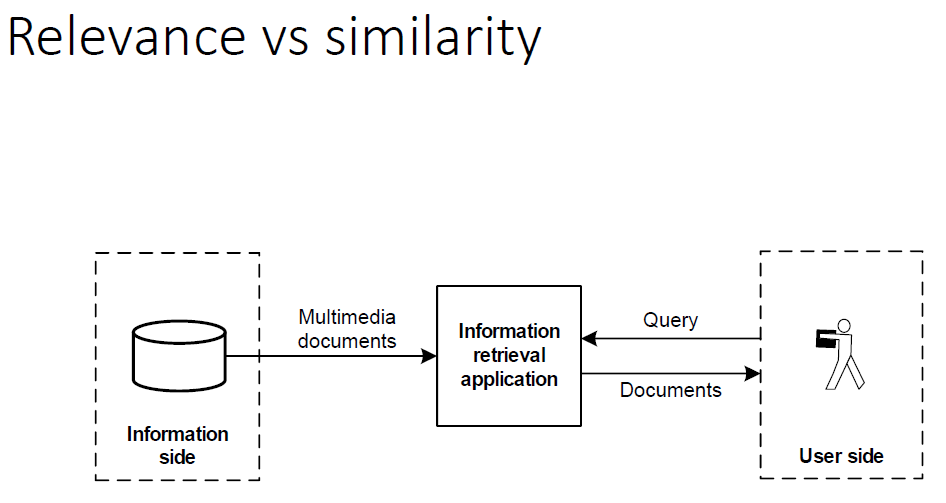
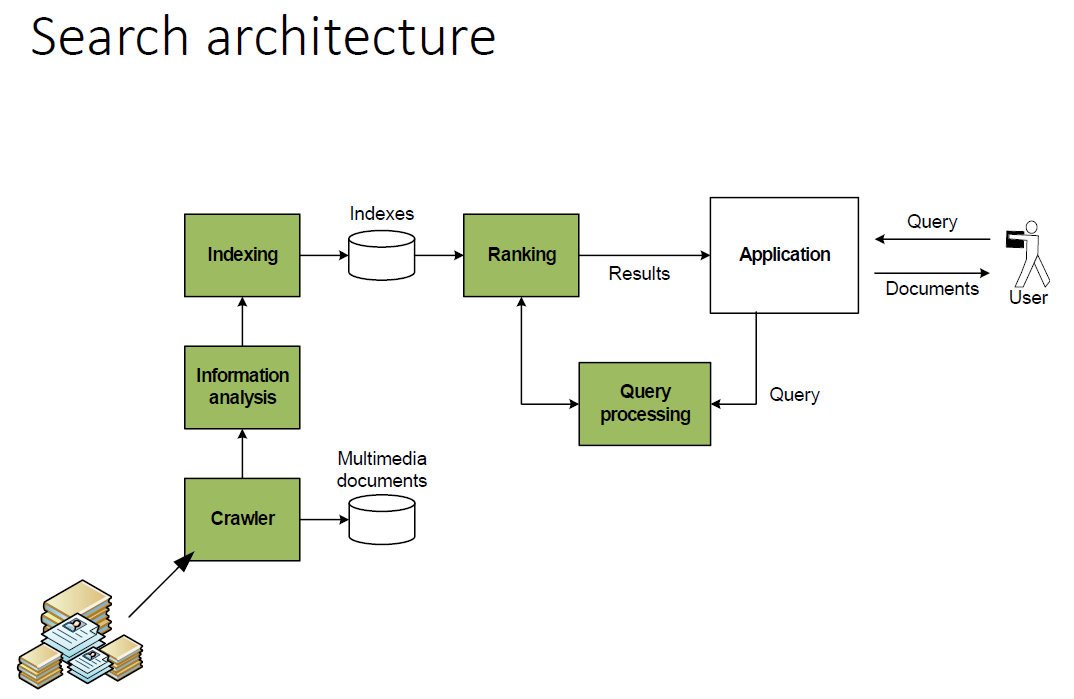
**Recuperação de Informação**





# Search architecture components

1. Crawl data for storage
2. Analyze documents and compute meaningful representations of natural language
3. Store data in an efficient manner
4. Process user information needs
5. Find answer to user request

Text processing  
**Words and Corpus, text tokenization, stemming, lemmatizing, PoS and NE**

## Natural language parsing

* Word tokenization and sentence delimitation
* Part of speech tagging
* Word sense disambiguation (Terms weighting, Vector space models, language models, word embeddings)
* Named entities (Linking and relation)
* Subjective attributes (Sentiment, emotion, sarcasm, politeness, …)

### **Basic text processing**

# Words and corpora

**Lemma** => cat and cats are the same lemma

**Type** => an element of the vocabulary.

**Token** => an instance of that type in running text.

**Vocabulary** => set of types

##### Corpora

A text is produced by:

1. a specific writer(s),
2. at a specific time,
3. in a specific variety,
4. of a specific language,
5. for a specific function.

# Word tokenization

##### Text Normalization

Every NLP task requires text normalization:

1. Tokenizing (segmenting) words
2. Normalizing word formats
3. Segmenting sentences

##### Space-based tokenization

A very simple way to tokenize:

* For languages that use space characters between words (Arabic, Cyrillic, Greek, Latin, etc., based writing systems)
* Segment off a token between instances of spaces

##### Scikit-learn Tokenization

O feito nas aulas práticas

##### Issues in Tokenization

Can't just blindly remove punctuation:

* m.p.h., Ph.D., AT&T, cap’n
* prices ($45.55)
* dates (01/02/06)
* URLs (http://www.stanford.edu)
* hashtags (#nlproc)
* email addresses ([someone@cs.colorado.edu](mailto:someone@cs.colorado.edu))

**Clitic**: a word that doesn't stand on its own

"are" in we're, French "je" in j'ai, "le" in l'honneur

When should multiword expressions (MWE) be words? (New York, rock ’n’ roll)

# Data-driven Word Tokenization (BPE)

Instead of:

* white-space segmentation
* single-character segmentation

Use the **data** to tell us how to tokenize.

##### Subword tokenization

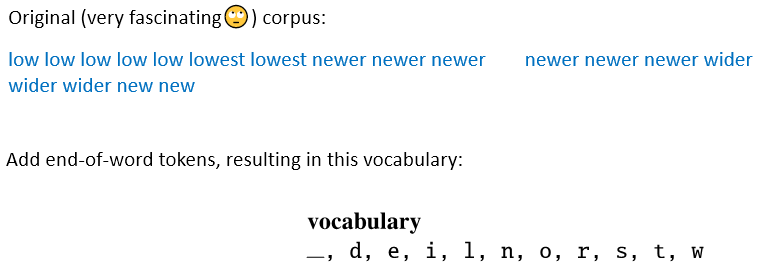
Three common algorithms:

* Byte-Pair Encoding (BPE) (Sennrich et al., 2016)
* Unigram language modeling tokenization (Kudo, 2018)
* WordPiece (Schuster and Nakajima, 2012)

All have 2 parts:

1. A token learner that takes a raw training corpus and induces a vocabulary (a set of tokens).
2. A token segmenter that takes a raw test sentence and tokenizes it according to that vocabulary

##### BPE token learner



##### Properties of BPE tokens

Usually include frequent words

And frequent subwords:

* which are often morphemes like -est or –er

A morpheme is the smallest meaning-bearing unit of a language:

* unlikeliest has 3 morphemes un-, likely, and -est

# Word Normalization and other issues

##### Character processing and stop-words

1. Numbers/dates
2. Acronyms
3. Multi-language documents
4. Stop-words - remove words that are present in all documents: a, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will…

##### Word Normalization

Putting words/tokens in a standard format

U.S.A. or USA / uhhuh or uh-huh / Fed or fed / am, is, be, are

##### Lemmatization

Represent all words as their lemma, their shared root = dictionary headword form:

* am, are, is  be
* car, cars, car's, cars'  car
* Spanish quiero (‘I want’), quieres (‘you want’)
* He is reading detective stories 🡪He be read detective story

**Lemmatization is done by Morphological Parsing**

* Morphemes:
  + The small meaningful units that make up words
  + Stems: The core meaning-bearing units
  + Affixes: Parts that adhere to stems, often with grammatical functions
* Morphological Parsers:
  + Parse cats into two morphemes cat and s

##### Stemming

Reduce terms to stems, chopping off affixes crudely

**Porter Stemmer**

Based on a series of rewrite rules run in series

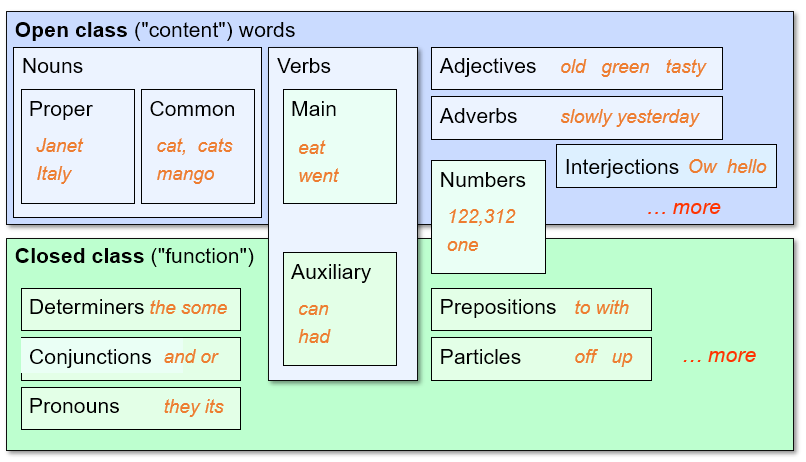
* A cascade, in which output of each pass fed to next pass

##### Sentence Segmentation

The sinal !, ? mostly unambiguous but period “.” is very ambiguous

# Part of Speech tagging

##### Two classes of words: Open vs. Closed



##### Why Part of Speech Tagging?

Can be useful for other NLP tasks

* Parsing: POS tagging can improve syntactic parsing
* MT: reordering of adjectives and nouns (say from Spanish to English)
* Sentiment or affective tasks: may want to distinguish adjectives or other POS
* Text-to-speech (how do we pronounce “lead” or "object"?)

Or linguistic or language-analytic computational tasks

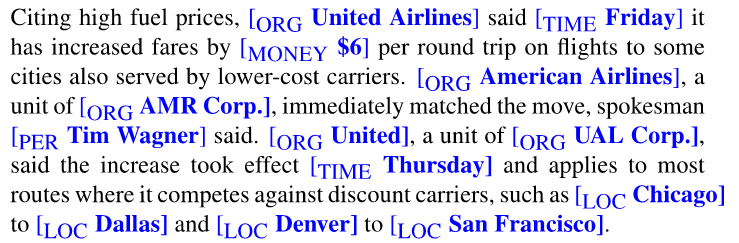
* Need to control for POS when studying linguistic change like creation of new words, or meaning shift
* Or control for POS in measuring meaning similarity or difference

##### Named Entity tagging

The task of named entity recognition (NER):

* find spans of text that constitute proper names
* tag the type of the entity.

**Output:**



**Sentiment analysis:** consumer’s sentiment toward a particular company or person?

**Question Answering:** answer questions about an entity?

**Information Extraction:** Extracting facts about entities from text.

### Vector Space Model and Language Models

**VSM, LM Jelinek Mercer Smoothing and LM Dirichlet Smoothing**

# Retrieval models

Geometric/linear spaces

* Vector space model

Language models approach to IR

* Language models and smoothing

Probabilistic retrieval model

* Binary independence model
* Okapi’s BM25

# Term weighting

Boolean retrieval looks for terms overlap

It doesn’t consider:

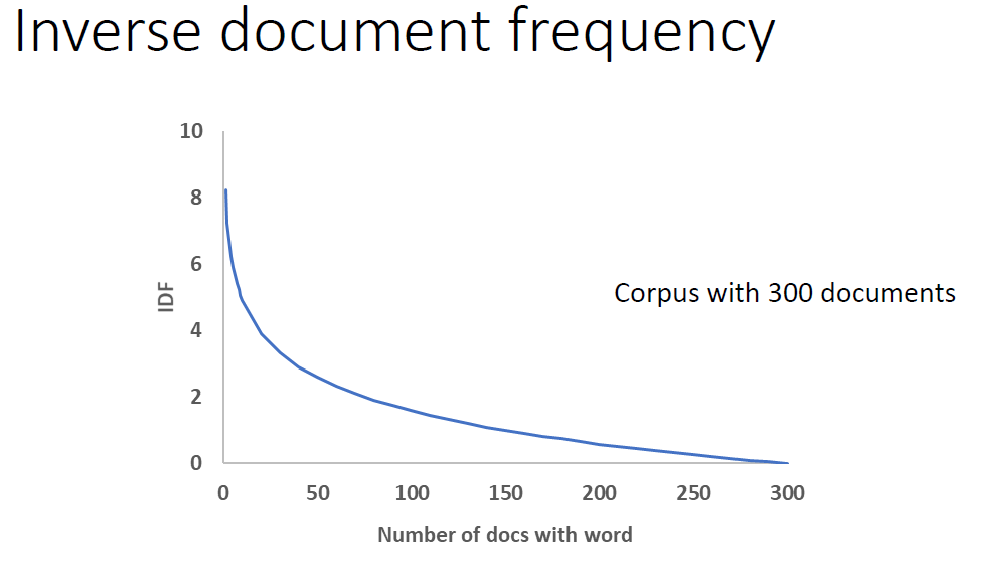
* Term frequency in document
* Term scarcity in collection (document mention frequency) :
  + *of* is more common than ideas or march
* Length of documents
  + (And queries: score not normalized)

Term weighting tries to reflect the importance of a document for a given query term. Term weighting must consider two aspects:

1. The frequency of a term in a document
2. Consider the rarity of a term in the repository

**Text terms should be weighted according to: (final -> tf idf)**

* their importance for a given document
* and how rare a word is



# Vector Space Model

Each doc d can now be viewed as a vector of tf-idf values, one component for each term

So, we have a vector space where:

* terms are axes
* docs live in this space
* even with stemming, it may have 50,000+ dimensions

# Documents and queries

* Documents are represented as an histogram of terms, n-grams and other indicators
* The text query is processed with the same text pre processing techniques.
  + A query is then represented as a vector of text terms and n-grams

# Intuition

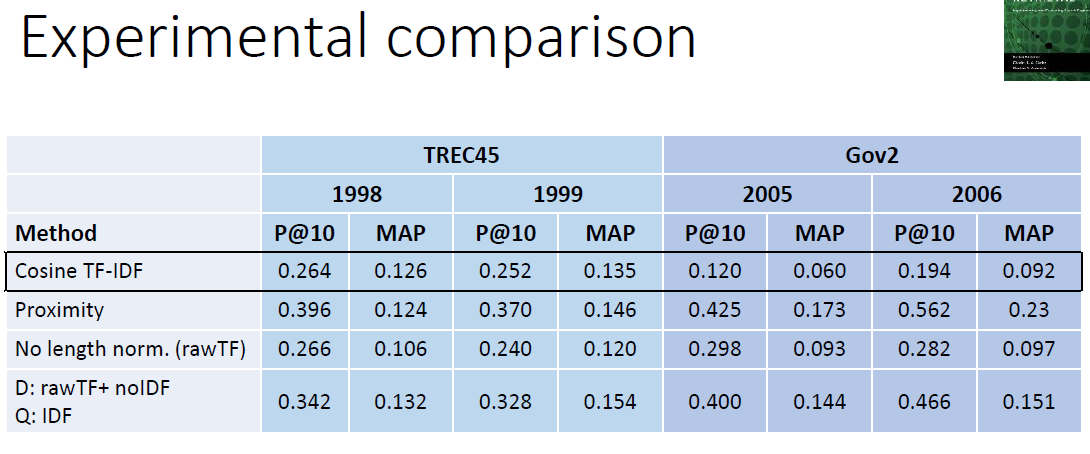
# First cut

# Angle as a similarity

# Vectors normalization

# Cosine similarity

# Improvedsemantics

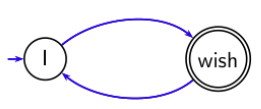


# Probability Ranking Principle

* We have a collection of documents
* User issues a query
* A list of documents needs to be returned
* Ranking method is the core of an IR system

# Language Models

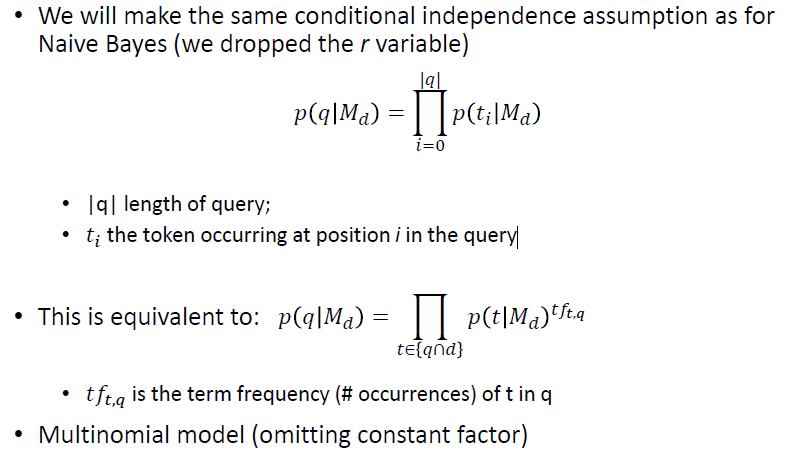
We can view a finite state automaton as a deterministic language model.



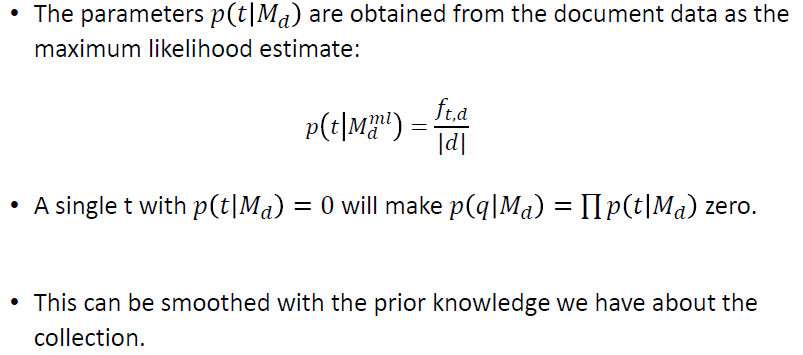
##### Document models 𝑀𝑑

* Count the number of word occurrences
* Divide by the length of the document

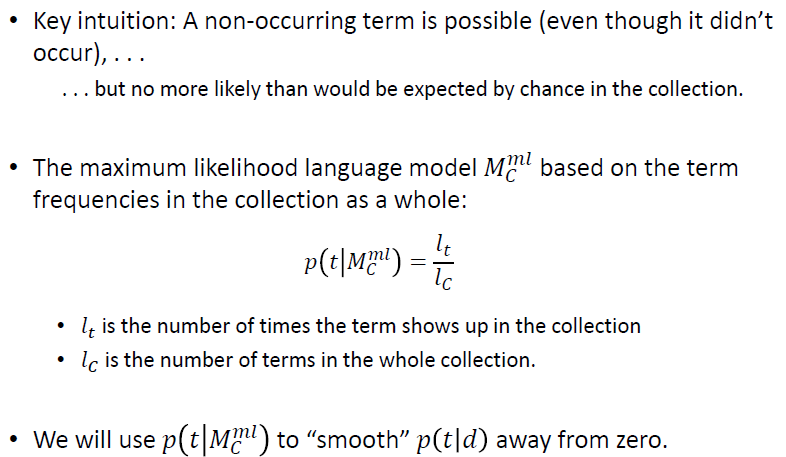
##### How to compute 𝑝𝑞𝑑



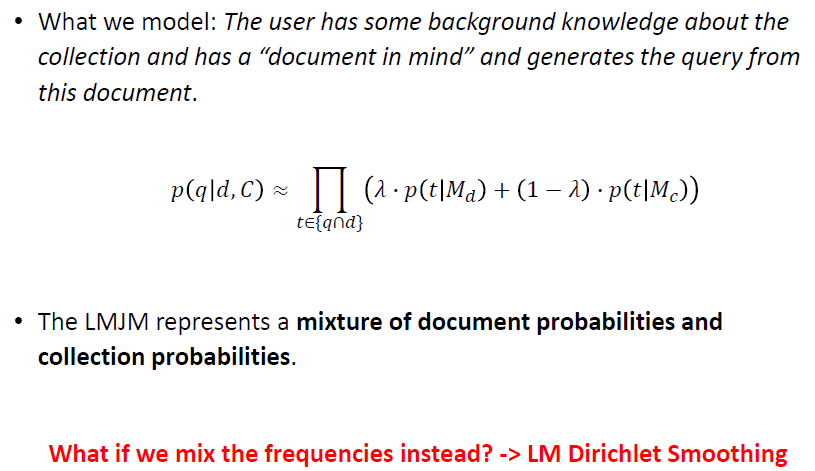
##### Parameter estimation



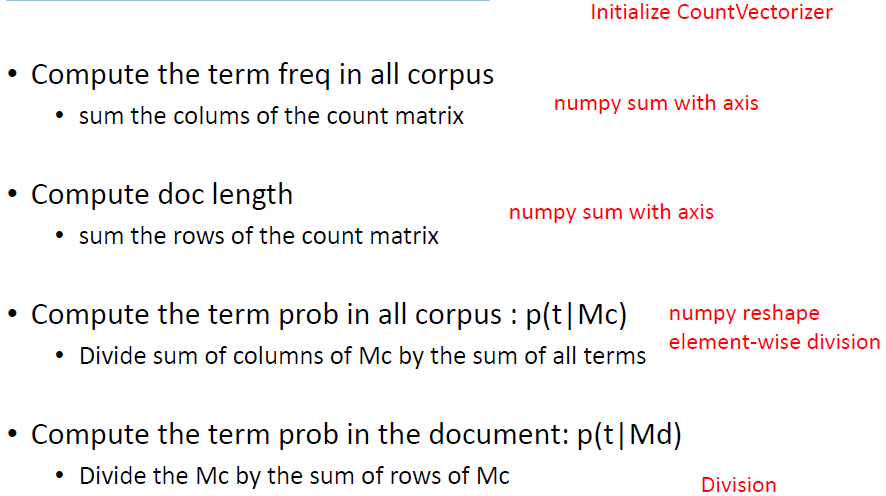
##### Smoothing



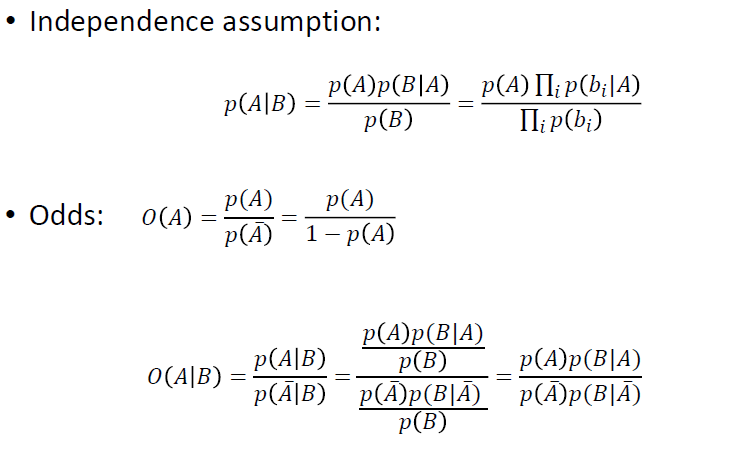
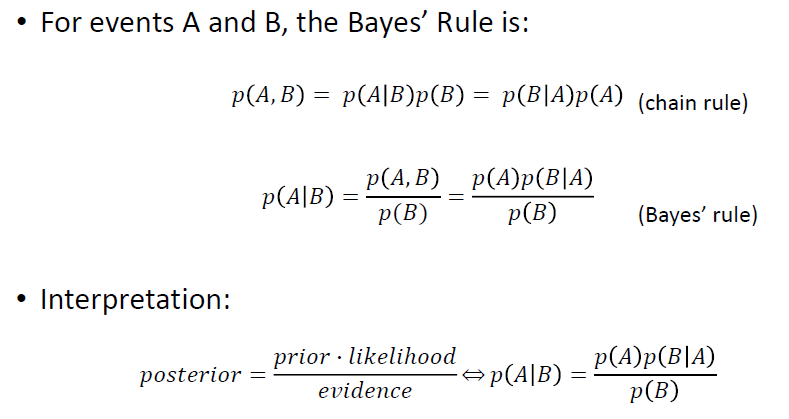
##### Mixture model: Summary



# LMJM implementation



# Recap of basic probability



### Evaluation

**Experimental protocols, datasets, metrics**

# Reproducible experimentation

1. **Experimental protocol**

* Is the task/problem clear? Is it a standard task
* Detailed description of the experimental setup: identify all steps of the experiments.

1. **Reference dataset**

* Use a well known dataset if possible.
  + If not, how was the data obtained?
* Clear separation between training and test set.

1. **Evaluation metrics**

* Prefer the commonly used metrics by the community.
* Check which statistical test is most adequate.

# Reference datasets

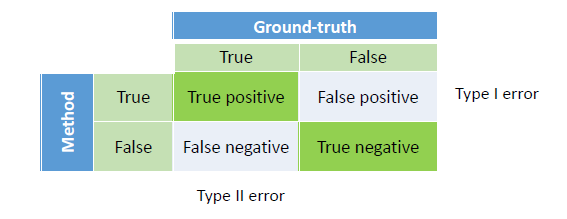
A reference dataset is made of:

* a collection of documents
* a set of training data
* a set of test data
* the relevance judgments or groundtruth

# Typesof evaluation

* With groundtruth
* A /B testing
* A combination of the two

# Groundtruth



* Judgments can be obtained by experts or by crowdsourcing
* Human relevance judgments can be incorrect and inconsistent

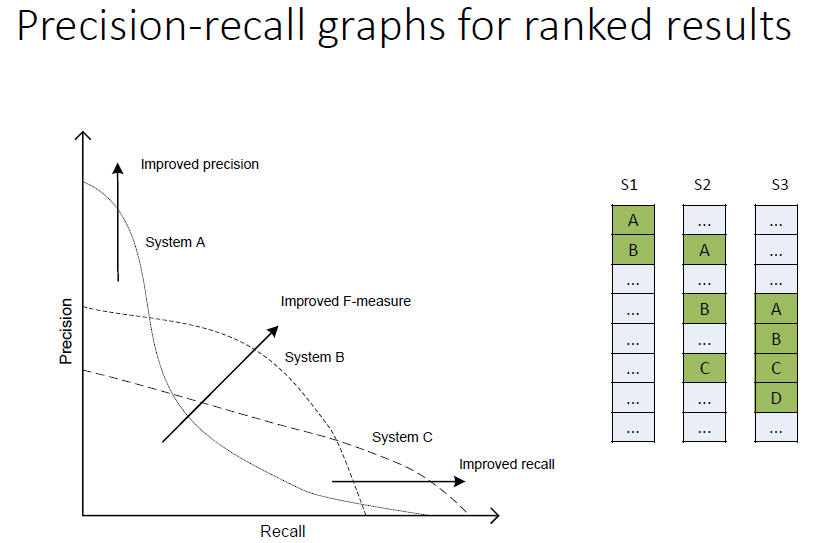
# Evaluation metrics

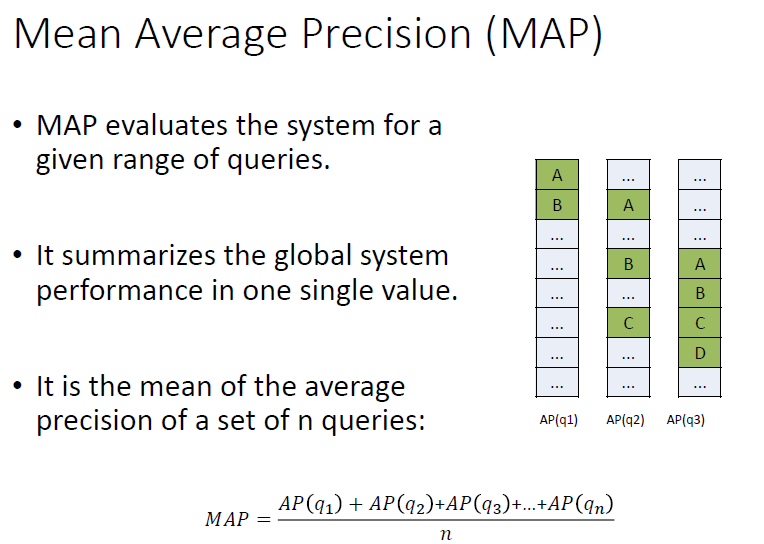
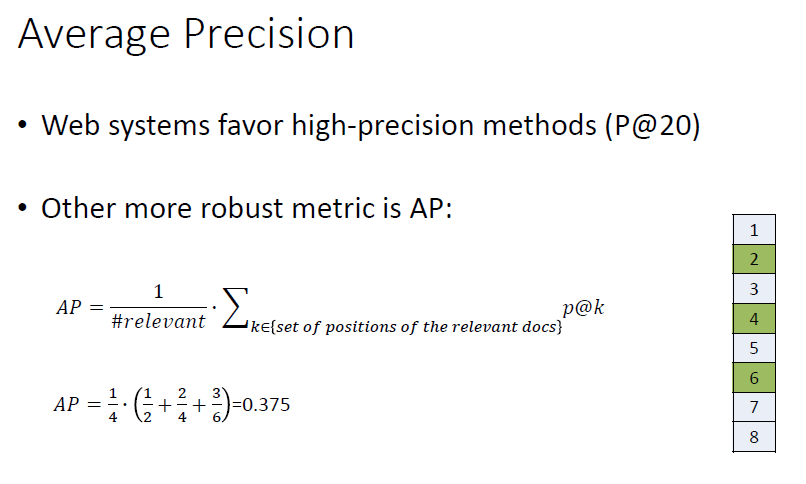
Utility metrics are focused in evaluating the results that are presented to the user

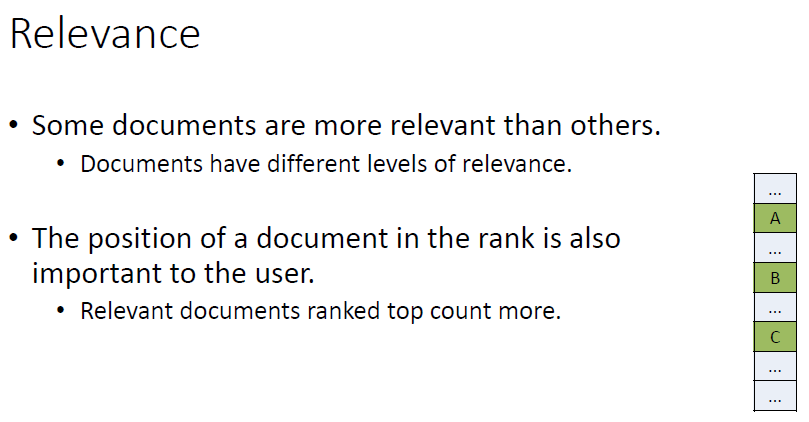
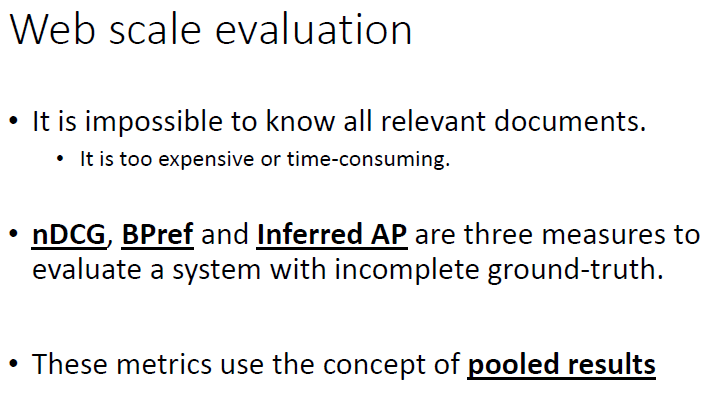
* Usually, this is done with relevance judgments on the top results
* Common metrics for binary relevance judgments: Top Precision and Recall
* Common metrics for binary relevance judgments : NDCG

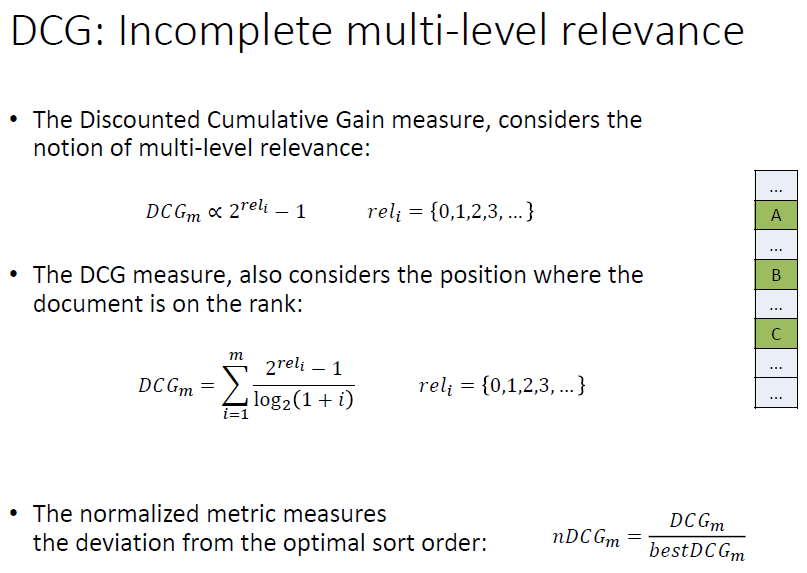
Stability metrics are focused in evaluating the robustness of the system results.

* Usually, this is done with binary relevance judgments across a wide range of data
* Common metrics: MAP, AP, Precision Recall curves

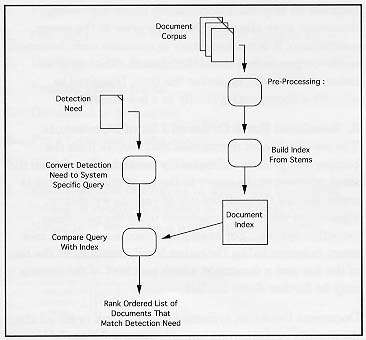








### Pipeline (project)



## 1. Document Corpus

* A corpus is comprised of the source documents from which the user will select the document sub-set. The content of the corpus may have significant the performance in some applications.

## 2. Pre-Processing of Document Corpus

* Pre-processing of the corpus is an area of continuing research and is a key discriminator in Document Detection methods.
* Most systems use some method of stemming. Stemming is the reduction of a word to its root. For example, "Contrary, Contradiction, Contraband" all have 'contra' as a stem which permits a certain amount of generalization over the meaning of the sentences or document.
* Most systems also use a list of Stop words. A Stop word is a word which is usually ignored, e.g., 'a', 'an', 'the'. Research is progressing at identifying phrases and multi-term items such as dates and personal names so that these can be indexed as single terms.

## 3. Building Index from Stems

* This function is frequently very system specific because it is a key place in the detection system for optimizing run-time performance. Document Detection systems in general are concerned with speed due to the very large number of the documents to be searched.
* It may take a very long time to build the index for a large corpus. New indexes may only be built weekly or even less often. Some provision is always made for incrementally indexing documents that have been added to the corpus since the last full indexing.

## 4. Document Index

* A document index is essentially a list of terms, stems, phrases, etc. (depending upon the search algorithm) with each term having an associated list of document identifiers which point to documents and stem locations that contain the particular item. Further information resulting from analyses of the frequency of terms in the document and corpus and of the co-occurrence of terms within the corpus may also be stored in the index to aid in the ranking of documents in the returned document set.
* Frequently the index may be as large as the original document corpus and various design and compression techniques are usually used to condense it.

## 5. Detection Need

* A Detection Need expresses the user's criteria for a relevant document. A Detection Need may take a number of forms described at the beginning of this paper.

## 6. Convert Detection Need to System Specific Query

* When being processed the Detection Need is transformed in two stages: it is first transformed into a detection query, and then into a retrieval query. Some information in the Detection Need, such as keywords, may not require transformation.
* Detection Needs are independent of the specific retrieval engine employed, while detection queries and retrieval queries are specific to a particular retrieval engine. By de-coupling the Detection Need and the system specific query, detection systems can more easily be ported to different domains and employ different indexing algorithms. This allows a more consistent interface with the user.
* The detection query is specific to the retrieval engine but independent of the corpus over which retrieval is to be performed. The retrieval query is specific to the retrieval engine, to the operation, and to the corpus. The retrieval query may incorporate term weights based on the inverse document frequencies in a collection.
* The interpretation, translation and processing of a Detection Need is also performance sensitive part of the Detection application. Again, because of the large number of documents against which it will be compared.
* Research is progressing on the use of phrase lists and term expansion when determining system specific queries. Certain words or phrases are replaced with more informative words which are determined from the document corpus itself. Abbreviations are frequently expanded to their full meaning.

## 7. Compare Query with Index

* In attempting to select a desired document the query is compared item by item with the corpus index, recognizing any imbedded logic, such as include - do not include.
* It is not necessary to examine each document in the corpus since it's important constituent items were placed in the index prior to the query comparison. The use of an index significant improves the time, typically a few seconds, required to identify a document.

## 8. Resultant Rank Ordered List of Documents

* The list of relevant documents that results from the comparison process is ranked ordered from the most relevant to the query to the least relevant. This is accomplished through the use of various weighting algorithms which are dependent upon the particular detection system. For example, a document, which met every criterion in the Detection Need, would be at the top of the list and a document which met 90% of the criteria may be further down the list.
* Document Detection systems typically rank order all the documents in the corpus but only return the top 'N' documents depending upon the desired cut-off specified.

### Dicionário

**CountVectorizer** - usado para transformar um determinado texto em um vetor com base na frequência (contagem) de cada palavra que ocorre em todo o texto

**Fit\_transform** – É usado nos training data para que possamos dimensionar os dados de treinamento e também aprender os parâmetros de dimensionamento. Aqui, o modelo construído aprenderá a média e a variância das características do conjunto de treinamento. Esses parâmetros aprendidos são usados para dimensionar nossos dados de teste