max planck institut
informatik

Advanced Topics in Information Retrieval
Natural Language Processing for IR & IR Evaluation

Vinay Setty
vsetty@mpi-inf.mpg.de

Jannik Strötgen
jannik.stroetgen@mpi-inf.mpg.de

ATIR – April 28, 2016

Organizational Things

please register – if you haven't done so

- mail to **atir16 (at) mpi-inf.mpg.de**
- (i) name, (ii) matriculation number, (iii) preferred email address
- even if you do not want to get the ECTS points
- important for announcements about assignments, rooms etc.

assignments

- first assignment today
- **remember: we can only open pdfs**
- 50% of points (not of exercises) with serious, presentable

Outline

1. Simple Linguistic Preprocessing
2. Linguistics
3. Further Linguistic (Pre-)Processing
4. NLP Pipeline Architectures
5. Evaluation Measures

Why NLP Foundations for IR?

Why NLP Foundations for IR?

different types of data

- structured data vs. unstructured data (vs. semi-structured data)

structured data

- typically refers to information in tables

Employee	Manager	Salary
Johnny	Frank	50000
Jack	Johnny	60000
Jim	Johnny	50000

- numerical range and exact match (for text) queries, e.g.,
  Salary < 60000 AND Manager = Johnny

Why NLP Foundations for IR?

unstructured data

- typically refers to "free text"
- not just string matching queries

NLP foundations important for IR

typical distinction

- structured data → "databases"
- unstructured data → "information retrieval"

actually: **semi-structured data**

- almost always some structure: title, bullets
- facilitates semi-structured search

  title contains <u>NLP</u> and *bullet* contains <u>data</u>
- (not to mention the linguistic structure of text . . .)

Why NLP Foundations for IR?

standard procedure in IR

- starting point: documents and queries
- pre-processing of documents and queries typically includes
  - **tokenization** (e.g., splitting at white spaces and hyphens)
  - **stemming** or **lemmatization** (group variants of same word)
  - **stopword removal** (get rid of words with little information)
- this results in a bag (or sequence) of indexable terms

Documents & Queries

- Pre-processing of documents and queries typically includes
  - **tokenization** (e.g., splitting them up at white spaces and hyphens)
  - **stemming** or **lemmatization** (to group variants of the same word)
  - **stopword removal** (to get rid of words that bear little information)
- This results in a **bag (or sequence) of indexable terms**

More in next lecture

Investigators entered the company's HQ located in Boston MA on Thursday.	→	{ investig enter compani hq locat boston ma thursdai }

max planck institut informatik

24

Why NLP Foundations for IR?

standard procedure in IR

- starting point: documents and queries
- pre-processing of documents and queries typically includes
  - **tokenization** (e.g., splitting at white spaces and hyphens)
  - **stemming** or **lemmatization** (group variants of same word)
  - **stopword removal** (get rid of words with little information)
- this results in a bag (or sequence) of indexable terms

many NLP concepts mentioned in previous lecture

today: linguistic / NLP foundations for IR

Why NLP Foundations for IR?

goal of this lecture

NLP concepts are not just buzz words,
NLP concepts shall be understood

example:

what's the difference between lemmatization and stemming?

Contents

Tokenization

the task

- given a character sequence, split it into pieces called tokens

tokens are often loosely referred to as terms/words

- last lecture: "splitting at white spaces and hyphens"
- seems to be trivial

type vs. token (vs. term)

- **token**: instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit
- **type**: class of all tokens containing same character sequence
- **term**: (normalized) type included in IR system's dictionary

Tokenization – Example

type vs. token – example

- a rose is a rose is a rose
- how many tokens? 8
- how many types? 3 ({a, is, rose})

set-theoretical view

tokens → multiset
(multiset: *bag of words*)
types → set

type vs. token – example

- A rose is a rose is a rose
- knowing about normalization is important

Tokenization – Example

tokenization – example

- Mr. O'Neill thinks rumors about Chile's capital aren't amusing.

simple strategies

- split at white spaces and hyphens
- split on all non-alphanumeric characters
- mr | o | neill | thinks | rumors | about | chile | s | captial | aren | t | amusing

is that good? there are many alternatives

$\rightarrow$ o | neill – oneill – neill – o'neill – o' | neill

$\rightarrow$ aren | t – arent – are | n't – aren't

even simple (NLP) tasks not trivial!

most important

queries and **documents** have to be **preprocessed identically**!

Tokenization

queries and documents have to be preprocessed identically

- tokenization choices determine which (Boolean) queries match
- guarantees that sequence of characters in query matches the same sequence in text

further issues

- what about hyphens? co-education vs. drag-and-drop
- what about names? San Francisco, Los Angeles
- tokenization is language-specific
  - 这是几个单词序列 “this is a sequence of several words”
  - noun compounds are not separated in German: “Lebensversicherungsgesellschaftsangestellter” vs. “life insurance company employee”

compound splitter may improve IR

Lemmatization & Stemming

tokenization is just one step during preprocessing

- lemmatization
- stemming
- stopword removal

lemmatization and stemming

- two tasks, same goal

$\rightarrow$ to group variants of the same word

what's the difference?

stemming vs. lemmatization
stem vs. lemma

Lemma & Lemmatization

idea

- reduce inflectional forms (all variants of a “word”) to base form

examples

- am, are, be, is $\rightarrow$ be
- car, cars, car’s, cars’ $\rightarrow$ car

lemmatization

- proper reduction to dictionary headword form

lemma

- dictionary form of a set of words

Stem & Stemming

idea

- reduce terms to their "roots"

examples

- are → ar
- automate, automates, automatic, automation → automat

stemming

- suggests crude affix chopping

stem

- root form of a set of words (not necessarily a word itself)

Stemming and Lemmatization – Examples

the boy's cars are different colors

lemmatized

the | boy | car | be | different | color

stemmed

the | boy | car | ar | differ | color

Stemming and Lemmatization – Examples

for example compressed and compression are both accepted as equivalent to compress.

lemmatized

for | example | compress | and | compression | be | both | accept | as | equivalent | to | compress

stemmed

for | exampl | compress | and | compress | ar | both | accept | as | equival | to | compress

Stemming

popular stemmers

- porter's algorithm (`http://tartarus.org/martin/PorterStemmer/`)
- snowball (`http://snowballstem.org/demo.html`)

what's better for IR? stemming or lemmatization?

try it yourself!

Stop Words

stop words

- have little semantic content
- are extremely frequent: about 30% of postings top 30 words
- occur in almost each document, i.e., are not discriminative

$\rightarrow$ high document frequency

example of a stop word list

- a, an, and, are, as, at, be, by, for, from, has, he, in
- is, it, its, of, on, that, the, to, was, were, will, with

what types of words are these?

Stop Word Removal

idea

- based on stop list, remove all stop words, i.e., stop words are not part of IR system's dictionary
- saves a lot of memory
- makes query processing much faster

trend (in particular in web search):

- no stop word removal
- there are good compression techniques
- there are good query optimization techniques

stop words are needed – examples

- King of Norway
- let it be
- to be or not to be

Contents

Parts-of-Speech

alternative distinction between stop words and others

- function words: used to make sentences grammatically correct
- content words: carry the meaning of a sentence

function words

- auxiliary verbs
- prepositions
- conjunctions
- determiners
- pronouns

content words

- nouns
- verbs
- adjectives
- adverbs

how many parts-of-speech are there?

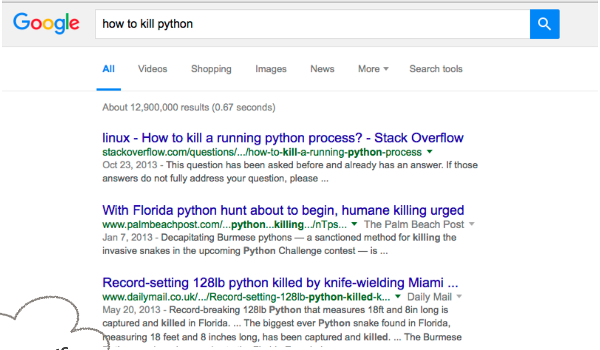
- between 8 and hundreds of different parts-of-speech
- what's useful depends on the application and language

Ambiguities

one word, one part-of-speech?

- can we can fish in a can?
- can: auxiliary, verb, noun

Information Need



max planck institut informatik

19

Levels of Ambiguities

speech recognition

- it's hard to recognize speech
- it's hard to wreck a nice beach

disambiguation

resolution of ambiguities

prepositional attachment

- the boy saw the man with the telescope

syntax / morphology

- time flies (noun / verb) like (verb / preposition) an arrow

word level ambiguities

- "can": auxiliary, verb, noun

word level ambiguities

most crucial for IR

Semantic Relations between Words

synonyms → query for one, find documents with either one

- different words, same meaning
- car vs. automotive

homographs → disambiguate or diversify results

- same spelling, different meaning
- bank vs. bank

homophons → problem with spoken queries

- same pronunciation, different meaning
- there vs. their vs. they're

homonyms

same spelling, same pronunciation, different meaning

Named Entities

entity

anything you can refer to with a name

- location, person, organization
- facilities, vehicles, songs, movies, products
  (and domain-dependent ones: genes & proteins, ...)
- sometimes: numbers, dates

relevant in IR

entities are popular and extremely frequent in queries

names are highly ambiguous

- Washington $\rightarrow$ place(s), person(s), (government)
- Springfield

Contents

Normalizations

indexed terms have to be normalized

- lemmatization
- stemming

some things need to be done before that:

- U.S.A. vs. USA
- anti-discriminatory vs. antidiscriminatory
- usa vs. USA

terms

- normalization results in *terms*
- a term is a normalized word type, an *entry in an IR system's dictionary*

Part-of-Speech Tagging

Zipf's law

$P_n \propto 1/n^a$

idea

- number of words in a language unlimited
  - few frequent words, many infrequent words
- number of parts-of-speech limited
  - Dionysios Thrax von Alexandria (100 BC): 8 parts-of-speech
  - in NLP: up to hundreds of part-of-speech tags (application- and language-dependent)
- many words are ambiguous

example

- The/*DET* newspaper/*NN* published/*VD* ten/*CD* articles/*NNS* ./*.*
- Can/*AUX* we/*PRP* can/*VB* fish/*NN* in/*IN* a/*DET* can/*NN* ./*.*

Part-of-Speech Tagging

part-of-speech tags

- allow for higher degree of abstraction to estimate likelihoods

what's the likelihood of:

- "an amazing" – is followed by "goalkeeper"
- "an amazing" – is followed by "scored"
- "determiner adjective" – is followed by "noun"
- "determiner adjective" – is followed by "verb"

automatic assignment of part-of-speech tags

- e.g., Penn Treebank tagset: 36 tags (+ 9 punctuation tags)
- ambiguities can be resolved via contexts

Part-of-Speech Tagging

way to go:

- input: sequence of (tokenized) words
- output: chain of tokens with their part-of-speech tags
- goal: most likely part-of-speech tags for the sequence
  $\rightarrow$ ambiguities shall be resolved
- a typical classification problem

is it tough?

- most words in English are not ambiguous
- most occurring words in English are ambiguous
- disambiguation is required

today's taggers

- about 97% accuracy (but highly domain-dependent)

Part-of-Speech Tagging

approaches

- rule-based taggers
- probabilistic taggers
- transformation-based taggers

probabilistic taggers

- given: manually annotated training data ("gold standard")
- learn probabilities based on training data
- estimate probabilities of pos tags given a word in a context
  $\rightarrow$ Hidden Markov Models

Part-of-Speech Tagging

Hidden Markov Models

- based on Bayesian inference
- goal: given sequence of tokens, assign sequence of pos tags
  given all possible tag sequences, which one is most likely?

$\hat{t}_1^n = \text{argmax } P(t_1^n | w_1^n)$

using Bayes, we get

$\hat{t}_1^n = \text{argmax } \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)} \rightarrow \hat{t}_1^n = \text{argmax } P(w_1^n|t_1^n)P(t_1^n)$

assumptions:

- probability of a word depends on own tag only
  $P(w_1^n|t_1^n) \approx \prod_{i=1}^n P(w_i|t_i)$
- probability of a tag depends on previous tag only
  $P(t_1^n) \approx \prod_{i=1}^n P(t_i|t_{i-1})$

Part-of-Speech Tagging

Hidden Markov Models

- based on Bayesian inference
- goal: given sequence of tokens, assign sequence of pos tags

  given all possible tag sequences, which one is most likely?

$$\hat{t}_1^n = \text{argmax } P(w_1^n|t_1^n)P(t_1^n) \approx \text{argmax } \prod_{i=1}^n P(w_i|t_i)P(t_i|t_{i-1})$$

maximum likelihood estimation based on a corpus

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1},t_i)}{C(t_{i-1})} \qquad P(w_i|t_i) = \frac{C(t_i,w_i)}{C(t_i)}$$

Part-of-Speech Tagging

in information retrieval

- determine content words in a query based on pos tags
- helpful for named entity recognition → semantic search

Chunking

(simple) grouping of token's that belong together

- most popular: noun phrase (NP) chunking
- but also: verb phrases

example

- [Paris $]_{NP}$ [has been $]_{VP}$ [a wonderful stop $]_{NP}$ during [my travel $]_{NP}$ – just as [New York City $]_{NP}$.

why chunking for IR?

- simpler than full syntactic analysis
- already provides some structure

Parsing

goal: syntactic structure of a sentence

two views of linguistic structure

- constituency (phrase) structure

example (man has the telescope)

- The boy saw the man with the telescope
- [[The boy $]_{NP}$
  [[saw $]_{VP}$ [[the man $]_{NP}$ [with [the telescope $]_{NP}$ $]_{PP}$ $]_{NP}$ $]_{VP}$
  $]_{S}$

Parsing

goal: syntactic structure of a sentence

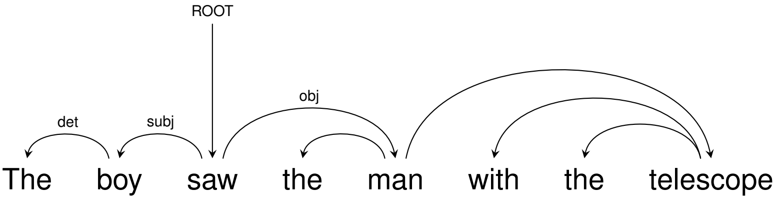
two views of linguistic structure

- constituency (phrase) structure
- dependency structure

example (man has the telescope)

- The boy saw the man with the telescope

helpful for IR?
relation extraction
for knowledge
harvesting



Named Entity Recognition

tasks

- extraction $\rightarrow$ determine the boundaries
- classification$\rightarrow$ assign class (PER, LOC, ORG, ...)

systems

- rule-based $\rightarrow$ with gazetteers, context-based rules (Mr.), ...
- machine learning $\rightarrow$ features: mixed case (eBay), ends in digit (A9), all caps (BMW), ...
- several tools available (e.g., Stanford NER)

extraction is good, but normalization is better

Named Entity Normalization

same task, many names

- normalization
- linking
- resolution
- grounding

example: Washington

- `/wiki/Washington,_D.C.`
- `/wiki/Washington_%28state%29`
- `/wiki/Washington_Irving`
- `/wiki/Washington_Redskins`
- `/wiki/George_Washington`

tools

- several tools available (AIDA, . . .)

Contents

NLP Pipeline Architectures

NLP tasks can often be split into multiple sub-tasks

- e.g., dependency parsing:
  - sentence splitting
  - tokenization
  - part-of-speech tagging
  - parsing

several pre-processing components in Elasticsearch

- pre-processing of corpora, e.g., for semantic search
- UIMA `https://uima.apache.org/`
- GATE `https://gate.ac.uk/`
- NLTK `http://www.nltk.org/`
- Stanford CoreNLP `http://stanfordnlp.github.io/CoreNLP/`

The Pipeline Principle – Why a (UIMA) Pipeline

... postponed to the information extraction lecture

Contents

Evaluation Measures

what is "good" / "correct" in information retrieval?

Evaluation Measures in NLP

let's start with a simple NLP task

example

- given a sequence of tokens, nouns

can	a	red	rose	be	a	tree	or	a	fly	or	just	a	rose
gold annotations													
can	a	red	**rose**	be	a	**tree**	or	a	**fly**	or	just	a	**rose**
example system output													
can	a	**red**	**rose**	be	a	**tree**	or	a	fly	or	just	a	**rose**

how good is the system's output?

Evaluation Measures in NLP

frequently used measures

- precision, recall, f-score
- based on evaluating all system's decisions

can	a	red	rose	be	a	tree	or	a	fly	or	just	a	rose
gold annotations													
can	a	red	**rose**	be	a	**tree**	or	a	**fly**	or	just	a	**rose**
example system output													
can	a	**red**	**rose**	be	a	**tree**	or	a	fly	or	just	a	**rose**

correct decisions: 3 + 8 = 11?

Evaluation Measures in NLP

frequently used measures

- precision, recall, f-score
- based on evaluating all system's decisions

can	a	red	rose	be	a	tree	or	a	fly	or	just	a	rose
gold annotations													
can	a	red	**rose**	be	a	**tree**	or	a	**fly**	or	just	a	**rose**
example system output													
can	a	**red**	**rose**	be	a	**tree**	or	a	fly	or	just	a	**rose**

we should count them separately

- true positives: 3
- true negatives: 8
- false positives: 2
- false negatives: 1

Evaluation Measures in NLP

confusion matrix

		ground truth: pos	ground truth: neg
system	pos	TP	FP
	neg	FN	TN

precision $= \frac{TP}{TP+FP}$ recall $= \frac{TP}{TP+FN}$ f_1-score $= \frac{2 \times \mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$

or in words

- precision: ratio of instances correctly marked as positive by the system to all instances marked as positive by the system
- recall: ratio of instances correctly marked as positive by the system to all instances marked as positive in the gold standard
- f1-score: balanced harmonic mean

Evaluation Measures in NLP

can	a	red	rose	be	a	tree	or	a	fly	or	just	a	rose

gold annotations

can	a	red	**rose**	be	a	**tree**	or	a	**fly**	or	just	a	**rose**

example system output

can	a	**red**	**rose**	be	a	**tree**	or	a	fly	or	just	a	**rose**

- true positives: 3
- true negatives: 8
- false positives: 2
- false negatives: 1

- precision = 3 / (3+2) = 0.6
- recall = 3 / (3+1) = 0.75
- f1-score = $(2 \times 0.6 \times 0.75) / (0.6 + 0.75) = 2/3$

Evaluation Measures in NLP

is precision then the accuracy?

accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$

in our example

- precision = 0.6
- accuracy = 0.78

what makes sense depends on the task

difference

- precision is about system's decisions about instances marked as positive in the gold standard
- accuracy is about correctness of all decisions

Evaluation Measures in IR

which of the measures make sense to evaluate IR:

precision, recall, f1-score, accuracy?

what's the goal of IR systems?

- is the information need satisfied?
- is the user happy?
- **happiness is elusive to measure**

what's an alternative?

- relevance of search results
- now: how to measure relevance?

Evaluation Measures in IR

measuring relevance with a benchmark

- a set of queries
- a document collection
- relevance judgments

TREC data sets are popular benchmarks

there are several issue, which we ignore (for now)

confusion matrix for IR

		manual judgments	
		relevant	not relevant
system	relevant	TP	FP
	not relevant	FN	TN

Evaluation Measures in IR

we can calculate

- precision
- recall
- f1-score
- accuracy

but are we done?

we need measures for ranked retrieval

short-comings

- only for binary judgments (relevant / not relevant)
- only for unranked results
- how do we get manual judgments for all documents?

Measures for Ranked Retrieval

precision at k

- set rank threshold k (e.g., 1, 3, 5, 10, 20, 50)
- compute percentage of relevant documents in k
- precision $= \frac{\text{“}TP\ in\ k\text{''}}{k}$
- ignores all documents ranked lower than k

example

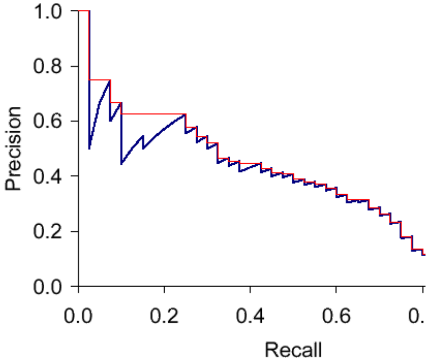
rank											precision @			
1	2	3	4	5	6	7	8	9	10	11	1	3	5	10
n	r	r	r	n	n	n	n	r	n	r	0	0.667	0.6	0.4

Measures for Ranked Retrieval

recall at k

- as precision at k

precision recall curve (http://nlp.stanford.edu/IR-book/html/htmledition/img532.png)



Measures for Ranked Retrieval

average precision

- precision at all ranks *r* with relevant document
- compute precision at k for each *r*
- (typically with cut-off, i.e., lower ranks not judged / considered)

example

rank											average precision
1	2	3	4	5	6	7	8	9	10	11	
n	r	r	r	n	n	n	n	r	n	r	$\frac{1/2+2/3+3/4+4/9+5/11}{5} = .56$

- compute: p@2, p@3, p@4, p9, p11
- number of relevant documents: 5

Measures for Ranked Retrieval

so far

- measures for single queries only

MAP is frequently reported in research papers

mean average precision

- sum of average precision divided by number of queries
- $MAP = \frac{\sum_{i=1}^{u} AP_i}{u}$

attention:

each query is worth the same!

example

- for query-1, $AP_1 = 0.62$
- for query-2, $AP_2 = 0.44$

assumption:

the more relevant documents, the better

- $MAP = \frac{AP_1 + AP_2}{2} = 0.53$

Beyond Binary Relevance

not realistic

- documents either relevant or not relevant (0 / 1)

much better

- highly relevant documents more useful
- lower ranks are less useful (likely to be ignored)

Beyond Binary Relevance

discounted cumulative gain

- graded relevance as measure of usefulness (*gain*)
- gain is accumulated,
  starting at the top,
  reduced (*discounted*) at lower ranks

discount rate

- typically used: 1/log (*rank*) (with base 2)

relevance judgments

- scale of [0,r], with $r > 2$

Beyond Binary Relevance

cumulative gain

- ratings of top n ranked documents $r_1, r_2, ...r_n$
- $CG = r_1 + r_2 + ... + r_n$

discounted cumulative gain

scores highly depend on judgments for queries

- at rank n
- $DCG = r_1 + \frac{r_2}{log_2(2)} + \frac{r_3}{log_2(3)} + ... + \frac{r_n}{log_2(n)}$

normalized discounted cumulative gain

- normalize DCG at rank n by DCG at n of ideal ranking
- ideal ranking of relevance scores: 3, 3, 3, 2, 2, 1, 1, 1, 0, 0, . . .

Beyond Binary Relevance

popular to evaluate Web search

- nDCG
- reciprocal rank: $rr = \frac{1}{K}$, with K rank of first relevant document
- mean reciprocal rank: mean rr over multiple queries
- exploiting click data (you need the data to do that . . .)

Summary

NLP 4 IR

- as text is not fully structured, plain keyword search not enough
- pre-processing documents and queries is important
- tokenization, stemming, lemmatization, stop word removal are frequently used

Ambiguities

- language is often ambiguous
- there are several levels of ambiguities

NLP tasks

- part-of-speech tagging helps to generalize
- named entities are important in IR

Summary

Evaluation Measures

- precision, recall, f1-score (in NLP)
- IR evaluation is different from NLP evaluation

Assignment 1

- the slides will help you a lot!

Thank you for your attention!

Thanks

some slides / examples are taken from / similar to those of:

- Klaus Berberich, Saarland University, previous ATIR lecture
- Manning, Raghavan, Schütze: Introduction to Information Retrieval (including slides to the book)
- Yannick Versley, Heidelberg University, Introduction to Computational Linguistics.