Data Science – Topic IENLP

Information Extraction and Natural Language Processing

Christin Seifert

February 5, 2019

University of Twente, Q3 2019/2020

Overview

1. Introduction
2. Regular Expressions
3. Pre-Processing
4. Text Classification
5. Evaluation
6. Further Topics
7. Tools and Software

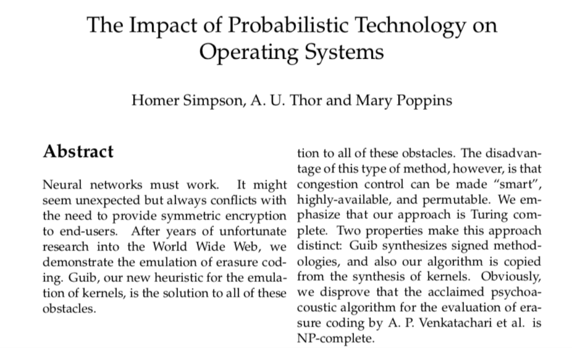
Introduction

Automatic Question Answering outperformed humans in 2011.

Figure 1: IBM Watson beats humans at Jeopardy (via Wikimedia commons)

Paper Generator

Automatically generated scientifically looking papers were accepted at a conference (as a way to identify conferences with low scientific standards) `https://pdos.csail.mit.edu/archive/scigen/`



The Impact of Probabilistic Technology on Operating Systems

Homer Simpson, A. U. Thor and Mary Poppins

Abstract

Neural networks must work. It might seem unexpected but always conflicts with the need to provide symmetric encryption to end-users. After years of unfortunate research into the World Wide Web, we demonstrate the emulation of erasure coding. Guib, our new heuristic for the emulation of kernels, is the solution to all of these obstacles.

tion to all of these obstacles. The disadvantage of this type of method, however, is that congestion control can be made "smart", highly-available, and permutable. We emphasize that our approach is Turing complete. Two properties make this approach distinct: Guib synthesizes signed methodologies, and also our algorithm is copied from the synthesis of kernels. Obviously, we disprove that the acclaimed psychoacoustic algorithm for the evaluation of erasure coding by A. P. Venkatachari et al. is NP-complete.

Figure 2: Automatically Generated Nonsense Paper

Regular Expressions

Regular Expressions

- Specific Letter: `[Aa]` matches either `A` or `a`.
- Range: `[A-Za-z0-9]` matches all alphabet in both cases and all numerals.
- Negation: `[^A-Z]` match all characters except capital letters.
- Disjunction: `red|green` matches either `red` or `green`
- Question mark (?): `colou?r` matches either `colour` or `color`.
- Asterisk(*): `wo*w`, zero or more occurrences of the previous character, matches `ww`, `wow`, `woow`, `wooooooooow`, etc.
- Plus(+): `wo+w`, one or more of the previous character, matches `wow`, `woow`, `wooow`, `wooooooooow`, etc.

Regular Expressions

- Dot(.): `end.`, matches exactly one character; matches `end.` or `end!` or `end?`, etc.
- Escape (\): escapes special characters. `end\` matches `end.`
- Carat (ˆ): represents the start of a string $\Rightarrow$ ˆ[A-Z] matches all strings that start with capital letter
- Word boundary (\b): `\b ha\b` matches `ha` but not `hahaha`.

Pre-Processing

Preprocessing of Text

Possible elements of text pre-processing

1. Remove layout elements
2. Detect the language of the text
3. Detect term (word) boundaries
4. Detect sentence boundaries
5. Remove stop words
6. Stemming and normalization

Pre-Processing

Example phrase: $\boxed{\text{to sleep perchance to dream}}$

Token Sequence of characters representing a useful semantic unit, instance of a type

5 tokens: $\boxed{\text{to}}$ $\boxed{\text{sleep}}$ $\boxed{\text{perchance}}$ $\boxed{\text{to}}$ $\boxed{\text{dream}}$

Type Common concept for tokens, element of the vocabulary (”something with a unique meaning in the real world”)

4 types: $\boxed{\text{sleep}}$ $\boxed{\text{perchance}}$ $\boxed{\text{to}}$ $\boxed{\text{dream}}$

Term Representation of a type that is stored in the dictionary of an index, might be a normalized version

if stopwords are not important for the task at hand we have 3 terms: $\boxed{\text{sleep}}$ $\boxed{\text{perchance}}$ $\boxed{\text{dream}}$

Prodecure:

1. Map all possible tokens to the corresponding type
2. Store all terms of the relevant type in the dictionary. That’s our **vocabulary**.

Example

Hell, if I could explain it to the average person, it wouldn't have been worth the Nobel prize.[1]

Tokens

`Hell` `if` `I` `could` `explain` `it` `to` `the` `average` `person` `it` `wouldn't` `have` `been` `worth` `the` `Novel prize`

Types (e.g.)

`Hell` `would` `it` `Novel prize` ...

- Count of terms as one or multiple tokens ("Novel prize") depends on the goal.
- Count of types (e.g, is "would" and "wouldn't" of same type) depends on the goal.

[1] Richard Feynman, People Magazine, 1985

Language Detection

- Later processing might depend on the language of the text. For instance stop-word lists are language-dependent.
- Simplestt method is based on **Letter Frequencies**

	Percentage of occurrence	
Letter	English	German
A	8.17	6.51
E	12.70	17.40
I	6.97	7.55
O	7.51	2.51
U	2.76	4.35

- Better—more elaborate—methods use statistics over more than one letter, e.g., statistics over two, three or even more consecutive letters (**N-Gram Frequencies**).

Tokenization

Goal

- Split text into tokens to identified the units later processing should work on (e.g. part-of-speech taggers consider one token at a time)

Simples method: tokenization using whitespaces as delimiters

- "San Francisco" $\rightarrow$?
- "I'm" and "I am"$\rightarrow$?
- "camel case" and "camel-case" and "camelcase" and "CamelCase"$\rightarrow$?
- "coarse-grained" and coarse grained $\rightarrow$?
- "kmh" and "km/h"$\rightarrow$?
- German compound nouns: e.g., *Donaudampfschiffahrtsgesellschaft*, meaning *Danube Steamboat Shipping Company*
- French articles: "L'atelier" $\rightarrow$? (should match un atelier)

Tokenization using Supervised Learning

- Train a model with annotated training data, use the trained model to tokenize unknown text
- Hidden-Markov-Models and conditional random fields are commonly used

Tokenization using dictionaries

- Build a dictionary (i.e., list) of tokens
- Go over the sentence and always take the longest fitting token (greedy algorithm!)
- Remark: Works well for Asian languages without white spaces and short words. Problematic for European languages

Naive approach

Every period and every “?” and “!” mark the end of a sentence

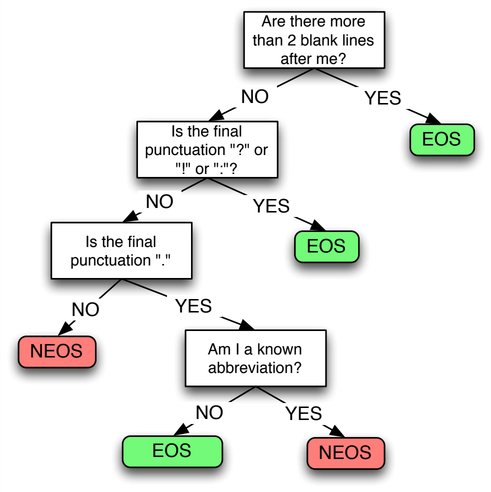
Problem

Periods also mark abbreviations, decimal points, email-addresses, e.g.,

- Dr. House is calling.
- The height is 0.14 inch.
- Mendeley Ldt. now is part of the Elsevier Corporation.

Solution

Build a binary classifier, deciding for each period whether it is the end of the sentence (EOS) or not (NEOS).



More Features:

- Is the next letter capitalized?
- Is the period surrounded by digits?
- Has the next word a high probability of occuring at the beginning of a sentence (e.g, Die, The)?

- Extremely common words that appear in nearly every text
- As stop words are so common, their occurrence does not characterize a text
- Just drop them

Stop word list Ignore word that are given on a list (black list), e.g., articles (*a*, *an*, *the*), conjunction (*and*, *or*, *but*, ...), pre- and postposition (*in*, *for*, *from*)

Problem Special names and phrases (*The Who*, *Let It Be*, ...)

Solution Make another list... (white list)

Normalization

Some tokens cary the same meaning for most applications, e.g.

- Search for `UK` should return documents with `U.K.`
- Search for `car` should return documents with `cars`[2]
- Search for `beauty` should return documents with `beautiful`

Possible steps

- Ignore cases (`UK` $\rightarrow$ `uk`)
- Stemming: removal of affixes (e.g., `beauty` $\rightarrow$ `beautiful` and `hammers` $\rightarrow$ `hammer`)
- Lemmatization: map inflections and variant forms to their base form
- Rule based removal of hyphens, periods, white spaces, accents, diacritics (Be aware of possible different meaning after removal)

[2] documents with the synonym `automobile` require different processing

Stemming & Lemmatization

Goal of both Reduce different grammatical forms of a word to their base form.

Stemming Find the word stem by chopping off/replacing last part of words (Example: Porter's algorithm).

Lemmatization Find the correct dictionary form.

Examples

- Map `do`, `doing`, `done` to common infinitive `do`
- Map `digitalizing`, `digitalized` to `digital`
- Map `master's`, `maters'`, `master` to `master`

Porter Stemmer

- Most common English stemmer
- Set of rules, applied in predefined sequence, for example

step	rule	example
1a	sses $\rightarrow$ ss	misses $\rightarrow$ miss
	ies $\rightarrow$ i	libraries $\rightarrow$ librari
	ss $\rightarrow$ ss	miss $\rightarrow$ miss
	s $\rightarrow$ $\emptyset$	houses $\rightarrow$ house
1b	(*vowel*)ing $\rightarrow$ $\emptyset$	dancing $\rightarrow$ danc
		king $\rightarrow$ king
	(*vowel*)ed $\rightarrow$ $\emptyset$	danced $\rightarrow$ danc
2	*for longer stems*	
	ational $\rightarrow$ ate	international $\rightarrow$ internate
	ator $\rightarrow$ ate	terminator $\rightarrow$ terminate
3	*for longer stems*	
	able $\rightarrow$ $\emptyset$	understandable $\rightarrow$ understand
	al $\rightarrow$ $\emptyset$	survival $\rightarrow$ surviv

Sample text: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Lovins stemmer: such an analys can reve featur that ar not eas vis from th vari in th individu gen and can lead to a pictur of expres that is mor biolog transpar and acces to interpres

Porter stemmer: such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret

Paice stemmer: such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret

Figure 3: Comparison of different stemming algorithms, examples from [2]

Text Classification

Document-Term-Matrices

- Example (from [1]): 9 documents (consider title only) in two categories “human-computer-interaction” (h) and “graphs” (g)

h1 `Human machine interface for Lab ABC computer applications`

h2 `A survey of user opinion of computer system response time`

h3 `The EPS user interface management system`

h4 `System and human system engineering testing of EPS`

h5 `Relation of user-perceived response time to error measurement`

g1 `The generation of random, binary, unordered trees`

g2 `The intersection graph of paths in trees`

g3 `Graph minors IV: Widths of trees and well-quasi-ordering`

g4 `Graph minors:  A survey`

Document-Term-Matrices

- Assume pre-processing (e.g., sentence splitting, stopword removal) has been done already
- Represent the documents as document-term-matrix (term-document-matrix) $\rightarrow$ Vectorspace Model
- Example: stopwords removed (“the”, “in”, “and”), for other terms: count how often they occur in each document

h1 Human machine interface for Lab ABC computer applications

	h1	h2	h3	h4	h5	g1	g2	g3	g4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Term Weighting

- TF (term frequency), i.e. word count, captures how important a word is for a document
- But some words are more frequent in general, TF-IDF weighting scheme aims to capture this (by downweighting those words)
  - tf_t^d – term frequency, number of times term t occurs in document d
  - N – total number of documents
  - df_t – document frequency, number of documents term d occurs in
  - idf_t – inverse document frequency

The TF-IDF measure for term t is defined as

$$\text{tf-idf}_t^d = \mathrm{tf}_t^d \cdot \mathrm{idf}_t \qquad \text{with } \mathrm{idf}_t = log\frac{N}{\mathrm{df}_t}$$

Bayes' Rule

Bayes's rule defines how conditional probabilities can be calculated from each other.

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{1}$$

with $P(A) \neq 0$.

Using notation from classification (C - class, D - data) it translates to

$$P(C|D) = \frac{P(D|C)P(C)}{P(D)}$$

Bayes' Rule for Classification

$$P(C|D) = \frac{P(D|C) \cdot P(C)}{P(D)}$$

- $P(C|D)$ – posterior probability, probability that class C occurs, given the data
- $P(D|C)$ – likelihood, how likely is it, that we observe the features in the class (given the class)
- $P(C)$ – prior/prior probability, how likely is the class (if we know nothing else?)
- $P(D)$ – is called the evidence.
- A Bayesian classification task should estimate $P(C|D)$ given a specific data sample
- In the data, we can observe (count/estimate) $P(D|C), P(C), P(D)$

Naive Bayes

- Usually, we have more than one feature in the data set D.
- For 2 features X_1, X_2 and a class C the joint probability can be written as follows (using the chain rule of probability)

$$p(X_1, X_2, C) = p(X_1|X_2, C) \cdot p(X_2|C) \cdot p(C)$$

- conditional probabilities like $p(X_1|X_2, C)$ are inefficient to estimate from data

$\rightarrow$ Naive Bayes' assumption: We assume that features are independent given the class

- Features are independent given the class
- Example: X_1 - color of shoes, X_2 - color of shirt, C - teacher or student
- We assume that in the class teacher (or student), the event of red shoes is independent of the event of red shirt (or blue shoes and red shirt ..). Meaning, that teachers don't pay attention to the color of their shirt when they decide which shoes to wear

$$X_1|c \not\perp\!\!\!\perp X_2|p(X_1|X_2, C) = p(X_1|C)$$

- This leads to

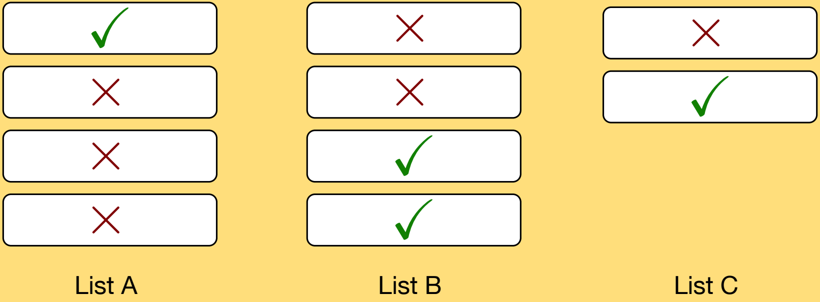
$$\begin{aligned} p(X_1, X_2, C) &= p(X_1|X_2, C) \cdot p(X_2|C) \cdot p(C) \\ &= p(X_1|C) \cdot p(X_2|C) \cdot p(C) \end{aligned}$$

where all terms can easily estimated from data

Evaluation

Small Example

Assume we have 3 retrieval algorithms A, B, and C. For a given query the return the following result lists. Which of the three algorithms returns better results (for this query)?



Ground Truth and Retrieval Function

Ground-truth indicates the (binary) relevancy of a document for a query.

Query	Document	Relevancy
q_1	D_1	1
q_1	D_5	1
q_2	D_7	1
	$\dots$	

Retrieval function returns a ranked list of documents for a query.

$$\rho : Q \times D \rightarrow \mathcal{R}$$

Types of Documents

For one specific query, documents can either be relevant or not relevant (according to the ground truth). And they can either be retrieved by the retrieval algorithm or not.

not relevant	relevant

not retrieved	retrieved

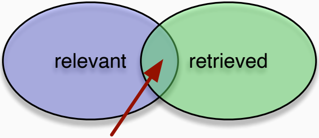
In the ideal world..

1. We want all relevant documents to be retrieved.
2. We want all *not* relevant documents *not* to be retrieved. Or: We want *only* relevant documents to be retrieved.

In other words: We want high recall (1.) and high precision (2.)

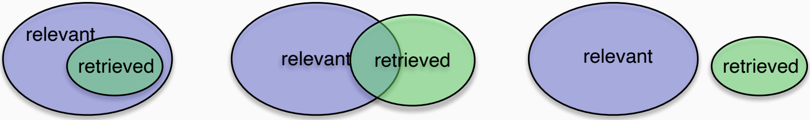
Precision

Precision π: The fraction of relevant documents in the result set.



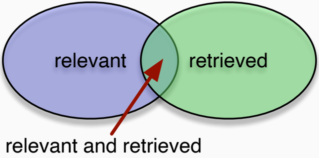
$$\pi = \frac{|\text{relevant and retrieved}|}{|\text{retrieved}|} = \frac{|\text{retrieved}| \cap |\text{relevant}|}{|\text{retrieved} \ |}$$

Precision is maximal if the result list only contains relevant documents, and minimal if the result list contains no relevant documents.



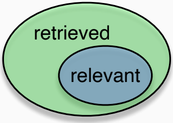
max precision medium precision min precision

Recall ρ: The fraction of relevant documents retrieved.

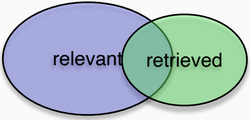


$$\rho = \frac{|\text{relevant and retrieved}|}{|\text{relevant}|}$$
$$= \frac{|\text{retrieved}| \cap |\text{relevant}|}{|\text{relevant}|}$$

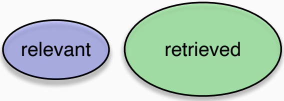
Recall is maximal if the result list contains all relevant documents, and minimal if the result list contains no relevant documents.



max recall



medium recall



min recall

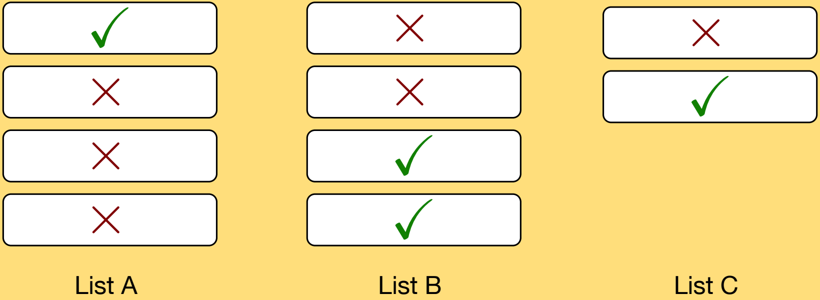
Confusion Matrix

		Retrieved / Prediction: Yes	Retrieved / Prediction: No
Truth	Relevant/Yes	TP	FN
Truth	Not Relevant/No	FP	TN

- Recall is also called *True Positive Rate*, *Hit Rate* or *Sensitivity*
- Precision is the also called the *Positive Predictive Value*

Quick Quiz

What are precision and recall for the 3 different retrieval algorithms below? Assume that we have 2 relevant documents in total.



$$\pi_A = \frac{1}{4}, \rho_A = \frac{1}{2}, \quad \pi_B = \frac{1}{2}, \rho_B = 1, \quad \pi_C = \frac{1}{2}, \rho_C = \frac{1}{2}$$

Which one should we select?

F-MEASURE

F_1**-measure**: harmonic mean of precision and recall

$$F_1 = 2\frac{\pi\rho}{\pi + \rho}$$

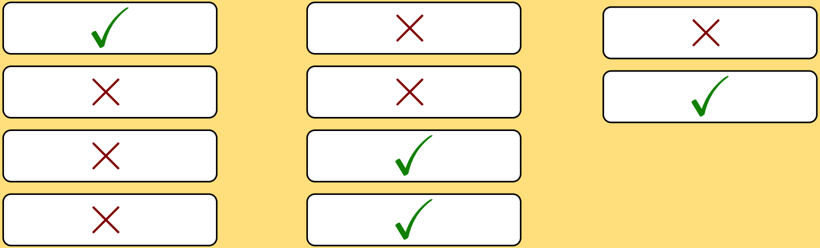
F_β-measure: generalisation allowing to emphasize either precision or recall (recall is β times more important)

$$F_\beta = (1 + \beta^2)\frac{\pi\rho}{\beta^2\pi + \rho}$$

π	ρ	F_1	$F_{0.5}$	F_2
1	0.1	0.18	0.12	0.36
0.1	1	0.18	0.36	0.12
0	1	0.00	0.00	0.00
1	0	0.00	0.00	0.00
0.7	0.3	0.42	0.34	0.55
0.3	0.7	0.42	0.55	0.34
0.94	0.96	0.95	0.96	0.94

Quick Quiz

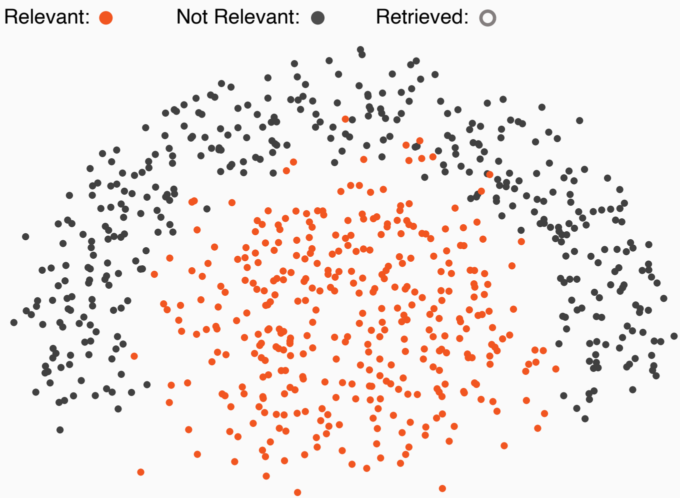
What is the F1 measure for the 3 different retrieval algorithms below? Assume that we have 2 relevant documents in total.



List A | List B | List C

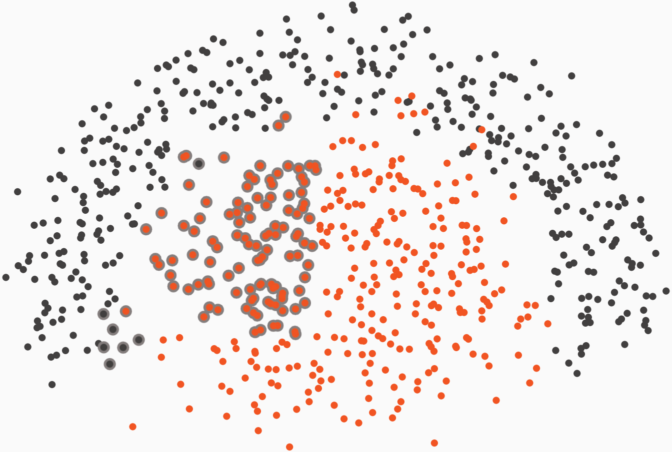
$$\pi_A = \frac{1}{4}, \rho_A = \frac{1}{2}, F_{1_A} = 0.33$$
$$\pi_B = \frac{1}{2}, \rho_B = 1, F_{1_B} = 0.67$$
$$\pi_C = \frac{1}{2}, \rho_C = \frac{1}{2}, F_{1_A} = 0.50$$

Precision, Recall, F1



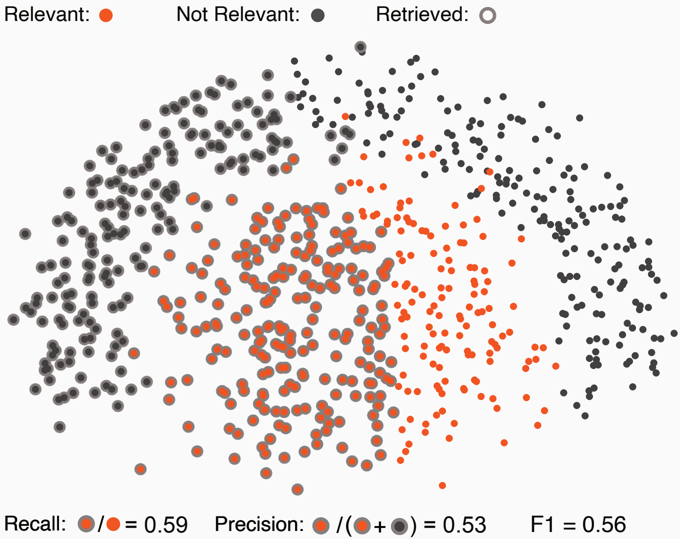
Precision, Recall, F1

Relevant: Not Relevant: Retrieved:

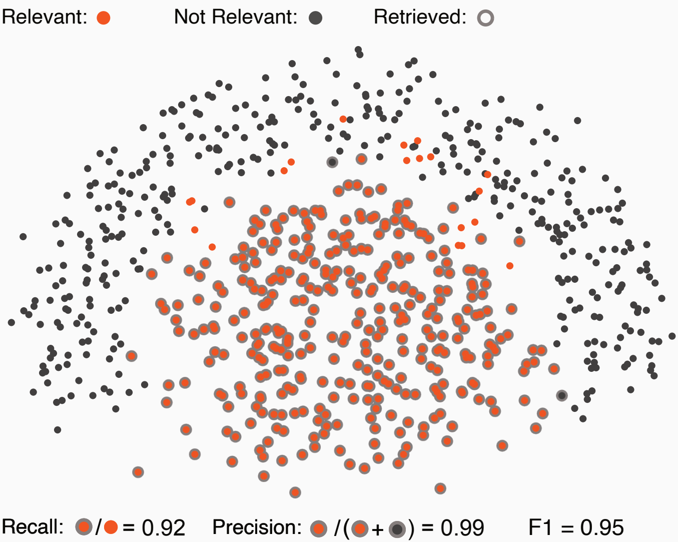


Recall: / = 0.26 Precision: /(+) = 0.94 F1 = 0.40

Precision, Recall, F1

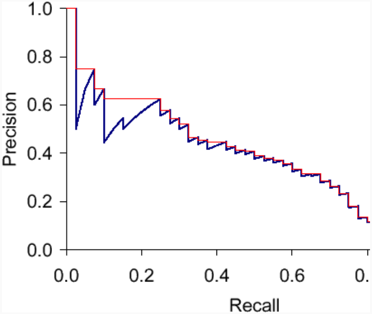


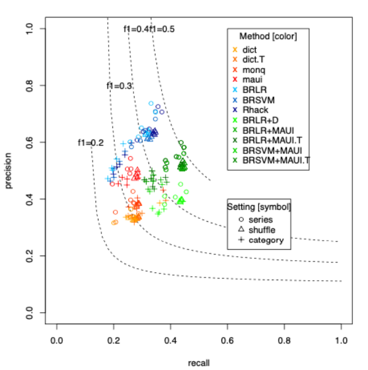
Precision, Recall, F1



Precision and Recall Tradeofff

In practice there is a tradeoff between precision and recall





Further Topics

Named Entity Recognition

Augusta Ada King, Countess of Lovelace (née Byron; 10 December 1815 – 27 November 1852) was an English mathematician and writer, chiefly known for her work on Charles Babbage's proposed mechanical general-purpose computer, the Analytical Engine.

Potential tags:

- LOCATION
- ORGANIZATION
- DATE
- MONEY
- PERSON
- PERCENT
- TIME

Figure 4: Named Entity Recognition with Stanford NER

`http://nlp.stanford.edu:8080/ner/process`

Entity Disambiguation

Linking mentions to knowledge bases

First documented in the 13th century, Berlin was the capital of the Kingdom of Prussia (1701–1918), the German Empire (1871–1918), the Weimar Republic (1919–33) and the Third Reich (1933–45). Berlin in the 1920s was the third largest municipality in the world. After World War II, the city became divided into East Berlin -- the capital of East Germany -- and West Berlin, a West German exclave surrounded by the Berlin Wall from 1961–89. Following German reunification in 1990, the city regained its status as the capital of Germany, hosting 147 foreign embassies.

Napoleon [Napoleon] was the emperor of the First French Empire. He was defeated at **Waterloo** [Battle of Waterloo] by **Wellington** [Arthur Wellesley, 1st Duke of Wellington] and **Blücher** [Gebhard Leberecht von Blücher]. He was banned to **Saint Helena** [Saint Helena], died of stomach cancer, and was buried at Invalides.

Figure 5: Disambiguation Examples by DBpedia Spotlight `https://www.dbpedia-spotlight.org/demo/` and AIDA `https://gate.d5.mpi-inf.mpg.de/webaida/`

Extraction of relations between objects

To give an example of Relation Extraction, if we are trying to find a birth date in:

> *"John von Neumann (December 28, 1903 – February 8, 1957) was a Hungarian and American pure and applied mathematician, physicist, inventor and polymath."*

then IEPY's task is to identify "`John von Neumann`" and "`December 28, 1903`" as the subject and object entities of the "`was born in`" relation.

Figure 6: Relation Extraction example from PyPi

https://pypi.org/project/iepy/

Annotate tokens with lexical function

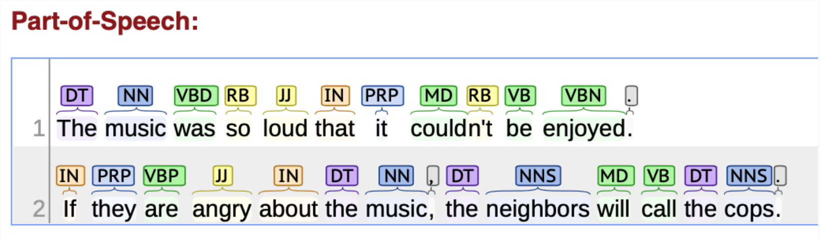


Figure 7: Part-of-Speech Tagging with Standford NLP

`http://nlp.stanford.edu:8080/corenlp/process`

Co-reference Resolution

Resolve words that reference another

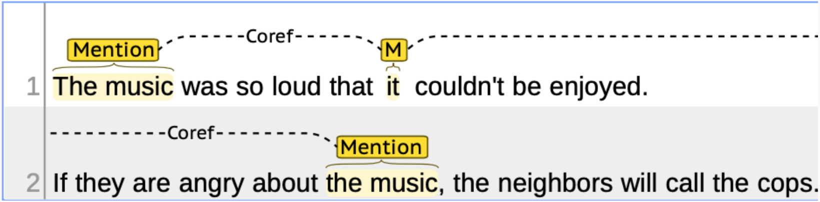


Figure 8: Coreference Resolution with Standford NLP

`http://nlp.stanford.edu:8080/corenlp/process`

Language Models, Language Generation

The Impact of Probabilistic Technology on Operating Systems

Homer Simpson, A. U. Thor and Mary Poppins

Abstract

Neural networks must work. It might seem unexpected but always conflicts with the need to provide symmetric encryption to end-users. After years of unfortunate research into the World Wide Web, we demonstrate the emulation of erasure coding. Guib, our new heuristic for the emulation of kernels, is the solution to all of these obstacles.

tion to all of these obstacles. The disadvantage of this type of method, however, is that congestion control can be made “smart”, highly-available, and permutable. We emphasize that our approach is Turing complete. Two properties make this approach distinct: Guib synthesizes signed methodologies, and also our algorithm is copied from the synthesis of kernels. Obviously, we disprove that the acclaimed psychoacoustic algorithm for the evaluation of erasure coding by A. P. Venkatachari et al. is NP-complete.

Figure 9: Automatically Generated Nonsense Paper

Tools and Software

Example Notebooks (link on canvas)

- Regular Expressions, Text Preprocessing
- Named Entity Recognition, Text Classification

Tools

- Regular Expressions Online Test `http://regexpal.com/`
- Open Source NLP Toolkit SpaCy (python, with models for Dutch) `https://spacy.io`
- OpenNLP from the Apache Project (Java) `https://opennlp.apache.org`
- Gate language toolkit (Java) `https://gate.ac.uk`
- Stanford NLP `http://nlp.stanford.edu:8080/corenlp/process`

- Some slides (precision/recall) are adapted from the course on Machine Learning and Data Mining at the Bauhaus University Weimar (Prof. Dr. Benno Stein).
  - Overview: `webis.de/lecturenotes/overview/overview.html`
  - Slides: `webis.de/lecturenotes/slides/slides.html`
- Roman Kern, Technical University Graz (some preprocessing slides)
- Regular Expression Examples from a slide set from Mena Habib, University of Twente

Further Readings

- Introduction to Information Retrieval
  Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008. ISBN: 0521865719, 9780521865715. URL: https://nlp.stanford.edu/IR-book/

References

[1] Scott Deerwester et al. "Indexing by latent semantic analysis". In: *Journal of the Society for Information Science* 41.6 (1990), pp. 391–407. URL: http://lsi.research.telcordia.com/lsi/LSIpapers.html.

[2] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008. ISBN: 0521865719, 9780521865715. URL: https://nlp.stanford.edu/IR-book/.