their senses (later)
- Term-order independent
  - But order carries semantic meaning
- Assumes that all terms are independent
  - Clearly wrong: some terms are semantically closer than others
    - Their co-appearance doesn’t mean more than only one appearance
    - The appearance of “red” in a doc with “wine” doesn’t mean much

Distributional Semantics

- „You shall know a word by the company it keeps" [7]
  - The distribution of words co-occurring (context) with a given word x is characteristic for x
  - To learn about x, look at its context
  - If x and y are semantically similar, also their contexts are similar
  - If x and y are a bit different, also their contexts will be a bit different
  - Holds in all domains and all corpora of sufficient size
- Central idea: Represent a word by its context
  - For similarity: compare contexts, not single tokens / terms
- Approaches can be grouped in two categories: count- and prediction-based methods

Count-based: Naive Approach

- Given a large corpus D and a vocabulary V
- Define a context window (typically a sentence or n words)
- Represent every $t \in V$ as a $|V|$-dimensional vector v_t
  - Find set W of all context windows containing t
  - For every $t' \neq t$, count frequency of t' in W: $v_t[t'] = freq(t', W)$
- Similarity: Compute cosine similarity between word vectors
- Problem: Still high dimensional representation
  - Many dimensions will be 0 for a given tokens t
  - We need an efficient and conservative dimensionality reduction
    - Efficient: Fast to compute; conservative: distances are preserved
    - Algorithms: Principal component analysis, Singular value decomposition

Prediction-based approaches

- Very popular technique since approximately 2013
- Idea: Cast the task to an classification problem and use machine learning techniques
  - No algebraic solution - though the border is not always clear at all
- Goal: Learning word vectors (“word embeddings”)
  - Low dimensional – typically 100-500 (a hyperparameter)
  - Unsupervised learning – may use extremely large corpora
  - Specific techniques to scale-up training (e.g. GPUs)
  - Can be precomputed and used without re-training in many applications / tasks

Language Modelling

- Task: Given a sentence prefix, predict the next word

The cat sat on the _____

wall?   jumping

mat?

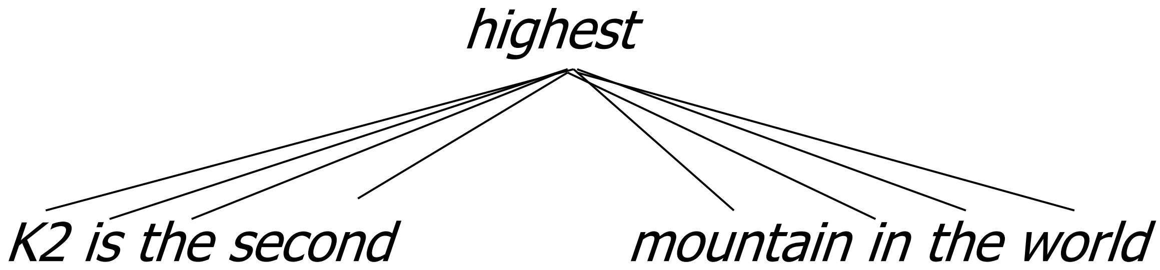
- Can be modelled as multi-class classification problem
  - There are as many classes as words
- Effective task to pretrain word embeddings (and neural networks) [3,4,5,6,8]
  - Huge text collections available as training data

Word2Vec

- Introduces two different language modelling tasks
  - CBOW: Given the context words, predict the word in the middle

K2 is the second ______ mountain in the world

  - SkipGram: Given the center word, predict the words in the context



SkipGram learning task

- Given a sequence of training words $w_1, w_2, w_3, \ldots, w_T$
  - Maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

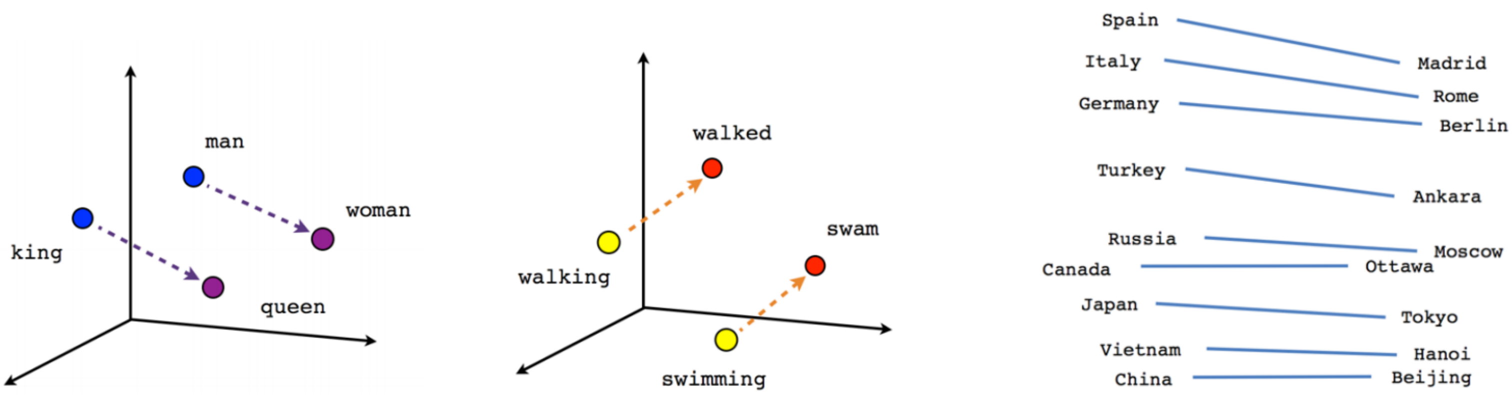
  - c is the size of the training context

- The probability $p(w_O|w_I)$ is modelled as softmax:

$$p(w_O|w_I) = \frac{\exp({v'}_{w_O} * {}^{T}v_{w_I})}{\sum_{w=1}^{V}\exp({v'}_{w} * {}^{T}v_{w_I})}$$

  - v_w and v'_w are the input and output representation of word w

Does it work?



Male-Female Verb tense Country-Capital

king – man ~ queen – woman

walking – walked ~ swimming – swam

Russia – Moscow ~ Vietnam – Hanoi

https://cdn-images-1.medium.com/max/2000/1*2r1yj0zPAuaSGZeQfG6Wtw.png

All shortcomings eliminated?

- No! For instance, word embeddings still suffer to model homonyms
  - Each word gets a single, fixed embedding assigned
  - No possibility to reflect different meanings!

- Improvement: Contextualized word embeddings
  - Do not assign a fixed vector to each word
  - Calculate a distinct embedding for each occurrence of a word based on it‵s current context
  - Plethora different approaches: ELMO, FLAIR, BERT, GPT, ... [3,4,5,6,8]

Feature engineering

Some ideas for features

- BoW uses every word as a feature, but … shortcomings
- Alternatives
  - Remove stop words
  - Remove very rare words
  - Use bi-grams, tri-grams … (beware sentence breaks)
  - Perform part-of-speech tagging and keep only very and nouns
  - Perform shallow parsing and only keep noun phrases
  - Use noun phrases as additional features
  - Use different tokenizations at the same time
  - Use subword information
  - …

Feature selection (FS)

- Features may be redundant, correlated, irrelevant, …
- Many features bring much noise
  - Difficult to separate the signal from the noise
  - Most methods get slower with more features
- Traditional step in pre-processing: feature selection
  - Less noise
  - Smaller models, easier to understand, maybe even graphical
  - Faster classification

Types of FS types

- Find a subset of features by …

- Wrapper methods
  - Find the best set of features by trying many subsets in cross-validation
    - Usually requires an initialization and a search procedure
    - Very expensive

- Filter methods
  - Score each feature and remove the bad ones

- Embedded methods
  - Perform feature selection as part of model construction

Filter method: Mutual information (MI)

- Mutual information: How much does the presence of a feature tell me about the class of a document?

- For each feature e_t, compute

$$\sum_{e \in \{0,1\}} \sum_{c \in \{0,1\}} p(e,c) * \log\left(\frac{p(e,c)}{p(e) * p(c)}\right)$$

  - e: Feature present or not (for binary features)
  - c: The two classes (for binary classification)

- Keep only features with highest MI

Filter Method: Chi-Square ($\mathcal{X}^2$)

- Chi-Square: Which features are significantly more often in one class than expected?

- For each feature e_t, compute

$$\sum_{e \in \{0,1\}} \sum_{c \in \{0,1\}} \frac{(freq(e,c) - \exp(e,c))^2}{\exp(e,c)}$$

  - freq: Frequency of e in c
  - exp: Expected frequency of e in c assuming independence

- Keep only features with highest significance

Alternative: Feature extraction

- Derive a set of new features by dimensionality reduction methods
  - Find a low-dimensional representation such that … (for instance)
    - Principal component analysis (PCA): Variance in data is preserved
    - Multidimensional scaling: Distances between points are preserved
    - …

- Note: Many classifiers compute "new" features by combining existing ones
  - Linear classifiers: Linear combinations of features
  - ANN: Non-linear combinations

Thank you for your interest!

Literatur

[1] Jurish, Bryan, and Kay-Michael Würzner. "*Word and Sentence Tokenization with Hidden Markov Models.*" JLCL 28.2 (2013)

[2] Proisl, Thomas, and Peter Uhrig. "*SoMaJo: State-of-the-art tokenization for German web and social media texts.*" Proceedings of the 10th Web as Corpus Workshop. 2016.

[3] Devlin, Jacob, et al. "Bert: *Pre-training of deep bidirectional transformers for language understanding.*" arXiv preprint arXiv:1810.04805 (2018).

[4] Akbik, Alan, et al. *"FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP."* Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics. 2019.

Literatur

[5] Peters, Matthew E., et al. *"Deep contextualized word representations."* Proceedings of NAACL-HLT. 2018.

[6] Howard, Jeremy, and Sebastian Ruder. "*Universal Language Model Fine-tuning for Text Classification.*" Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics. 2018.

[7] Firth, J. R.. *"A synopsis of linguistic theory 1930-55.."* 1952-59 (1957): 1-32.

[8] Radford, Alec, et al. "*Language models are unsupervised multitask learners.*" OpenAI Blog 1.8, 2019