Seminar “Text Classification”

Text Preprocessing

Ulf Leser

Studentische Hilfskraftstelle, Projekt PREDICT

Am Lehrstuhl "Wissensmanagement in der Bioinformatik", Prof. Ulf Leser, am Institut für Informatik der Humboldt-Universität zu Berlin ist ab 1.6.2017 eine studentische Hilfskraftstelle (41h/Monat) im BMBF Verbundprojekt "PREDICT" für zu besetzen. In diesem Projekt erforschen wir gemeinsam mit KollegInnen der Charite Verfahren zur semantischen Integration von biomedizinischen Daten für die Behandlung von bestimmten Tumoren. Das Projekt spannt den Bogen von der Datenintegration bis zur konkreten klinischen Behandlungssituation

Aufgaben

- Aufbereitung von großen biomedizinischen Datensätzen
- Aufbereitung von biomedizinischen und medizinischen Textsammlungen
- Datenintegration, Erstellen von Skripts, Datenbankprogrammierung
- Mitarbeit an Forschungsprojekten

Voraussetzungen

- Studium der Informatik oder der Bioinformatik
- Erfahrung in der Programmierung (insb. Java, SQL, Scriptsprachen)
- Erfahrung im Umgang mit biomedizinischen Daten wären ein Vorteil
- Interesse an an der angewandten Forschung
- Ein hohes Maß an Eigenmotivation und Kommunikationsfähigkeit

Die Gruppe

Der Lehrstuhl arbeitet an verschiedenen Fragestellungen in Management und Analyse biomedizinischer Daten. Schwerpunkte sind graphbasierte und statistische Analyseverfahren von Hochdurchsatzdaten, biomedizinisches Text-Mining, und Scientific Workflows. Die Gruppe umfasst zur Zeit ca. 12 Doktoranden und Doktorandinnen, 5 Postdocs und 7 studentische Hilfskräfte, finanziert über eine Reihe interdisziplinärer Projekte vor allem mit der Charité. Weitere Informationen finden Sie unter hier.

Bewerbungen

Bitte schicken Sie Ihre Bewerbung mit den üblichen Unterlagen (Lebenslauf, Notenspiegel, ggf. Referenzen vorheriger Arbeitgeber) per Mail an Ulf Leser. Wir freuen uns auch über Nachfragen an dieselbe Adresse. Schwerbehinderte werden bei gleicher Eignung bevorzugt. Bewerbungen qualifizierter

Content of this Lecture

- Text Preprocessing
- Representing Text
- Feature Engineering
- Feature Selection

Definitions

- Definition
  - *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*
  - *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 will mostly make no difference between token and terms (sadly ...)

Logical View

- Definition
  - *The logical view of a document denotes its representation inside the system*
- Determines what algorithms can use for classification
  - Only metadata, only title, only abstract, full text, phrases, stop words, special characters, ...
- Creating the logical view involves transformations
  - Stemming, stop word removal
  - Transformation of special characters (Umlaute, Greek letters, ...)
  - Removal of formatting information (HTML), tags (XML), ...
  - Bag of words (BoW)

Special Characters

- Umlaute, Greek letters, math symbols, ...
- Often part of ASCII/Unicode, but systems don't like them
  - Small alphabets make indexing, searching, GUIs etc. much easier
- Different way of representation
  - XML/HTML: , ä, <
- Removing special chars makes querying them impossible
  - How to query for α, ∑, €, ?
- Options
  - Remove special characters
  - Transcribe: ü->ue, α-> alpha, ∀->for all, Σ->sum? sigma? ...
  - Work with large alphabets (Unicode)

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 tokenization, NER, ...
  - Loss of semantic info (proper names, Essen versus essen,...)
- Different impact in different languages (German / English)

Sentence Splitting

- Most linguistic analysis works on sentence level
- Sentences group together entities and statements
- Naive approach: Reg-Exp search for “[.?!;] “
  - (note the 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 ...”
  - Direct speech
    - “By saying “It is the economy, stupid!”, B. Clinton meant that ...”
  - ...

Algorithms

- Place putative sentence boundaries after all occurrences of . ? ! (and maybe ; : —)
- Move the boundary after following quotation marks, if any.
- Disqualify a period boundary in the following circumstances:
  - If it is preceded by a known abbreviation of a sort that does not normally occur sentence finally, but is commonly followed by a capitalized proper name, such as *Prof.* or *vs.*
  - If it is preceded by a known abbreviation and not followed by an uppercase word. This will deal correctly with most usages of abbreviations like *etc.* or *Jr.* which can occur sentence medially or finally.
- Disqualify a boundary with a ? or ! if:
  - It is followed by a lowercase letter (or a known name).
- Regard other putative sentence boundaries as sentence boundaries.

Figure 4.1 Heuristic sentence boundary detection algorithm.

Quelle: [MS99]

- Advanced approaches (Schmid (2000), Mikheev (1998))
  - Machine learning (classification of each ".") reaches 99.5% accuracy on brown corpus, but slow

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.”
  - „SQL commands comprise SELECT ... FROM ... WHERE clauses; the latter may contain functions such as leftstr(STRING, INT).“
  - “This LCD-TV-Screen cost 3,100.99 USD.”
  - “[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)

Stop Words

- Words that are so frequent that their removal (hopefully) does not change the meaning of a doc
  - 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 a positional index by ~40%
  - Hopefully increases precision due to less spurious hits
  - Makes many phrase queries impossible
- 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.

100 stop words

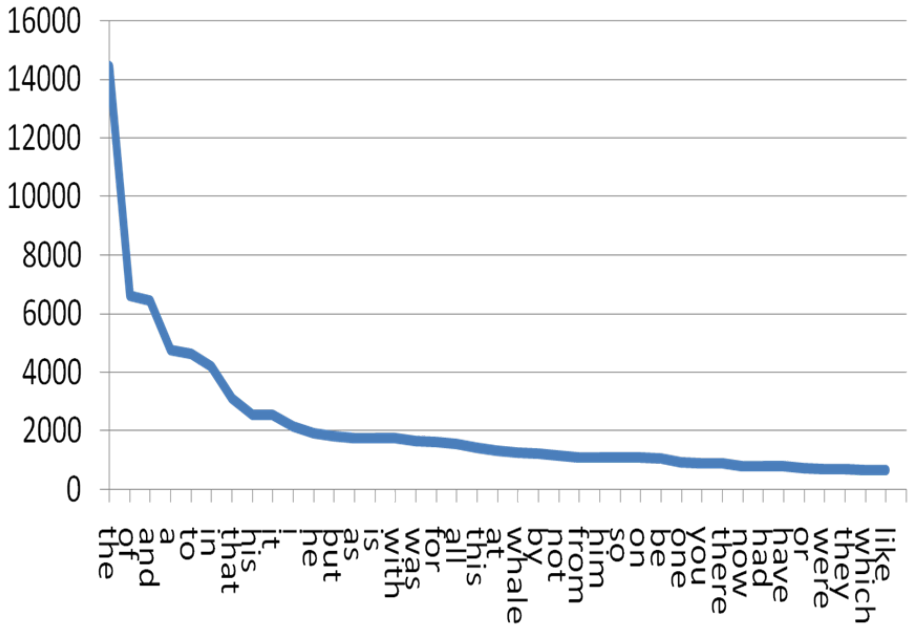
children obese overweight parents increased risk obesity. Subjects obese parents fatter childhood show stronger pattern tracking childhood adulthood. prevalence parental obesity increases general population extent child adult tracking BMI likely strengthen.

10 000 stop words

obese overweight obesity obese fatter adulthood prevalence parental obesity BMI

Zipf‘s Law

- Let f be the frequency of a word and r its rank in the list of all words sorted by frequency
- Zipf’s law: f ~ k/r for some constant k
- Example
  - Word ranks in Moby Dick
  - Good fit to Zipf’s law
  - Some domain-dependency (whale)
- Fairly good approximation for most corpora



Source: http://searchengineland.com/the-long-tail-of-search-12198

Content of this Lecture

- Text Preprocessing
- Representing Text
- Feature Engineering
- Feature Selection

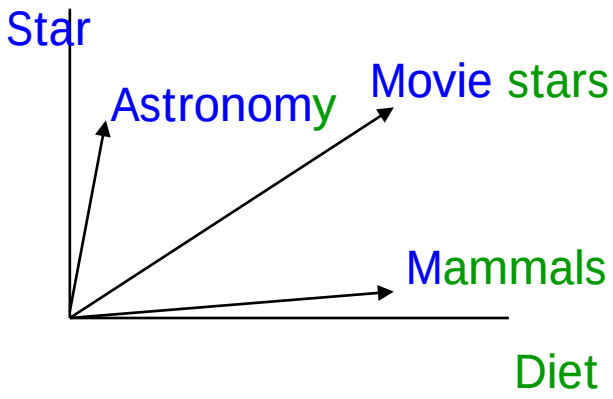
Notation

- "Bag of Words" view of a document
- Definition
  - *Let D be the set of all normalized documents, $d \in D$ is a document*
  - *Let K be the set of all terms in D, $k_i \in K$ is a term*
    - *Can as well be tokens*
  - *Let w be the function that maps a given d to its set of distinct terms in K (its bag-of-words)*
  - *Let v_d by a vector of size |K| for d (or a query q) with*
    - *$v_d[i]=0$ iff $k_i \notin w(d)$*
    - *$v_d[i]=1$ iff $k_i \in w(d)$*
  - *Often, we use weights instead of a Boolean membership function*
    - *Let $w_{ij} \geq 0$ be the weight of term k_i in document d_j ($w_{ij}=v_j[i]$)*
    - *$w_{ij}=0$ if $k_i \notin d_j$*

Vector Space Model

- Salton, G., Wong, A. and Yang, C. S. (1975). "A Vector Space Model for Automatic Indexing." *Communications of the ACM* **18**(11): 613-620.
  - A breakthrough in IR
  - Still most popular model today
- General idea
  - Fix vocabulary K (the dictionary)
  - View each doc (and the query) as point in a |K|-dimensional space
  - Rank docs according to distance from the query in that space
  - Here: Compare documents based their distance in BoW-space

Vector Space



- 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 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

$$|v| = \sqrt{\sum v_i^2} \qquad v \circ w = \sum_{i=1..n} v_i * w_i$$

Example

- Assume stop word removal, stemming, and binary weights

	Text	verkauf	haus	italien	gart	miet	blüh	woll
1	Wir verkaufen Häuser in Italien	1	1	1				
2	Häuser mit Gärten zu vermieten		1		1	1		
3	Häuser: In Italien, um Italien, um Italien herum		1	1				
4	Die italienschen Gärtner sind im Garten			1	1			
5	Der Garten in unserem italienschen Haus blüht		1	1	1		1	
Q	Wir wollen ein Haus mit Garten in Italien mieten		1	1	1	1		1

Ranking

$$sim(d,q) = \frac{\sum \left(v_q[i] * v_d[i] \right)}{\sqrt{\sum v_d[i]^2}}$$

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

- sim(d_1,q) = (1*0+1*1+1*1+0*1+0*1+0*0+0*1) / √3 ~ 1.15
- sim(d_2,q) = (1+1+1) / √3 ~ 1.73
- sim(d_3,q) = (1+1) / √2 ~ 1.41
- sim(d_4,q) = (1+1) / √2 ~ 1.41
- sim(d_5,q) = (1+1+1) / √4 ~ 1.5

Rg	Q: Wir wollen ein **Haus** mit **Garten** in **Italien mieten**
1	d_2: **Häuser** mit **Gärten** zu **vermieten**
2	d_5: Der **Garten** in unserem **italienschen Haus** blüht
3	d_4: Die **italienschen Gärtner** sind im **Garten**
	d_3: **Häuser**: In **Italien**, um **Italien**, um **Italien** herum
5	d_1: Wir verkaufen **Häuser** in **Italien**

Introducing Term Weights

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

Ranking with TF scoring

$$sim(d,q) = \frac{\sum \left(v_q[i] * v_d[i] \right)}{\sqrt{\sum v_d[i]^2}}$$

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

- sim(d_1,q) = (1*0+1*1+1*1+0*1+0*1+0*0+0*1) / √3 ~ 1.15
- sim(d_2,q) = (1+1+1) / √3 ~ 1.73
- sim(d_3,q) = (1+3) / √10 ~ 1.26
- sim(d_4,q) = (1+2) / √5 ~ 1.34
- sim(d_5,q) = (1+1+1) / √4 ~ 1.5

Rg	Q: Wir wollen ein **Haus** mit **Garten** in **Italien mieten**
1	d_2: **Häuser** mit **Gärten** zu **vermieten**
2	d_5: Der **Garten** in unserem **italienschen Haus** blüht
3	d_4: Die **italienschen Gärtner** sind im **Garten**
4	d_3: **Häuser**: In **Italien**, um **Italien**, um **Italien** herum
5	d_1: Wir verkaufen **Häuser** in **Italien**

Alternative Scoring: TF*IDF

- 1st problem: The longer a doc, the higher the probability of matching query terms by pure chance (it has more terms)
  - Solution: Normalize TF values on document length (yields $0 \leq w_{dk} \leq 1$)

$$tf'_{dk} = \frac{tf_{dk}}{|d|} = \frac{tf_{dk}}{\sum_{j=1..k} tf_{dj}}$$

  - Note: Longer docs also get down-ranked by normalization on doc-length in similarity function. Use only one measure!
- 2nd problem: Some terms are everywhere in D, don't help to discriminate, and should be scored less
  - Solution: Also use IDF scores

$$w_{dk} = \frac{tf_{dk}}{|d_d|} * idf_k$$

TF*IDF in Short

- Give terms in a doc 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
- Interferes with stop word removal
  - If stop words are removed, IDF might not be necessary any more
  - If IDF is used, stop word removal might not be necessary any more

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
  - Extension: Topic-based Vector Space Model
    - Latent Semantic Indexing (see IR lecture)

Content of this Lecture

- Text Preprocessing
- Representing Text
- Feature Engineering & Feature Selection

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
  - Word2Vec: Represent words as distributions
  - ...

Feature Selection

- 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 methods

- Find a subset of features by ...

- Wrapper methods
  - Find the best set of features by trying many subsets in CV
    - Usually requires an initialization and a search procedure
    - Very expensive
- Embedded methods
  - Perform feature selection as part of model construction
- Filter methods
  - Score each feature and remove the bad ones

Filter Method: Mutual Information

- 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

- 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

Unsupervised

- Unsupervised: Disregard distribution of feature values over classes
- Instead, consider (all) pairs of features to identify redundant ones
- Simple approach: Pearson correlation

$$\frac{\frac{1}{n-1}\sum_{i=1}^{n}\left(e_{t,i}-\bar{e_t}\right)*\left(e_{s,i}-\bar{e_s}\right)}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(e_{t,i}-\bar{e_t}\right)^2}*\sqrt{\frac{1}{n-1*}\sum_{i=1}^{n}\left(e_{s,i}-\bar{e_s}\right)^2}}$$

  - e_t, e_s are features, $\bar{e}$ is mean, n=|D|
  - When correlation is too high, remove one (which one?)

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: 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