THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Introduction

Instructor:
Walid Magdy

22-Sep-2021

1

Lecture Objectives

- Know about the course:
  - Topic
  - Objectives
  - Requirements
  - Format
  - Logistics
- Note:
  - No much technical content today
  - Don't assume next lectures would be the same!

Text Technologies for Data Science

= documents, words, terms, …

≠ images, videos, music *(with no text)*

Information Retrieval

Text Classification

Text Analytics

Search Engines Technologies

What is Information Retrieval (IR)?

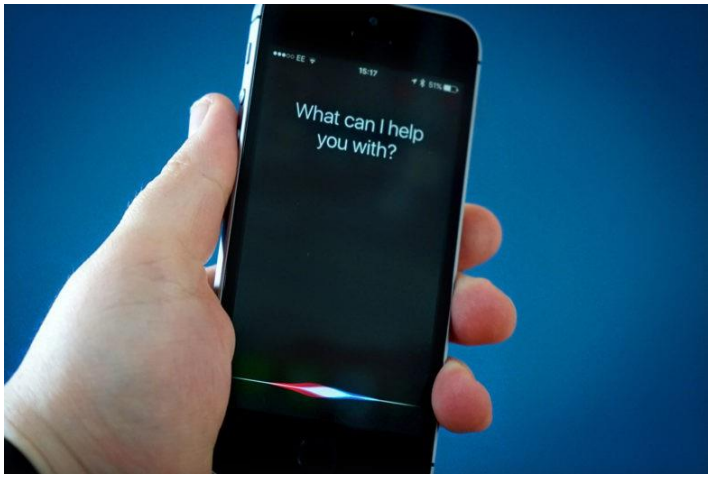
IR is NOT just

Google

Google Search I'm Feeling Lucky

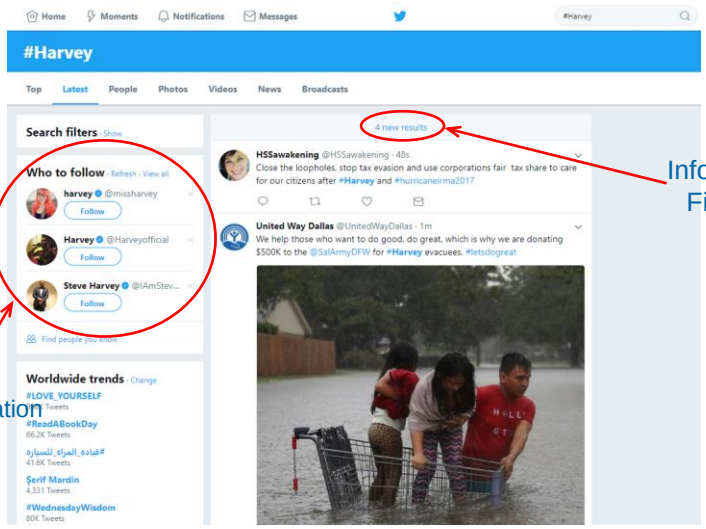
Web search

What is IR?



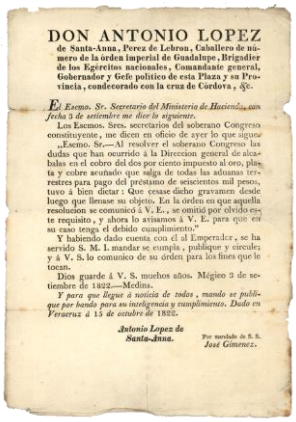
Speech - QA

What is IR?



Social search

What is IR?



DON ANTONIO LOPEZ
de Santa-Anna, Perez de Lebron, Caballero de número de la órden imperial de Guadalupe, Brigadier de los Egércitos nacionales, Comandante general, Gobernador y Gefe político de esta Plaza y su Provincia, condecorado con la cruz de Córdova, &c.

El Escmo. Sr. Secretario del Ministerio de Hacienda, con fecha 3 de setiembre me dice lo siguiente.

Los Escmos. Sres. secretarios del soberano Congreso constituyente, me dicen en oficio de ayer lo que sigue:

„Escmo. Sr.—Al resolver el soberano Congreso las dudas que han ocurrido á la Direccion general de alcabalas en el cobro del dos por ciento impuesto al oro, plata y cobre acuñado que saiga de todas las aduanas terrestres para pago del préstamo de seiscientos mil pesos, tuvo á bien dictar: Que cesase dicho gravamen desde luego que llenase su objeto. En la órden en que aquella resolucion se comunicó á V. E., se omitió por olvido este requisito, y ahora lo avisamos á V. E. para que en su caso tenga el debido cumplimiento."

Y habiendo dado cuenta con él al Emperador, se ha servido S. M. I. mandar se cumpla, publique y circule; y á V. S. lo comunico de su órden para los fines que le tocan.

Dios guarde á V. S. muchos años. Méjico 3 de setiembre de 1822.—Medina.

Y para que llegue á noticia de todos, mando se publique por bando para su inteligencia y cumplimiento. Dado en Veracruz á 15 de octubre de 1822.

Antonio Lopez de Santa-Anna.

Por mandado de S. E.
José Gimenez.

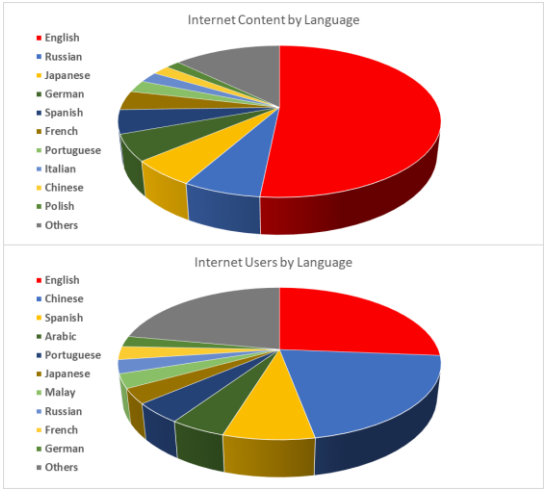
Library (book) search

1950's

What is IR?

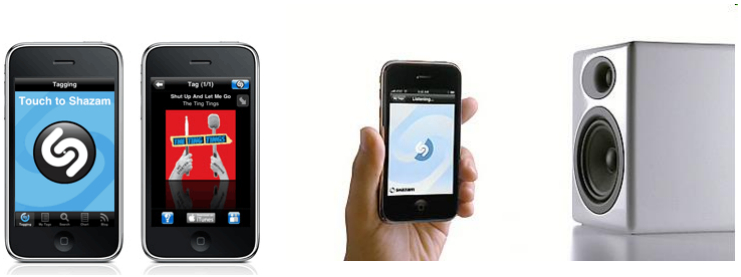
Legal search

What is IR?



Cross-Language search

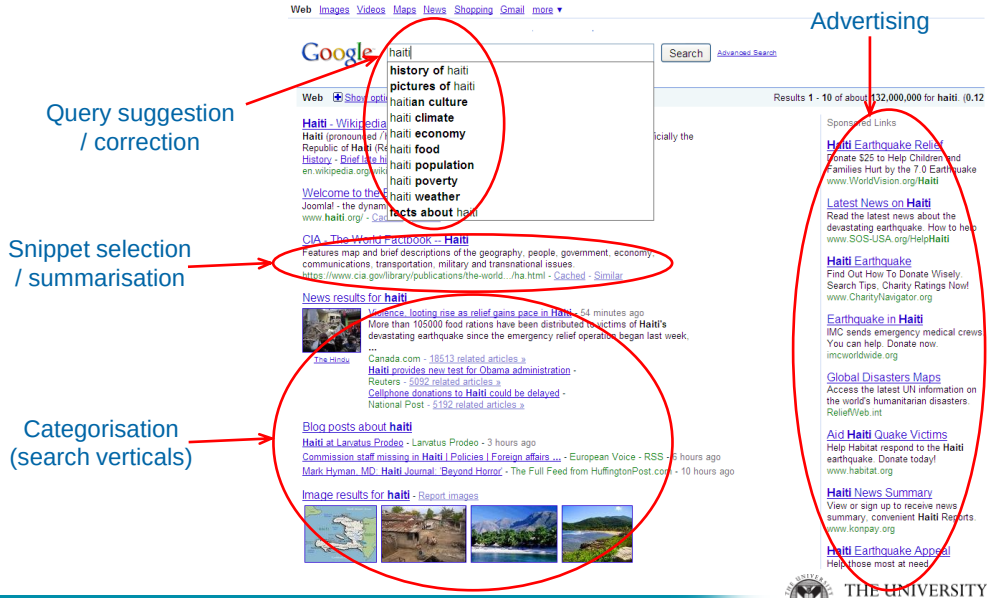
What is IR?



Content-based music search

*Source: Matt Lease (IR Course at U Texas)

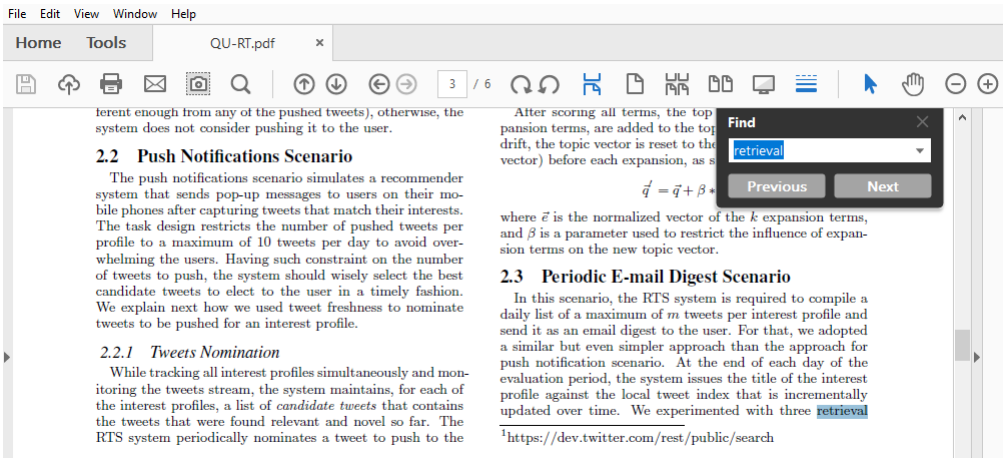
What is IR?



THE UNIVERSITY of EDINBURGH

11

What is IR? Find?



IR ≠ Find

- Sequential
- Exact match

THE UNIVERSITY of EDINBURGH

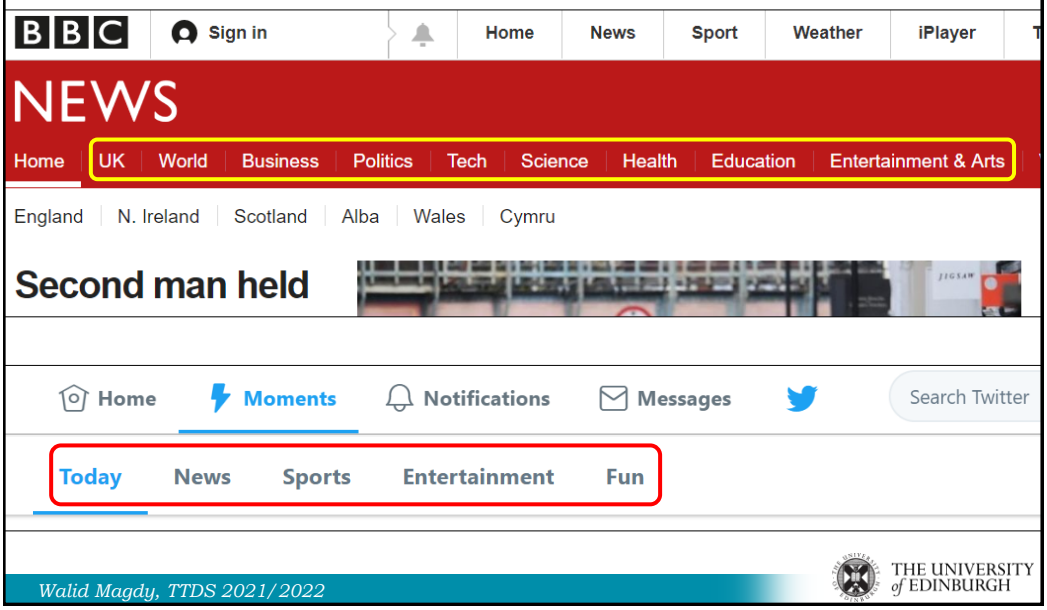
12

What is IR?

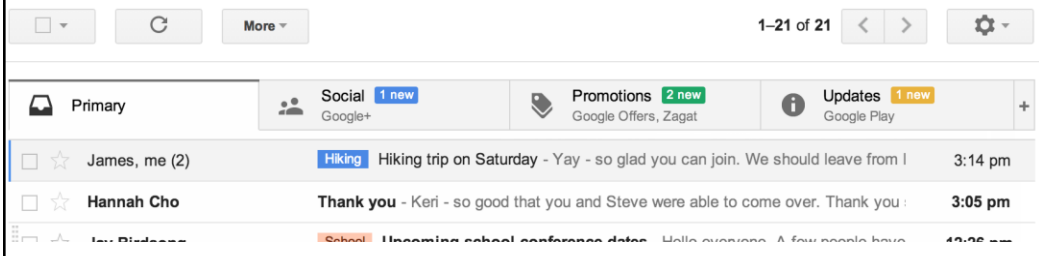
- **IR** is finding material of an unstructured nature that satisfies an information need from within large collections

- Find → Task
- Unstructured → Nature
- Information need → Target
- Satisfies → Evaluation

Text classification



Text classification



Text classification

US008881191B2

(12) **United States Patent**
Magdy et al.

(10) **Patent No.: US 8,881,191 B2**
(45) **Date of Patent: Nov. 4, 2014**

(54) **PERSONALIZED EVENT NOTIFICATION USING REAL-TIME VIDEO ANALYSIS**

(75) Inventors: **Walid Magdy**, Giza (EG); **Motaz El-Saban**, Giza (EG)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this

(51) **Int. Cl.**
H04H 60/65 (2008.01)
H04H 60/48 (2008.01)
G06F 17/30 (2006.01)

(52) **U.S. Cl.**
CPC ***H04H 60/48*** (2013.01); ***H04H 60/65*** (2013.01); ***G06F 17/30787*** (2013.01); ***G06F 17/30831*** (2013.01)
USPC **725/32**; 725/43; 725/52; 382/181; 348/460

What is text classification?

- **Text classification** is the process of classifying documents into predefined categories based on their content.

- Input: Text (document, article, sentence)
- Task: Classify into one/multiple categories
- Categories:
  - Binary: relevant/irrelevant, spam .. etc.
  - Few: sports/politics/comedy/technology
  - Hierarchical: patents

In this course, we will learn to

- How to build a search engine
  - which search results to rank at the top
  - how to do it fast and on a massive scale
- How to evaluate a search algorithm
  - is system A really better than system B
- How to work with text
  - two tweets talk about the same topic?
  - handle misspellings, morphology, synonyms
- How to classify text
  - into categories (sports, news, comedy, …)
  - features to use
  - evaluate classification quality
- Apply text analytics
  - Find what makes a set of document different from others

How this course is different from others?

- ANLP, FNLP
  - Some text processing
  - Text laws
  - No NLP (word/phrase level vs document level)
- ML practical
  - Text classification
  - No ML (using off-the-shelf ML tool)
- It does not overlap with others on:
  - Search engines
  - IR methods/models
  - IR evaluation
  - Text analysis
  - Processing large amount of textual data

Some terms you will learn about

- Inverted index
- Vector space model
- Retrieval models: TFIDF, BM25, LM
- Page rank
- Learning to rank (L2R)
- MAP, MRR, nDCG
- Mutual information, information gain, Chi-square
- binary/multiclass classification, ranking, regression

This Course is Highly Practical

- 70% of the mark is on practical work
- You will implement 50+% of what you learn
- By W5, you should have developed a basic working Search Engine from scratch
- Practical Lab every week
- Two coursework, mostly coding
- A course group project to develop a full system

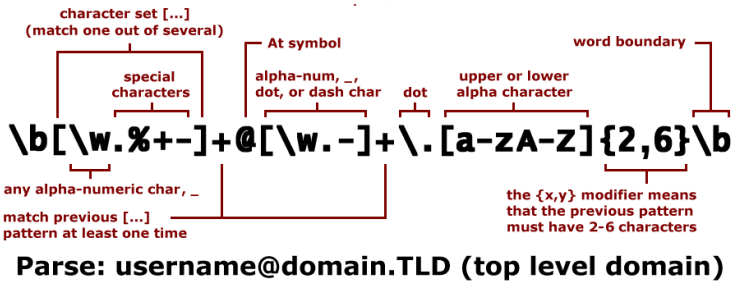
Pre-requests (1/3)

- Maths requirements:
  - Linear algebra: vectors/matrices (addition, multiplication, inverse, projections ... etc).
  - Probability theory: Discrete and continuous univariate random variables. Bayes rule. Expectation, variance. Univariate Gaussian distribution.
  - Calculus: Functions of several variables. Partial differentiation. Multivariate maxima and minima.
  - Special functions: Log, Exp, Ln.

$$\text{BM25}(D,Q) = \sum_{i=1}^{n} \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} \cdot \left[\frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)} + \delta \right]$$

Pre-requests (2/3)

- Programming requirements:
  - Python
  - Knowledge in regular expressions
  - Shell commands (cat, sort, grep, uniq, sed, ...)
  - Data structures and software engineering for course project.



Parse: username@domain.TLD (top level domain)

Pre-requests (3/3)

- Team-work requirement:
  - Final course project would be in groups of 5-6 students.
  - Working in a team for the project is a requirement.
  - No exceptions will be allowed!

Skills to be gained !!!

- Working with large text collections
- Few shell commands
- Some Python programming
- Software engineering skills
- Build text classifier in few mins
- TEAM WORK
  - Project management
  - Time management
  - Task assignment + system integration

Course Structure

- 20 Lectures:
  - 2 lectures → Introduction (today)
  - 14 lectures → IR (50% practical lectures)
  - 4 lectures → Text Analytics/Classification
- 8-10 Labs:
  - Practice what you learn
- No Tutorials
- Some self-reading
- Lots of system implementation
- Few online videos

Course Instructors

Walid Magdy
Reader
(14 lectures)

Bjorn Ross
Lecturer
(5 lectures)

+ 1 guest lecture

Lecture Format

- 2 Lectures at a time
- Questions are allowed any time. Feel free to interrupt
- 5-10 mins break after L1
  - Feel free to go out and come back
  - Discuss 1st lecture with friends
  - Questions on L1 are allowed before starting L2
  - Mind teaser math problem (for fun)
- Some lectures are interactive. Please participate
- Some lectures will include demos (running code)

Labs

- Online!
- How it will work?
  - Relevant lab will be announced with each lecture on Wednesday
  - You should implement lab directly after lecture
  - Any issues → ask on Piazza (tag question by lab number)
  - Produced output → Share on Piazza (publicly)
  - Demonstrators → answer questions + validate your output
  - DO NOT ask a question before checking if it was asked before
  - Tuesdays → Optional Teams meetings for those still require support
- Live lab times: Tuesday, 9am, 11am, 6pm
- Demonstrators:
  Zheng Zhao, Ibrahim Abu Farha, and Youssef Al Hariri

Lab Zero (Lab 0)

- Please check Lab 0 before next week lectures
- Lab 0 is designed for one purpose:
  Help you decide to take TTDS or not
- Lab content:
  Read a text file word by word, lower-case letters, print
- If you Lab 0 challenging then:
  → Probably, TTDS would be very challenging to you
  → You will need much extra effort to implement labs and CW
  → Think wisely before you decide to take the course

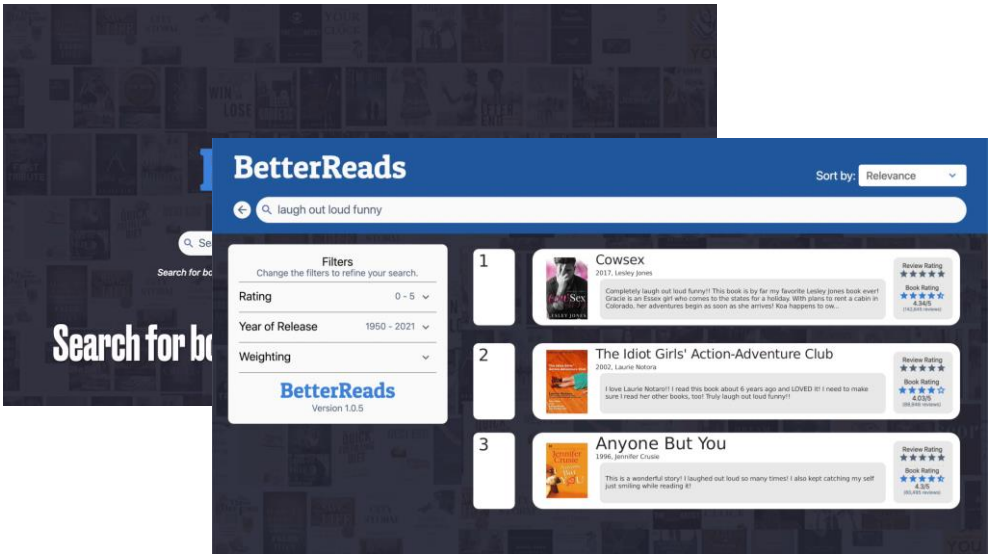
Assessments

- Coursework 1: **10%**
  The same as labs 1-3 → Build your first search engine
- Coursework 2: **20%**
  IR Evaluation, Text classification/analytics
- Group project: **40%**
  A full running search engine supported by text technologies
- Final Exam: **30%**

Group Project

- The largest weight: 40% of the total mark
- Teamwork → Group 5-6 (you select your group)
- Design a full end-to-end search engine that searches a large collection of documents with many functionalities.
- Mark = $Mark_{project}$ x $weight_{individual}$
  - $Mark_{project}$ → the same for all team members
    - How complete/effective/fast/nice is your search engine?
  - $weight_{individual}$ → weight for individual contribution.
    - ranges from 0 to 1. It should be 1.0 by default but can be different for each member according to their contribution.
- Project prize → a prize will be awarded to best project

Example: BetterReads



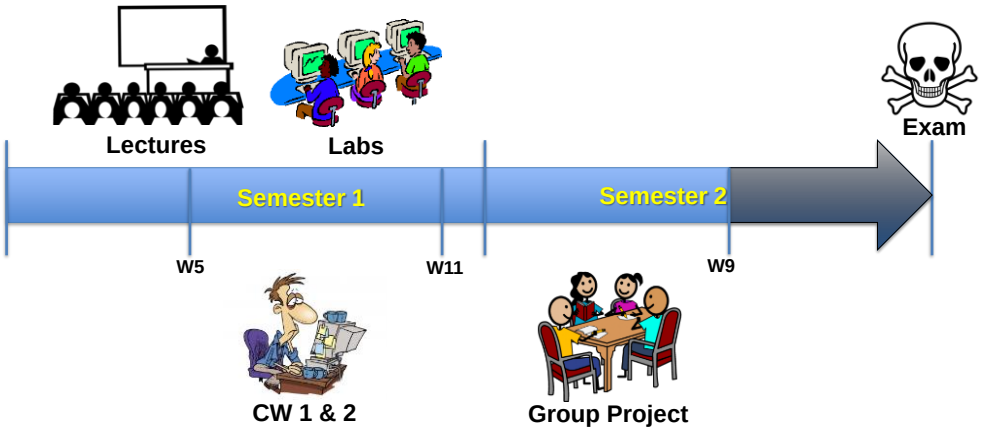
Example: BetterReads

- 11.5M Book reviews from Good reads
- Average query time: 1.3 secs
- New reviews are crawled and indexed automatically every month
- Ranking: Relevance + Sentiment
- Engine hosted on Google cloud compute

- *Note: we will provide credit to Google cloud to host your engine*

Timeline

- 2 Semesters (or one?)



Logistics (1/2)

- Lectures:
  - Live on 2 Wednesdays, 12.00-14.30 (some exception might occur)
  - Recording will be available
  - Handouts to be posted on the day of the lecture
- Course webpage:
  - Link: http://www.inf.ed.ac.uk/teaching/courses/tts/
  - Handouts, Labs, CW details, link to recordings
- Learn:
  - Lecture recordings
  - Deadlines
- Note: all course materials are made public including recordings. Feel free to share with anyone interested

Logistics (2/2)

- Pizza:
  - All communication will be there
  - Questions about lectures/labs/CW are there
  - Feel free to answer each other questions
  - Lab support will be mainly there
  - Please share your lab answers there
  - Join NOW: link
- Microsoft Teams:
  - Live lab support will be there
  - Join NOW: link

FAQ

- How the project would be managed? What if one member does not work?
- I am not that solid in programming, should I take this course?
- Can I audit the course?
- Anything else?

Next Lecture

- Definitions of IR main concepts (more introduction)

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Definitions

Instructor:
Walid Magdy

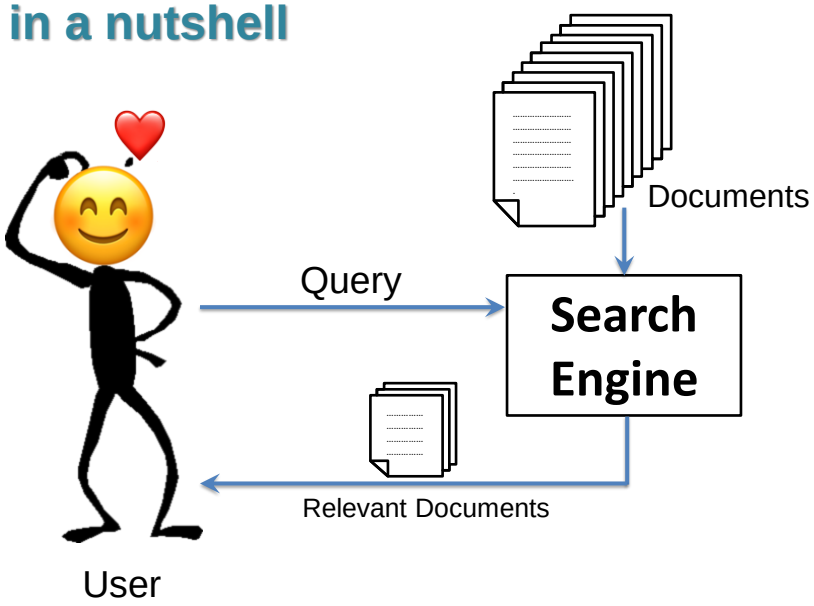
22-Sep-2021

1

Lecture Objectives

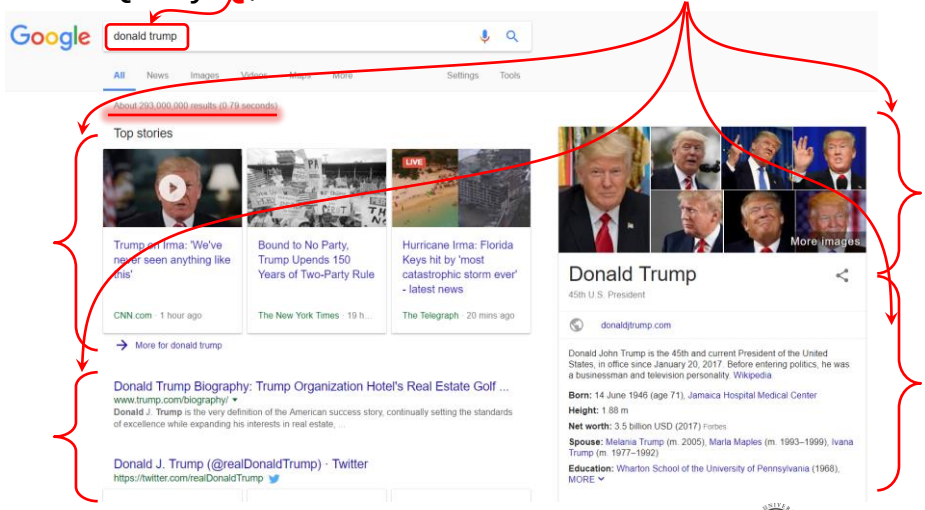
- Learn about main concepts in IR
  - Document
  - Information need
  - Query
  - Index
  - BOW

IR in a nutshell



IR, basic form

- Given Query **Q**, find relevant documents **D**



Two main Issues in IR

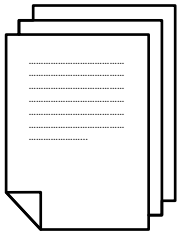
About 293,000,000 results (0.79 seconds)

- Effectiveness
  - need to find **relevant** documents
  - needle in a haystack
  - very different from relational DBs (SQL)
- Efficiency
  - need to find them quickly
  - vast quantities of data (100's billions pages)
  - thousands queries per second (Google, 63,000)
  - data constantly changes, need to keep up
  - compared with other NLP areas, IR is **very** fast

IR main components

- Documents
- Queries
- Relevant documents

Documents



- The element to be retrieved
  - Unstructured nature
  - Unique ID
  - *N* documents → Collection
- web-pages, emails, book, page, sentence, tweets
- photos, videos, musical pieces, code
- answers to questions
- product descriptions, advertisements
- may be in a different language
- may not have words at all (e.g. DNA)

Queries

- Free text to express user's information need
- Same information need can be described by multiple queries
  - Latest news on the hurricane in the US
  - North Carolina storm
  - Florence
- Same query can represent multiple information needs
  - Apple
  - Jaguar




Queries – different forms

- Web search → keywords, narrative …
- Image search → keywords, sample image
- QA → question
- Music search → humming a tune
- Filtering/recommendation → user's interest/history
- Scholar search → structured (author, title ..)

- Advanced search

  #wsyn(0.9 **#field** (title, **#phrase** (homer,simpson)) 0.7 **#and** (**#>** (pagerank,3), **#ow3** (homer,simpson)) 0.4 **#passage** (homer, simpson, dan, castellaneta))

Relevance

- At an abstract level, IR is about:
  - does item *D* ***match*** item *Q*? …or…
  - is item *D* ***relevant*** to item *Q*?
- Relevance a tricky notion
  - will the user like it / click on it?
  - will it help the user achieve a task? (satisfy information need)
  - is it novel (not redundant)?
- *Relevance = what is the topic about?*
  - i.e. *D*,*Q* share similar "meaning"
  - about the same topic / subject / issue

What is the challenge in relevance?

- No clear semantics, contrast:
  - *"William Shakespeare"*
  - Author history's? list of plays? a play by him?
- Inherent ambiguity of language:
  - synonymy: "North Carolina storm" = "Florence hurricane"
  - polysemy: "Apple", "Jaguar"
- Relevance highly subjective
  - Relevance: yes/no
  - Relevance: perfect/excellent/good/fair/bad
- On the web: counter SEOs / spam

Relevant Items are Similar

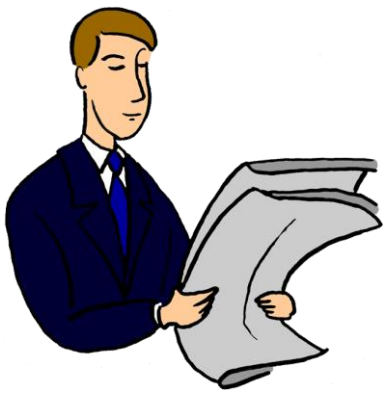
- Key idea:
  - Use similar vocabulary → similar meaning
  - Similar documents relevant to same queries
- Similarity
  - String match
  - Word overlap
  - P(D|Q) → retrieval model

IR vs. DB

	Databases	IR
What we're retrieving	Structured data. Clear semantics based on a formal model.	Mostly unstructured. Free text with some metadata.
Queries we're posing	Formally-defined (relational algebra, SQL). Unambiguous.	Free text ("natural language"), Boolean
Results we get	Exact (always "correct")	Imprecise (need to measure relevance)
Interaction with system	One-shot queries.	Interaction is important.

Tamer Elsayed, QU

How IR sees documents?



Bag-of-words trick

- Can you guess what this is about:
  - per is salary hour €4,200 Neymar's
  - obesity French is of full cause and fat fries
- Re-ordering doesn't destroy the topic
  - individual words – "building blocks"
  - "bag" of words: a "composition" of "meanings"

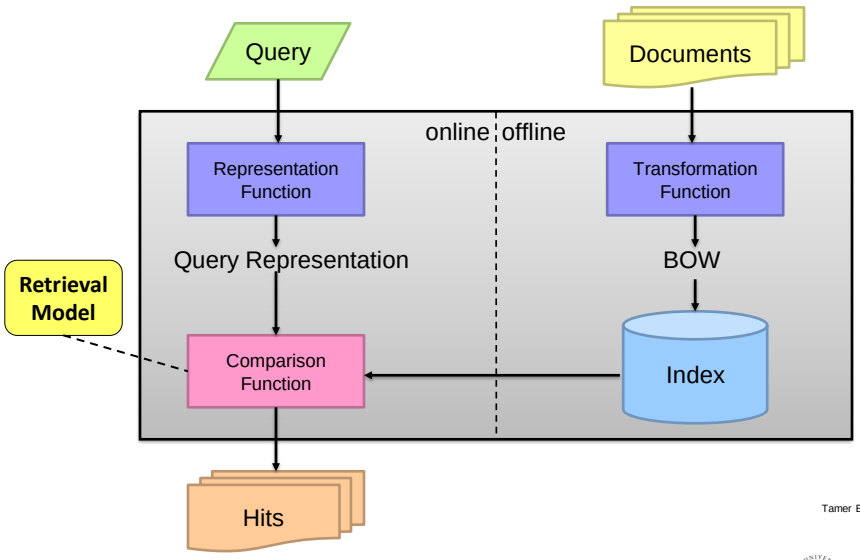
Bag-of-words trick

- Most search engines use BOW
  - treat documents, queries as bags of words
- A "bag" is a set with repetitions
  - match = "degree of overlap" between D,Q
- Retrieval models
  - statistical models (function) that use words as features
  - decide which documents most likely to be relevant
- What should be the top results for Q?
  - BOW makes these models tractable

Bag-of-words: Criticism

- word meaning lost without context
  - True, but BOW doesn't really discard context
- what about negations, etc.?
  - {no, climate change is real} vs. {climate change is no real}
- does not work for all languages
  - No natural "word" unit for Chinese, images, music
  - Solve by "segmentation" or "feature induction"

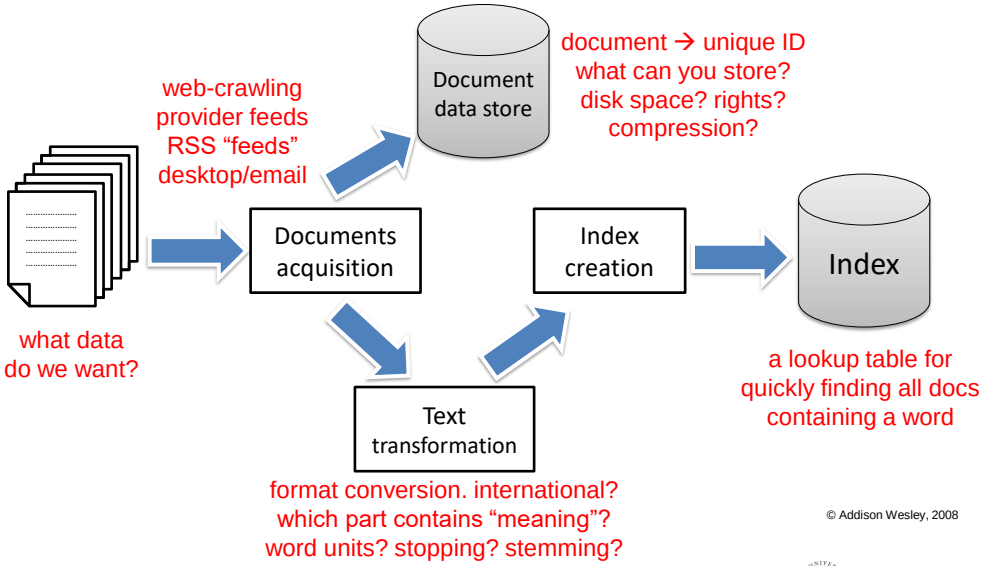
IR Black Box



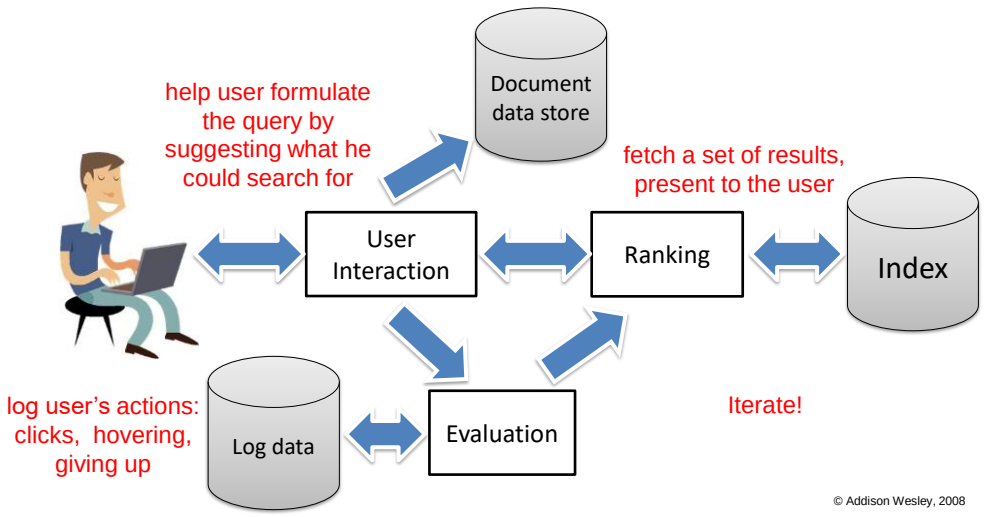
Systems perspective on IR

- Indexing Process: *(offline)*
  → get the data into the system
  - acquire the data from crawling, feeds, etc.
  - store the originals (if needed)
  - transform to BOW and "index"
- Search (retrieval) Process: *(online)*
  → satisfy users' requests
  - assist user in formulating query
  - retrieve a set of results
  - help user browse / re-formulate
  - log user's actions, adjust retrieval model

Indexing Process



Search Process



Summary

- Information Retrieval (IR): core technology
  - selling point: IR is very fast, provides context
- Main issues: effectiveness and efficiency
- Documents, queries, relevance
- Bag-of-words trick
- Search system architecture:
  - indexing: get data into the system
  - searching: help users find relevant data

Resources

- Search Engines: Information Retrieval in Practice, chapter 1 & 2
- Lab 0:
  - You have to be confident doing it!
  - If you have trouble finishing it, think twice before committing to the course

Questions

- Next time:
  - Laws of text (Zipf ….)
  - Vector space models
- Skill to learn by next time:
  - Read text file from disk
  - Read word by word
- Videos:
  - The Zipf Mystery, Vsauce
- Tools:
  - (Perl) regular expressions: https://perldoc.perl.org/perlre.html

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Laws of Text

Instructor:
Walid Magdy

29-Sep-2021

1

Lecture Objectives

- Learn about some text laws
  - Zipf's law
  - Benford's law
  - Heap's law
  - Clumping/contagion
- This lecture is practical

You can try with me …

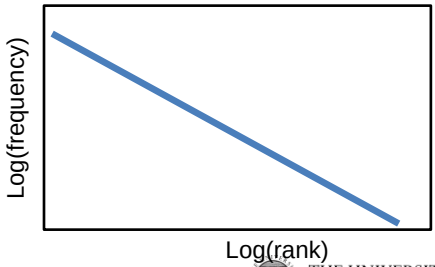
- Shell commands: cat, sort, uniq, grep
- Perl (or alternative)
- Excel (or alternative)
- Download the following:
  - Bible: http://www.gutenberg.org/cache/epub/10/pg10.txt

Words' nature

- Word → basic unit to represent text
- Certain characteristics are observed for the words we use!
- These characteristics are very consistent, that we can apply laws for them
- These laws apply for:
  - Different languages
  - Different domains of text

Frequency of words

- Some words are very frequent
  e.g. “the”, “of”, “to”
- Many words are less frequent
  e.g. “schizophrenia”, “bazinga”
- ~50% terms appears once
- Frequency of words has hard exponential decay

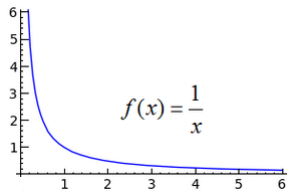


Zipf’s Law:

- For a given collection of text, ranking unique terms according to their frequency, then:

$$r \times P_r \cong const$$

- r, rank of term according to frequency
- P_r, probability of appearance of term
- $P_r \cong \frac{const}{r} \rightarrow f(x) \cong \frac{1}{x}$



Zipf's Law:

Wikipedia abstracts

→ 3.5M En abstracts

$r \times P_r \cong const \rightarrow$

$r \times freq_r \cong const$

Term	Rank	Frequency	r x freq
the	1	5,134,790	5,134,790
of	2	3,102,474	6,204,948
in	3	2,607,875	7,823,625
a	4	2,492,328	9,969,312
is	5	2,181,502	10,907,510
and	6	1,962,326	11,773,956
was	7	1,159,088	8,113,616
to	8	1,088,396	8,707,168
by	9	766,656	6,899,904
an	10	566,970	5,669,700
it	11	557,492	6,132,412
for	13	493,374	5,970,456
as	14	480,277	6,413,862
on	15	471,544	6,723,878
from	16	412,785	7,073,160

Practical

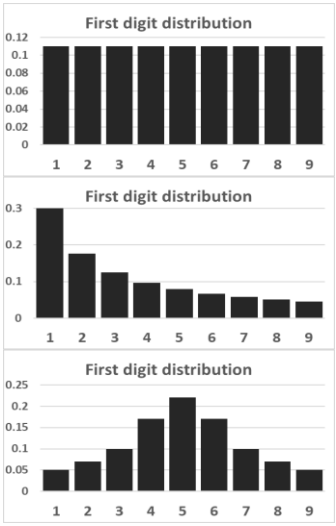
Collection	# words	File size
Bible	824,054	4.24 MB
Wiki abstracts	80,460,749	472 MB

Distribution of first digit in frequencies?

1) Uniform →

2) Exp decay →

3) Normal →

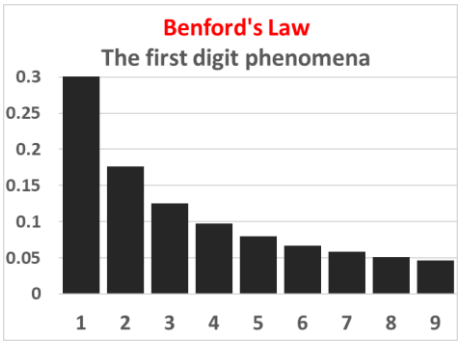


Term	Rank	Frequency
the	1	5 134,790
of	2	3 102,474
in	3	2 607,875
a	4	2 492,328
is	5	2 181,502
and	6	1 962,326
was	7	1 159,088
to	8	1 088,396
by	9	766,656
an	10	566,970
it	11	557,492
for	13	493,374
as	14	480,277
on	15	471,544
from	16	412,785

Benford's Law:

- First digit of a number follows a Zipf's like law!
  - Terms frequencies
  - Physical constants
  - Energy bills
  - Population numbers
- Benford's law:

$$P(d) = \log(1 + \frac{1}{d})$$



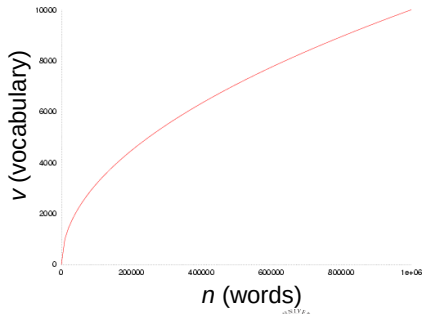
Practical

Heap's Law:

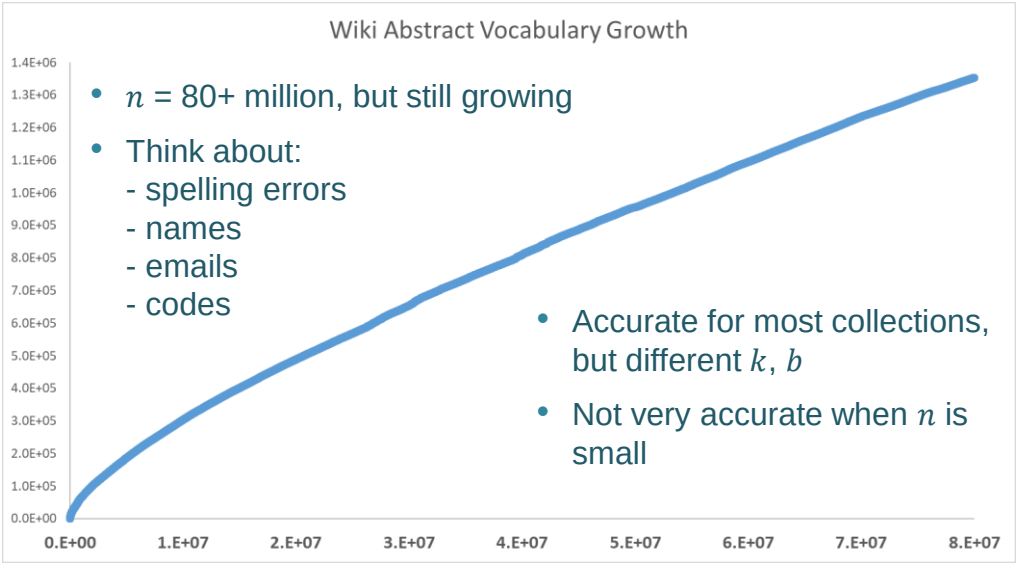
- While going through documents, the number of new terms noticed will reduce over time
- For a book/collection, while reading through, record:
  - n: number of words read
  - v: number of news words (unique words)
- Vocabulary growth:

$$v(n) = k \times n^b$$

where, $b < 1$
typically, $0.4 < b < 0.7$



Heap's Law: shouldn't it saturate?



- n = 80+ million, but still growing
- Think about:
  - spelling errors
  - names
  - emails
  - codes
- Accurate for most collections, but different k, b
- Not very accurate when n is small

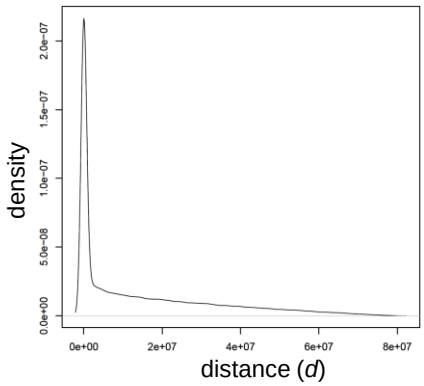
Practical

Clumping/Contagion in text

- From Zipf's law, we notice:
  - Most words do not appear that much!
  - Once you see a word once → expect to see again!
  - Words are like:
    - Rare contagious disease
    - Not, rare independent lightening
- Words are rare events, but they are contagious

Clumping/Contagion in text

- Wiki abstract collection
  - Identify terms appeared only twice
  - Measure distance between the two occurrences of the terms: $d = n_{occurence2} - n_{occurence1}$
  - Plot density function of d
- Majority of terms appearing only twice appear close to each other.



Applying the laws

- Given a collection of 20 billion terms,
- What is the number of unique terms?

- What is the number of terms appearing once?

Summary

- Text follows well-known phenomena
- Text Laws:
  - Zipf
  - Heap
  - Contagion in text

- Try it on another language …

Recourses

- Text book:
  - Search engines: IR in practice → chapter 4
- Videos:
  - Zipf's law, Vsouce: https://www.youtube.com/watch?v=fCn8zs912OE
  - Benford's law, Numberphile: https://www.youtube.com/watch?v=XXjlR2OK1kM
- Tools:
  - Unix commands for windows https://sourceforge.net/projects/unxutils

Next Lecture

- Getting ready for indexing?
- Pre-processing steps before the indexing process

- Reminder: 5-10 mins break after L1
  - Have a break, stretch, get food … etc.
  - Ask questions on chat
  - Questions on L1 are allowed before starting L2
  - Mind teaser math problem (for fun)

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Preprocessing

Instructor:
Walid Magdy

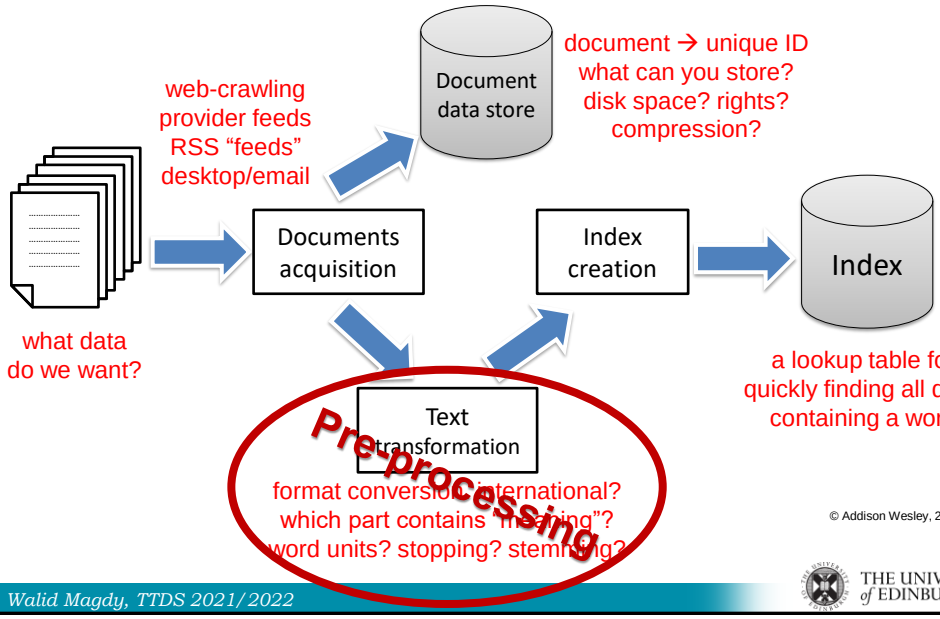
29-Sep-2021

1

Lecture Objectives

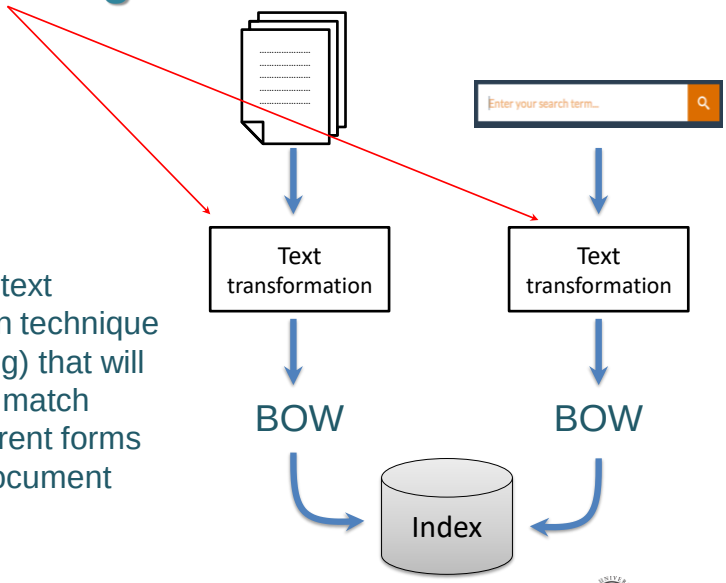
- Learn about and implement
- Standard text pre-processing steps:
  - Tokenisation
  - Stopping
  - Normalisation
    - Stemming

Indexing Process



Preprocessing

Find the best text transformation technique (preprocessing) that will lead to better match between different forms of words in document and query



Getting ready for indexing?

- BOW, what is a word?
- In IR, we refer to word-elements as "terms"
  - word "*preprocessing*"
  - part of a word "*pre*"
  - number / code "*INFR11145*"
- Pre-processing steps before indexing:
  - Tokenisation
  - Stopping
  - Stemming
- **Objective** → identify the optimal form of the term to be indexed to achieve the best retrieval performance

Tokenisation

- Input: "***Friends, Romans; and Countrymen!***"
- Output: Tokens
  - *Friends*
  - *Romans*
  - *and*
  - *Countrymen*
- Sentence → tokenization (splitting) → tokens
- A **token** is an **instance** of a sequence of characters
- **Typical technique:** split at non-letter characters
- Each such token is now a candidate for an index entry (**term**), after further processing

Issues in Tokenisation

- "*Finland's*" capital → *Finland*? *Finlands*? *Finland's*?
- Hewlett-Packard → one token or two?
  - **state-of-the-art**: break up hyphenated sequence.
  - *co-education*
  - *lowercase, lower-case, lower case ?*
  - It can be effective to get the user to put in possible hyphens
- **Numbers?**
  - 3/20/91 vs. Mar. 20, 1991 vs. 20/3/91
  - This course code is INFR11145
  - (800) 234-2333

Issues in Tokenisation

- **URLs:**
  - http://www.bbc.co.uk
  - http://www.bbc.co.uk/news/world-europe-41376577
- **Social Media**
  - Black lives matter
  - #Black_lives_matter
  - #BlackLivesMatter
  - #blacklivesmatter
  - @blacklivesmatter
- **San Francisco**: one token or two?
  - How do you decide it is one token?

Tokenisation for different languages

- French → ***L'ensemble*** → one token or two?
  - ***L*** ? ***L'*** ? ***Le*** ?
  - Want ***l'ensemble*** to match with ***un ensemble***
  - Until at least 2003, it didn't on Google
- German → compounds
  - ***Lebensversicherungsgesellschaftsangestellter***
    'life insurance company employee'
  - German retrieval systems benefit greatly from a **compound splitter** module → Can give a 15% performance boost for German
- Chinese and Japanese → no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达
  - Tokenisation → Segmentation

Tokenisation: common practice

- Just split at non-letter characters
- Add special cases if required
- Some applications have special setup
  - Social media: hashtags/mentions handled differently
  - URLs: no split, split at domain only, remove entirely!
  - Medical: protein & diseases names

Stopping (stop words removal)

- ~~This is a very~~ exciting lecture ~~on the~~ technologies ~~of~~ text
- **Stop words**: the most common words in collection → the, a, is, he, she, I, him, for, on, to, very, …
- There are a lot of them ≈ 30-40% of text
- New stop words appear in specific domains
  - Tweets: RT → *"RT @realDonalTrump Mexico will …"*
  - Patents: said, claim → *"a said method that extracts …."*
- Stop words
  - influence on sentence structure
  - less influence on topic (aboutness)

Stopping: always apply?

- Sometimes very important:
  - Phrase queries: "Let it be", "To be or not to be"
  - Relational queries:
    - flights to London from Edinburgh
    - flights from London to Edinburgh
- In Web search, trend is to keep them:
  - Good compression techniques means the space for including stop words in a system is very small
  - Good query optimization techniques mean you pay little at query time for including stop words.
  - Probabilistic retrieval models give them low weight.

Stopping: stop words

- Common practice in many applications
  → remove stop words
- There are common stop words list for each language
  - NLTK (python)
  - http://members.unine.ch/jacques.savoy/clef/index.html
- There are special stop words list for some applications
- How to create your list:
  - Sort all terms in a collection by frequency
  - Manually select the possible stop words from top *N* terms

Normalisation

- **Objective** → make words with different surface forms look the same
- Document: “this is my CAR!!”
  Query: “car”
  should “car” match “CAR”?
- Sentence → tokenisation → tokens → normalisation → terms to be indexed
- Same tokenisation/normalisation steps should be applied to documents & queries

Case folding and equivalents

- “A” & “a” are different strings for computers
- Case folding: convert all letters to lower case
  - CAR, Car, caR → car
  - Windows → windows, should we do that?
- Diacritics/Accents removal
  - French: Château → chateau
  - German: Tüebingen → tuebingen
  - Arabic: كُتُبٍ → كتب

Equivalence Classes

- U.S.A. → USA
- Ph.D. → PhD
- 92.3 → 923? 92 3?
- multi-disciplinary → multidisplinary ← multi disciplinary
- The most important criteria:
  - Be consistent between documents & queries
  - Try to follow users’ most common behaviour

Stemming

- Search for: “play”
  should it match: “played”, “playing”, “player”?
- Many morphological variations of words
  - *inflectional* (plurals, tenses)
  - *derivational* (making verbs nouns etc.)
- In most cases, aboutness does not change
- Stemmers attempt to reduce morphological variations of words to a common stem
  - usually involves removing suffixes (in English)
- Can be done at indexing time or as part of query processing (like stopwords)

Stemming

- Usually, it achieves 5-10% improvement in retrieval effectiveness, e.g. English
- For highly inflected languages, it is more critical:
  - 30% improvement in Finnish IR
  - 50% improvement in Arabic IR

They are Peter’s **children**	هؤلاء **أبناء** بيتر
The **children** behaved well	**الأبناء** تصرفوا جيدا
Her **children** are cute	**أبناءها** لطاف
My **children** are funny	**أبنائي** ظرفاء
We have to save our **children**	علينا أن نحمي **أبناءنا**
Patents and **children** are happy	الآباء **والأبناء** سعداء
He loves his **children**	هو يحب **أبناءه**
His **children** loves him	**أبناؤه** يحبونه

Stemming

- Two basic types
  - Dictionary-based: uses lists of related words
  - Algorithmic: uses program to determine related words
- Algorithmic stemmers
  - suffix-s: remove ‘s’ endings assuming plural
  - e.g., cats → cat, lakes → lake, windows → window
  - Many false negatives: supplies → supplie
  - Some false positives: James → Jame

Porter Stemmer

- Most common algorithm for stemming English
- Conventions + 5 phases of reductions
  - phases applied sequentially
  - each phase consists of a set of commands
  - sample convention:
    of the rules in a compound command, select the one that applies to the longest suffix.
- Example rules in Porter stemmer
  - *sses* → *ss* (processes → process)
  - *y* → *i* (reply → repli)
  - *ies* → *i* (replies → repli)
  - *ement* → null (replacement → replac)

Stemmed words are misspelled!!

- repli, replac, suppli, inform retriev, anim
- These are not words anymore, these are terms
- These terms are not seen by the user, but just used by the IR system (search engine)
- These represent the optimal form for a better match between different surface forms of a term
  - e.g. replac → replace, replaces, replaced, replacing, replacer, replacers, replacement, replacements.

Pre-processing: Common practice

- Tokenisation: split at non-letter characters
  - One line of code in Perl
    → process \w and neglect anything else
  - For tweets, you might want to keep "#" and "@"
- Remove stop words
  - find a common list, and filter these words out
- Apply case folding
  - One command in Perl or Python: lc($string)
- Apply Porter stemmer
  - Other stemmers are available, but Porter is the most famous with many implementations available in different programming languages

Limitations

- Irregular verbs:
  - saw → see
  - went → go
- Different spellings
  - colour vs. color
  - tokeni**s**ation vs. tokeni**z**ation
  - Television vs. TV
- Synonyms
  - car vs. vehicle
  - UK vs. Britain
- Solution → Query expansion …

Asymmetric Expansion

- Maintains relations between unnormalized tokens
- An alternative to equivalence classing
- An example of where this may be useful
  - query: *window* search: *window, windows*
  - query: *windows* search: *windows, Windows*
  - query: *Windows* search: *Windows*
- Potentially more powerful, but less efficient
  - More vocabulary, longer query
- Can be less effective:
  - Inaccurate stats on terms (“car” ≠ “Car”)

Summary

- Text pre-processing before IR:
  - Tokenisation → Stopping → Stemming

This is an example sentence of how the pre-processing is applied to text in information retrieval. It includes: Tokenization, Stop Words Removal, and Stemming

exampl sentenc pre process appli text inform retriev includ token stop word remov stem

Practical

Collection	Original		After Pre-processing	
	# words	File size	# words	File size
Bible	824,054	4.24 MB	358,112	2.05 MB
Wiki abstracts	78,137,597	472 MB	47,741,065	309 MB

Resources

- Text book 1: Intro to IR, Chapter 2 → 2.2.4
- Text book 2: IR in Practice, chapter 4
- Lab 1 → Implement what learnt in these two lectures
- Optional reading:
  if you think English pre-processing is hard!
  - Arabic Information Retrieval. *Darwish & Magdy*

Next lecture

- Indexing:
  How to build an index!
- Assignment 1 announcement:
  - Build indexing components
  - Today: build your pre-processing module!
  - Next time: build the index

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Indexing

Instructor:
Walid Magdy

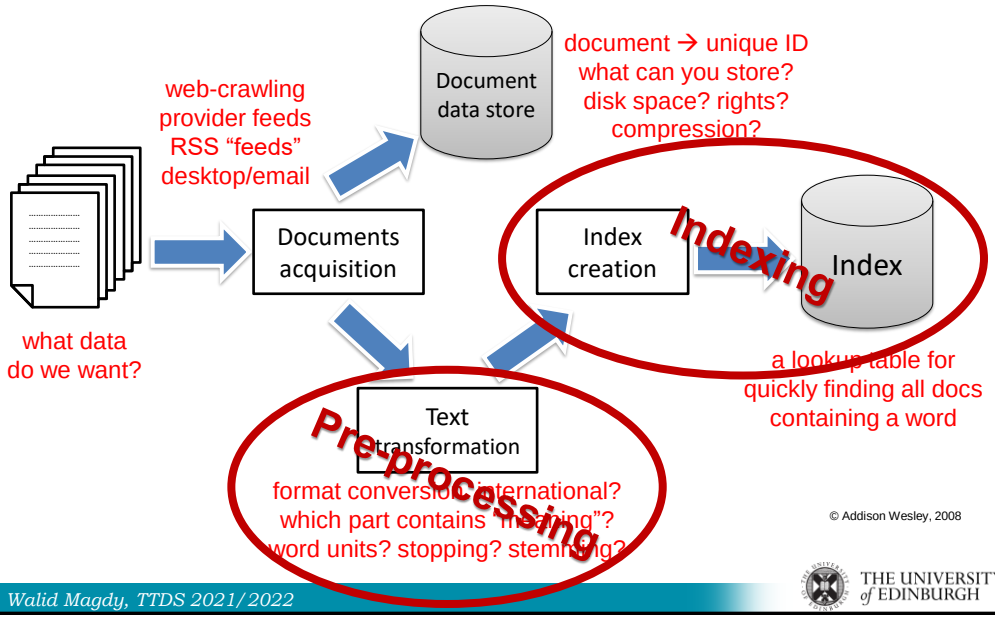
07-Oct-2020

1

Lecture Objectives

- Learn about and implement
- Boolean search
- Inverted index
- Positional index

Indexing Process



3

Pre-processing output

This is an example sentence of how the pre-processing is applied to text in information retrieval. It includes: Tokenization, Stop Words Removal, and Stemming

↓

exampl sentenc pre process appli text inform retriev includ token stop word remov stem

- Add processed terms to index
- What is "index"?

4

Index

- How to match your term in non-linear time?
- Find/Grep:
  Sequential search for term
- Index:
  Find term locations immediately

Book Index

Index

Indexing

- Search engines vs PDF find or grep?
  - Infeasible to scan large collection of text for every “search”
  - Find section that has: “UK and Scotland and Money”?!
- Book Index
  - For each word, list of “relevant” pages
  - Find topic in sub-linear time
- IR Index:
  - Data structure for fast finding terms
  - Additional optimisations could be applied

Document Vectors

- Represent documents as vectors
  - Vector → document, cell → term
  - Values: term frequency or binary (0/1)
  - All documents → collection matrix

he	drink	ink	likes	pink	think	wink	
2	1	0	2	0	0	1	← **D1:** He likes to wink, he likes to drink
1	3	0	1	0	0	0	← **D2:** He likes to drink, and drink, and drink
1	1	1	1	0	1	0	← **D3:** The thing he likes to drink is ink
1	1	1	1	1	0	0	← **D4:** The ink he likes to drink is pink
1	1	1	1	1	0	1	← **D5:** He likes to wink, and drink pink ink

number of occurrence of a term in a document

Inverted Index

- Represent terms as vectors
  - Vector → term, cell → document
  - Transpose of the collection matrix
  - Vector: inverted list

he	drink	ink	likes	pink	think	wink	
2	1	0	2	0	0	1	← **D1:** He likes to wink, he likes to drink
1	3	0	1	0	0	0	← **D2:** He likes to drink, and drink, and drink
1	1	1	1	0	1	0	← **D3:** The thing he likes to drink is ink
1	1	1	1	1	0	0	← **D4:** The ink he likes to drink is pink
1	1	1	1	1	0	1	← **D5:** He likes to wink, and drink pink ink

Boolean Search

- Boolean: exist / not-exist
- Multiword search: logical operators (AND, OR, NOT)
- Example
  - Collection: search Shakespeare's Collected Works
  - Boolean query: Brutus AND Caesar AND NOT Calpurnia
- Build a **Term-Document Incidence Matrix**
  - Which term appears in which document
  - Rows are terms
  - Columns are documents

Collection Matrix

Documents

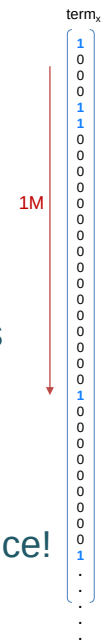
Terms	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

1 if ***document*** contains ***term***, 0 otherwise

Query: Brutus AND Caesar AND NOT Calpurnia
Apply on rows: **110100** AND **110111** AND !(**010000**) = **100100**

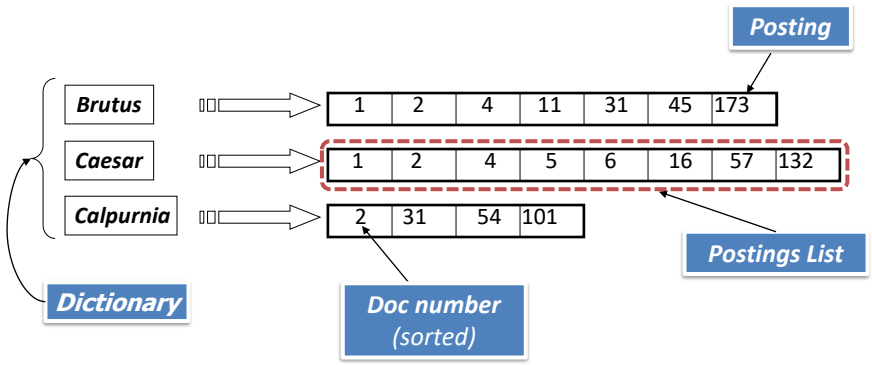
Bigger collections?

- Consider *N* = 1 million documents, each with about 1000 words.
- *n* = 1M x 1K = 1B words
  ➔ Heap's law → *v* ≈ 500K
- Matrix size = 500K unique terms x 1M documents = 0.5 trillion 0's and 1's entries!
- If all words appear in many documents
  → max{count(1's)} = *N* * doc. length = 1B
- Actually, from Zip's law → 250k terms appears once!
- Collection matrix is extremely **sparse**. (*mostly 0's*)

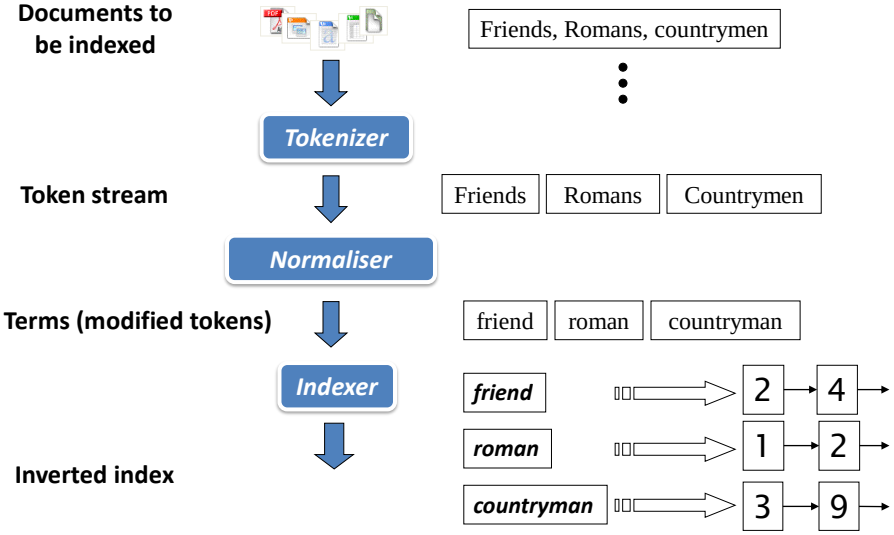


Inverted Index: Sparse representation

- For each term *t*, we must store a list of all documents that contain *t*.
  - Identify each by a **docID**, a document serial number



Inverted Index Construction



Step 1: Term Sequence

Doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Sequence of (term, Doc ID) pairs

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Step 2: Sorting

- **Sort by:**
  - **1) Term**
  - **then**
  - **2) Doc ID**

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

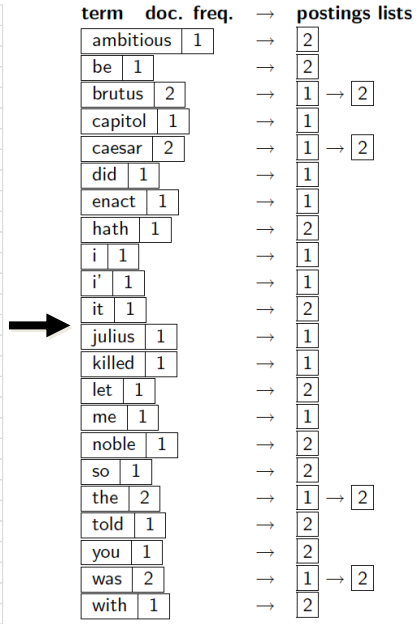
Sorting

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Step 3: Posting

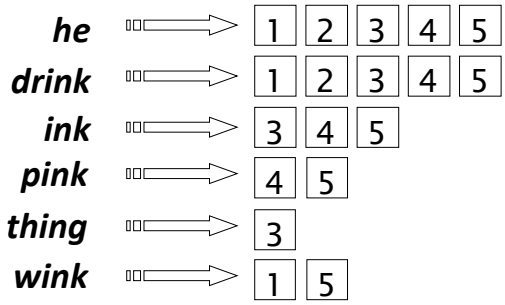
1. Multiple term entries in a single document are merged
2. Split into Dictionary and Postings
3. Doc. Frequency (*df*) information is added

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2



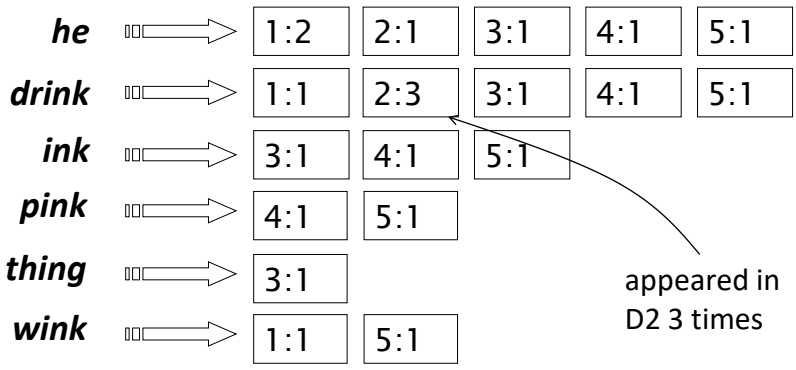
Inverted Index: matrix → postings

he	drink	ink	likes	pink	think	wink	
2	1	0	2	0	0	1	← **D1:** He likes to wink, he likes to drink
1	3	0	1	0	0	0	← **D2:** He likes to drink, and drink, and drink
1	1	1	1	0	1	0	← **D3:** The thing he likes to drink is ink
1	1	1	1	1	0	0	← **D4:** The ink he likes to drink is pink
1	1	1	1	1	0	1	← **D5:** He likes to wink, and drink pink ink



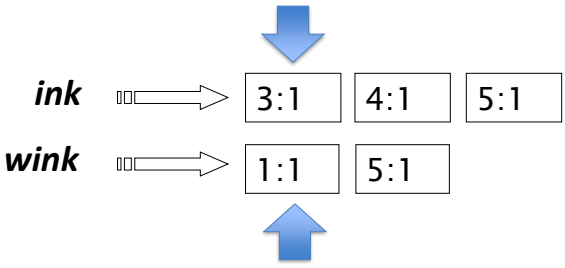
Inverted Index: with frequency

- Boolean: term → DocIDs list
- Frequency: term → touples (DocID,count(term)) lists



Query Processing

- Find documents matching query {ink **AND** wink}
  1. Load inverted lists for each query word
  2. Merge two postings lists → **Linear merge**
- Linear merge → O(*n*)
  n: total number of posts for all query words



Matches

1: f(0,1)

3: f(1,0)

4: f(1,0)

5: f(1,1)

Phrase Search

- Find documents matching query “pink ink”
  1. Find document containing both words
  2. Both words has to be a phrase

- Bi-gram Index:

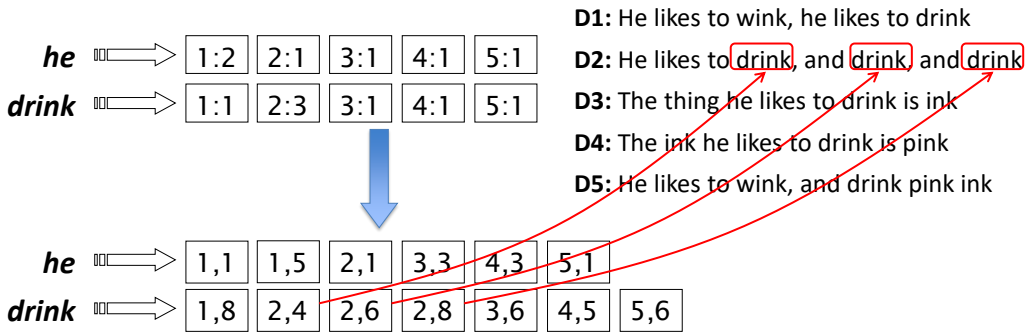
He likes to wink, and drink pink ink → Convert to bigrams

He_likes likes_to to_wink wink_and and_drink drink_pink pink_ink

- Bi-gram Index, issues:
  - Fast, but index size will explode!
  - What about trigram phrases?
  - What about proximity? “ink is pink”

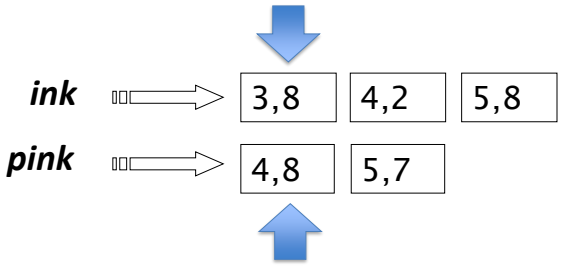
Proximity Index

- Terms positions is embedded to the inv. Index
  - Called proximity/positional index
  - Enables phrase and proximity search
  - Toubles (DocID, term position)



Query Processing: Proximity

- Find documents matching query “pink ink”
  1. Use **Linear merge**
  2. Additional step: check terms positions
- **Proximity search**:
  pos(term1) – pos(term2) < |*w*| → #5(pink,ink)



Matches

3: f(1,0) = **0**

4: f(1,1) = ? =
pos(ink) – pos(pink) == 1?

5: f(1,1) = ? =
pos(ink) – pos(pink) == 1?

Proximity search: data structure

- Possible data structure:
  <term: df;
  DocNo: pos1, pos2, pos3
  DocNo: pos1, pos2, pos3
  ……. >
- Example:
  <***be***: 993427;
  1: 7, 18, 33, 72, 86, 231;
  2: 3, 149;
  4: 17, 191, 291, 430, 434;
  5: 363, 367, …>

Summary

- Document Vector
- Term Vector
- Inverted Index
- Collection Matrix
- Posting
- Proximity Index
- Query Processing → Linear merge

Resources

- Textbook 1: Intro to IR, Chapter 1 & 2.4
- Textbook 2: IR in Practice, Chapter 5
- Lab 2

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Indexing (2)

Instructor:
Walid Magdy

06-Oct-2021

1

Lecture Objectives

- Learn more about indexing:
  - Structured documents
  - Extent index
  - Index compression
- Data structure
- Wild-char search and applications

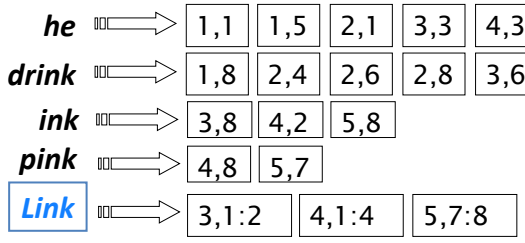
** You are not asked to implement any of the content in this lecture, but you might think of using some for your course project ☺*

Structured Documents

- Document are not always flat:
  - Meta-data: title, author, time-stamp
  - Structure: headline, section, body
  - Tags: link, hashtag, mention
- How to deal with it?
  - Neglect!
  - Create separate index for each field
  - Use “extent index”

Extent Index

- Special “term” for each element/field/tag
  - Index all terms in a structured document as plain text
  - Terms in a given field/tag get special additional entry
  - Posting: spans of window related to a given field
  - Allows multiple overlapping spans of different types



D1: He likes to wink, he likes to drink

D2: He likes to drink, and drink, and drink

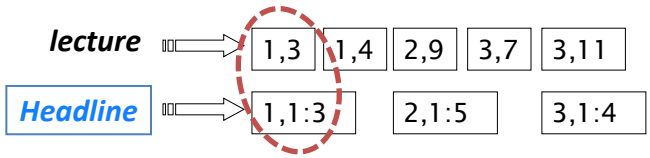
D3: The thing he likes to drink is ink

D4: The ink he likes to drink is pink

D5: He likes to wink, and drink pink ink

Using Extent

- Doc: 1 →
  Headline: "*Information retrieval lecture*"
  Text: "~~*this is*~~ *lecture 6* ~~*of the*~~ *TTDS course* ~~*on*~~ *IR*"
  1 2 3 4 5 6 7 8
- Query → Headline: lecture

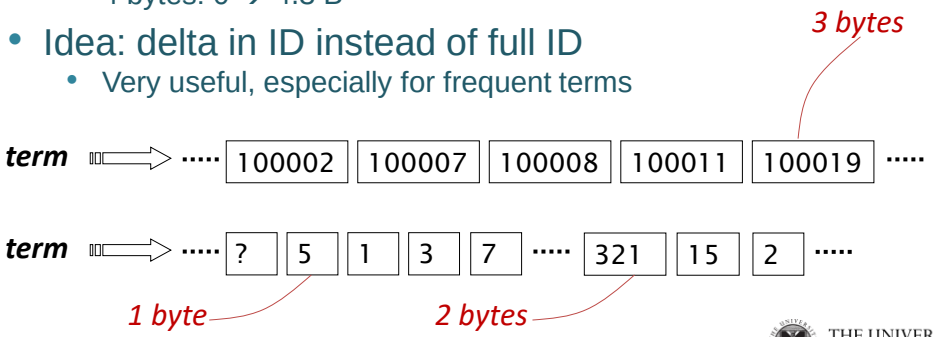


Index Compression

- Inverted indices are big
  - Large disk space → large I/O operations
- Index compression
  - Reduce space → less I/O
  - Allow more chunks of index to be cached in memory
- Large size goes to:
  - terms? document numbers?
  - Ideas:
    - Compress document numbers, how?

Delta Encoding

- Large collections → large sequence of doc IDs
  - e.g. Doc IDs: 1, 2, 3, … 66,032, ……, 5,323,424,235
- Large ID number → more bytes to store
  - 1 byte: 0→255
  - 2 bytes: 0 → 65,535
  - 4 bytes: 0 → 4.3 B
- Idea: delta in ID instead of full ID
  - Very useful, especially for frequent terms



v-byte Encoding

- Have different byte storage for each delta in index
  - Use fewer bits to encode
  - High bit in a byte → 1/0 = terminate/continue
  - Remaining 7 bits → binary number
  - Examples:
    - "6" → **1**0000110
    - "127" → **1**1111111
    - "128" → ~~**1**~~0000001~~**1**~~0000000
- Real example sequence:

10000101**0**0000001**1**0000101**1**0000111

0000101 → 0000001 0000101 → 0000111

5 → **130** → **7**

Index Compression

- There are more sophisticated compression algorithms:
  - Elias gamma code
- The more compression
  - Less storage
  - More processing
- In general
  - Less I/O + more processing > more I/O + no processing
    ">" = faster
  - With new data structures, problem is less severe

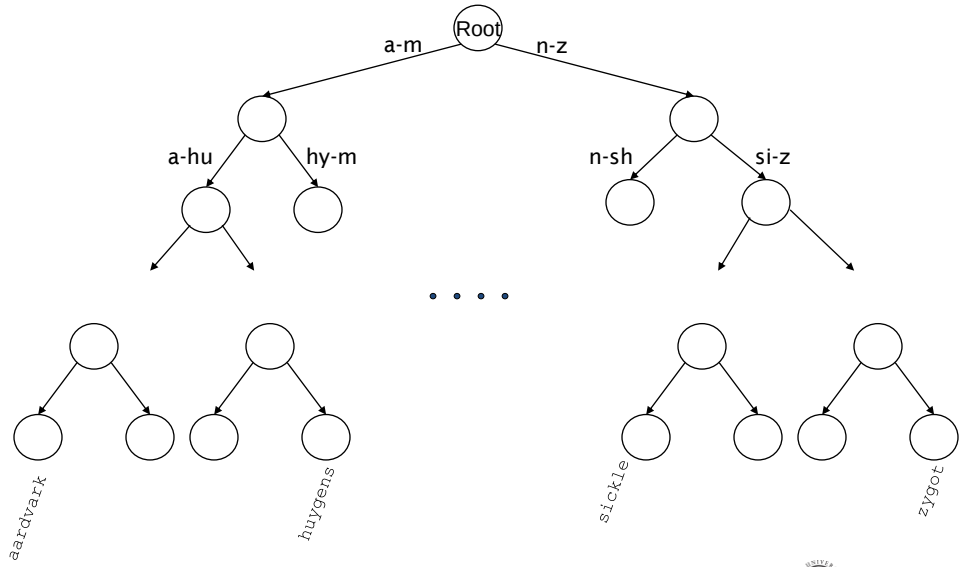
Dictionary Data Structures

- The dictionary data structure stores the term vocabulary, document frequency, pointers to each postings list …
- For small collections, load full dictionary in memory. In real-life, cannot load all index to memory!
  - Then what to load?
  - How to reach quickly?
  - What data structure to use for inverted index?

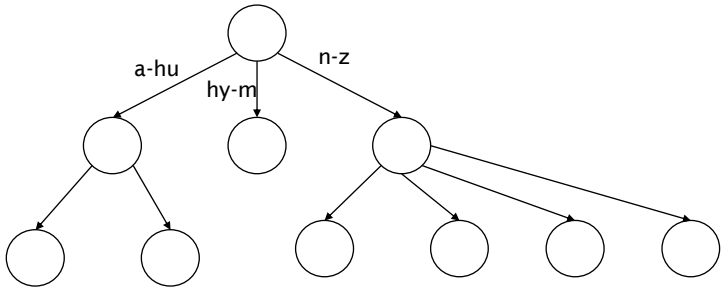
Hashes

- Each vocabulary term is hashed to an integer
- Pros
  - Lookup is faster than for a tree: O(1)
- Cons
  - No easy way to find minor variants:
    - judgment/judgement
  - No prefix search
  - If vocabulary keeps growing, need to occasionally do the expensive operation of rehashing everything

Trees: Binary Search Tree



Trees: B-tree



Every internal node has a number of children in the interval [a,b] where a, b are appropriate natural numbers, e.g., [2,4].

Trees

- Pros?
  - Solves the prefix problem (terms starting with “ab”)
- Cons?
  - Slower: O(log M) [and this requires balanced tree]
  - Rebalancing binary trees is expensive
    - But B-trees mitigate the rebalancing problem

Wild-Card Queries: *

- mon*: find all docs containing any word beginning "mon".
- Easy with binary tree (or B-tree) lexicon
- *mon: find words ending in "mon": harder
  - Maintain an additional B-tree for terms backwards.
- How can we enumerate all terms meeting the wild-card query pro*cent ?
- Query processing: se*ate AND fil*er ?
  - Expensive

Permuterm Indexes

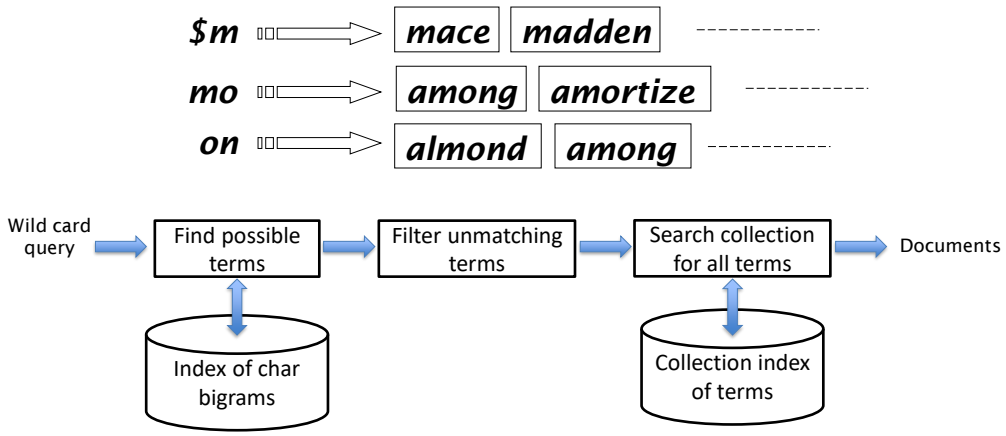
- Transform wild-card queries so that the * occurs at the end
- For term *hello*, index under:
  - hello$, ello$h, llohe, lohel, o$hell, $hello
    where $ is a special symbol.
- Rotate query wild-card to the end
- Queries:
  - X lookup on X$
  - X* lookup on $X*
  - *X lookup on
  - X*Y lookup on
- Index Size?

Character n-gram Indexes

- Enumerate all n-grams (sequence of *n* chars) occurring in any term
  - e.g., from text "*April is the cruelest month*" we get the 2-grams (bigrams) → a,ap,pr,ri,il,l,i,is,s,t,th,he,e,c,cr,ru,ue,el,le,es,st,t, m,mo,on,nt,h
  - $ is a special word boundary symbol
- Maintain a second inverted index from bigrams to dictionary terms that match each bigram.
  - Character n-grams → terms
  - terms → documents

Character n-gram Indexes

- The *n*-gram index finds *terms* based on a query consisting of *n*-grams (here *n*=2).



Character n-gram Indexes: Query time

- *Step 1*: Query **mon*** → **$m** AND **mo** AND **on**
  - It would still match **moon**.
- *Step 2*: Must post-filter these terms against query.
  - Phrase match, or post-step1 match
- *Step 3*: Surviving enumerated terms are then looked up in the term-document inverted index.
  → **Montreal** OR **monster** OR **monkey**
- Wild-cards can result in expensive query execution (very large disjunctions…)

Character n-gram Indexes: Applications

- Spelling Correction
  - Create n-gram representation for words
  - Build index for words:
    - Dictionary of words → documents (each word is a document)
    - Character n-grams → terms
  - When getting a search term that is misspelled (OOV or not frequent), find possible corrections
    - Possible corrections = most matching results

Query: elepgant → $e el le ep pg ga an nt t$
Results:
elegant → $e el le eg ga an nt t$
elephant → $e el le ep ph ha an nt t$

Character n-gram Indexes: Applications

- Char n-grams can be used as direct index terms for some applications:
  - Arabic IR, when no stemmer/segmenter is available
  - Documents with spelling mistakes: OCR documents
- Word char representation can by with multiple n's
  - "elephant" → 2/3-gram →
    "$e el le ep ph ha an nt t$ $el $ele lep eph pha han ant nt$"

The **children** behaved well
Her **children** are cute

الأبناء تصرفوا جيدا
أ**بناءها** لطاف

$ا ال لأ **أب بن نا اء** ء$
$أ **أب بن نا اء** ءه ها ا$

Document: Elepbant → $e el le ep pb ba an nt t$
Query: Elephant → $e el le ep ph ha an nt t$

Summary

- Index can by multilayer
  - Extent index (multi-terms in one position in document)
- Index does not have to be formed of words
  - Character n-grams representation of words
- Two indexes are sometimes used
  - Index of character n-grams to find matching words
  - Index of terms to search for matched words

Resources

- Text book 1: Intro to IR, Chapter 3.1 – 3.4
- Text book 2: IR in Practice, Chapter 5

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Ranked IR

Instructor:
Walid Magdy

13-Oct-2021

1

Lecture Objectives

- Learn about Ranked IR
  - TFIDF
  - VSM
  - SMART notation
- Implement:
  - TFIDF

Boolean Retrieval

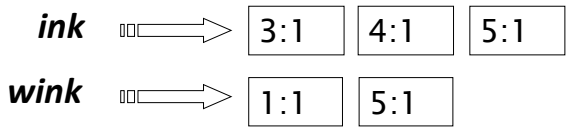
- Thus far, our queries have all been Boolean.
  - Documents either: “match” or “no match”.
- Good for expert users with precise understanding of their needs and the collection.
  - Patent search uses sophisticated sets of Boolean queries and check hundreds of search results
    (car OR vehicle) AND (motor OR engine) AND NOT (cooler)
- Not good for the majority of users.
  - Most incapable of writing Boolean queries.
  - Most don’t want to go through 1000s of results.
    - This is particularly true for web search
    - Question: What is the most unused web-search feature?

Ranked Retrieval

- Typical queries: free text queries
- Results are “ranked” with respect to a query
- Large result sets are not an issue
  - We just show the top k (≈ 10) results
  - We don’t overwhelm the user
- Criteria:
  - Top ranked documents are the most likely to satisfy user’s query
  - Score is based on how well documents match a query
    Score(*d*,*q*)

Old Example

- Find documents matching query {ink wink}
  1. Load inverted lists for each query word
  2. Merge two postings lists → **Linear merge**
- Apply function for matches
  - Boolean: exist / not exist = 0 or 1
  - Ranked: *f*(*tf*, *df*, *length*, ….) = 0 → 1



Matches

1: f(0,1)

3: f(1,0)

4: f(1,0)

5: f(1,1)

Function example: Jaccard coeffecient

- a commonly used measure of overlap of two sets *A* and *B*
- $jaccard(A,B) = \frac{|A \cap B|}{|A \cup B|}$
- $jaccard(A,A) = 1$
- $jaccard(A,B) = 0,$ if $A \cap B = 0$
- Example:
  - D1 ∪ D2 = {he, likes, to, wink, and, drink}
  - D1 ∩ D2 = {he, likes, to, drink}
  - $jaccard(D1,D2) = \frac{4}{6} = 0.6667$

D1: He likes to wink, he likes to drink

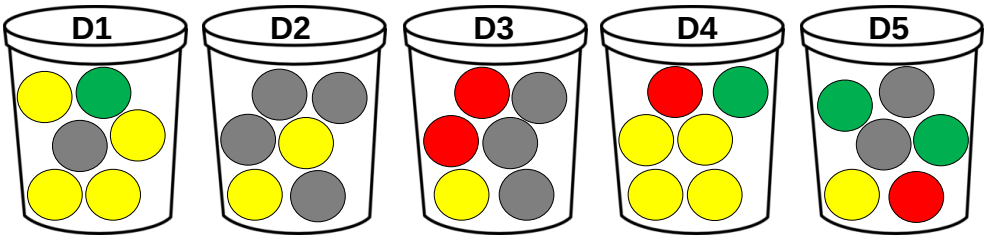
D2: He likes to drink, and drink, and drink

Jaccard coefficient: Issues

- Does not consider term frequency (how many times a term occurs in a document)
- It treats all terms equally!
  - How about rare terms in a collection? more informative than frequent terms.
  - *He likes to drink*, shall "to" == "drink"?
- Needs more sophisticated way of length normalization
  - |D1| = 3, |D2| = 1000!
  - D1 → Q, D2 → D

Should terms be treaded the same?

- Collection of 5 documents (balls = terms)
- Query 
- Which is the least relevant document?
- Which is the most relevant document?



TFIDF

- **TFIDF**:
  Term **F**requency, **I**nverse **D**ocument **F**requency
- ***tf(t,d)***:
  number of times term ***t*** appeared in document ***d***
  - As $tf(t,d)$ ↑↑ → importance of *t* in *d* ↑↑
  - Document about IR, contains "retrieval" more than others
- ***df(t)***:
  number of documents term ***t*** appeared in
  - As $df(d)$ ↑↑ → importance if *t* in a collection ↓↓
    - "the" appears in many document → not important
    - "FT" is not important word in financial times articles

DF, CF, & IDF

- **DF ≠ CF** (collection frequency)
  - ***cf(t)*** = total number of occurrences of term ***t*** in a collection
  - ***df(t)*** ≤ ***N*** (***N***: number of documents in a collection)
  - ***cf(t)*** can be ≥ ***N***
- **DF** is more commonly used in IR than **CF**
  - **CF** is still used
- ***idf(t)***: inverse of ***df(t)***
  - As $idf(t)$ ↑↑ → rare term → importance ↑↑
  - ***idf(t)*** → measure of the informativeness of *t*

DF vs CF

he	drink	ink	likes	pink	think	wink	
2	1	0	2	0	0	1	← **D1:** He likes to wink, he likes to drink
1	3	0	1	0	0	0	← **D2:** He likes to drink, and drink, and drink
1	1	1	1	0	1	0	← **D3:** The thing he likes to drink is ink
1	1	1	1	1	0	0	← **D4:** The ink he likes to drink is pink
1	1	1	1	1	0	1	← **D5:** He likes to wink, and drink pink ink
5	5	3	5	2	1	2	**DF**
6	7	3	6	2	1	2	**CF**

IDF: formula

$$idf(t) = log_{10}(\frac{N}{df(t)})$$

- Log scale used to dampen the effect of IDF
- Suppose N = 1 million →

term	*df(t)*	*idf(t)*
calpurnia	1	**6**
animal	100	**4**
sky	1,000	**3**
fly	10,000	**2**
under	100,000	**1**
the	1,000,000	**0**

TFIDF term weighting

- One the best known term weights schemes in IR
  - Increases with the number of occurrences within a document
  - Increases with the rarity of the term in the collection
- Combines TF and IDF to find the weight of terms

$$w_{t.d} = \left(1 + log_{10} tf(t,d)\right) \times log_{10}\left(\frac{N}{df(t)}\right)$$

- For a query *q* and document *d*, retrieval score *f(q,d)*:

$$Score(q,d) = \sum_{t \in q \cap d} w_{t.d}$$

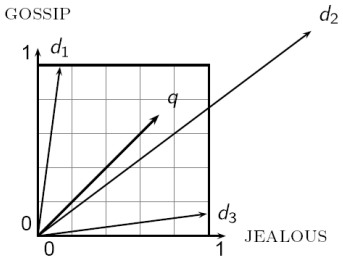
Document/Term vectors with tfidf

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

→ **Vector Space Model**

Vector Space Model

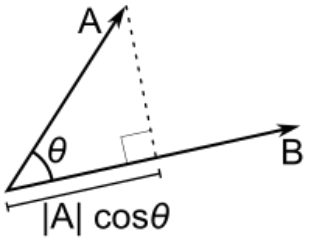
- Documents and Queries are presented as vectors
- Match (Q,D) = Distance between vectors
- Example: Q= Gossip Jealous
- Euclidean Distance?
  Distance between the endpoints of the two vectors
- Large for vectors of diff. lengths



- Take a document d and append it to itself. Call this document d′.
  - "Semantically" d and d′ have the same content
  - Euclidean distance can be quite large

Angle Instead of Distance

- The angle between the two documents is 0, corresponding to maximal similarity.
- Key idea: Rank documents according to angle with query.
  - Rank documents in increasing order of the angle with query
  - Rank documents in decreasing order of cosine (query, document)
- Cosine of angle = projection of one vector on the other



Length Normalization

- A vector can be normalized by dividing each of its components by its length – for this we use the L_2 norm:

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

- Dividing a vector by its L_2 norm makes it a unit (length) vector (on surface of unit hypersphere)
- Effect on the two documents d and d′ (d appended to itself) from earlier slide: they have identical vectors after length-normalization.
  - Long and short documents now have comparable weights

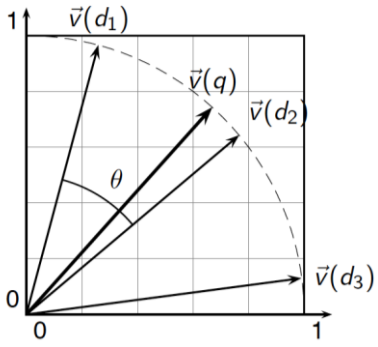
Example

- D1 = $\begin{bmatrix}1\\3\\2\end{bmatrix}$ ➔ $\left\|\overrightarrow{D1}\right\|_2 = \sqrt{1+9+4} = 3.74$
- $D1_{normalized}$ = $\begin{bmatrix}0.267\\0.802\\0.535\end{bmatrix}$
- D2 = $\begin{bmatrix}3\\9\\6\end{bmatrix}$ ➔ $\left\|\overrightarrow{D1}\right\|_2 = \sqrt{9+81+36} = 11.25$
- $D2_{normalized}$ = $\begin{bmatrix}0.267\\0.802\\0.535\end{bmatrix}$

Cosine "Similarity" (Query, Document)

- $\vec{q}_i$ is the tf-idf weight of term i in the query
- $\vec{d}_i$ is the tf-idf weight of term i in the document
- For normalized vectors:

$$\cos(\vec{q},\vec{d}) = \vec{q}\cdot\vec{d} = \sum_{i=1}^{|V|} q_i d_i$$



- For non-normalized vectors:

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{\|\vec{q}\|\|\vec{d}\|} = \frac{\vec{q}}{\|\vec{q}\|}\cdot\frac{\vec{d}}{\|\vec{d}\|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

Algorithm

CosineScore(q)

```
1  float Scores[N] = 0
2  float Length[N]
3  for each query term t
4  do calculate w_{t,q} and fetch postings list for t
5      for each pair(d, tf_{t,d}) in postings list
6      do Scores[d]+ = w_{t,d} × w_{t,q}
7  Read the array Length
8  for each d
9  do Scores[d] = Scores[d]/Length[d]
10 return Top K components of Scores[]
```

TFIDF Variants

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1+\log(\mathrm{tf}_{t,d})$	t (idf)	$\log\frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5+\frac{0.5\times\mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0,\log\frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases}1 & \text{if } \mathrm{tf}_{t,d}>0\\ 0 & \text{otherwise}\end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha<1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

- Many search engines allow for different weightings for queries vs. documents
- **SMART** Notation: use notation *ddd.qqq*, using the acronyms from the table
- A very standard weighting scheme is: *lnc.ltc*

For Lab and CW

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1+\log(\mathrm{tf}_{t,d})$	t (idf)	$\log\frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5+\frac{0.5\times\mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0,\log\frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases}1 & \text{if } \mathrm{tf}_{t,d}>0\\ 0 & \text{otherwise}\end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha<1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

"OR" operator, then:

$$Score(q,d)=\sum_{t\in q\cap d}\left(1+log_{10}tf(t,d)\right)\times log_{10}\left(\frac{N}{df(t)}\right)$$

Summary of Steps:

- Represent the query as a weighted *tf-idf* vector
- Represent each document as a weighted *tf-idf* vector
- Compute the cosine similarity score for the query vector and each document vector
- Rank documents with respect to the query by score
- Return the top K (e.g., $K = 10$) to the user

Retrieval Output

- For a query q_1, the output would be a list of documents ranked according to the $score(q_1,d)$
- Possible output format:

```
1,   710,   0.9234
1,   213,   0.7678
1,   103,   0.6761
1,   13,    0.6556
1,   501,   0.4301
```

Query id

document id

score

Resources

- Text book 1: Intro to IR, Chapter 6.2 → 6.4
- Text book 2: IR in Practice, Chapter 7

- Lab 3

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Ranked Retrieval (2)

Instructor:
Walid Magdy

13-Oct-2021

1

Lecture Objectives

- Learn about Probabilistic models
  - BM25
- Learn about LM for IR

Recall: VSM & TFIDF term weighting

- Combines TF and IDF to find the weight of terms

$$w_{t.d} = \left(1 + log_{10} tf(t,d)\right) \times log_{10}\left(\frac{N}{df(t)}\right)$$

- For a query *q* and document *d*, retrieval score *f(q,d)*:

$$Score(q,d) = \sum_{t \in q \cap d} w_{t.d}$$

- TFIDF observations **Can we do better?**
  - Term appearing more in a doc gets higher weight (TF)
  - First occurrence is more important (log)
  - Rare terms are more important (IDF)
  - Bias towards longer documents

IR Model

- VSM is very heuristic in nature
  - No notion of relevance is there (still works well)
  - Any weighting scheme, similarity measure can be used
    - Components not interpretable → no guide for what to try next
    - More engineering rather than theory → tweak, run, observe, tweak …
  - Very popular, hard to beat, strong baseline
    - Easy to adapt good ideas from other models
- **Probabilistic Model** of retrieval
  - Mathematical formulisation for relevant / irrelevant sets
    - Explicitly defines random variables (R,Q,D)
    - Specific about what their values are
    - State the assumptions behind each step
    - Watch out for contradictions

Probabilistic Models

- Concept: Uncertainty is inherent part of IR process
- Probability theory is strong foundation for representing and manipulating uncertainty
- Probability Ranking Principle (1977)

Stephan Robertson

Probability Ranking Principle

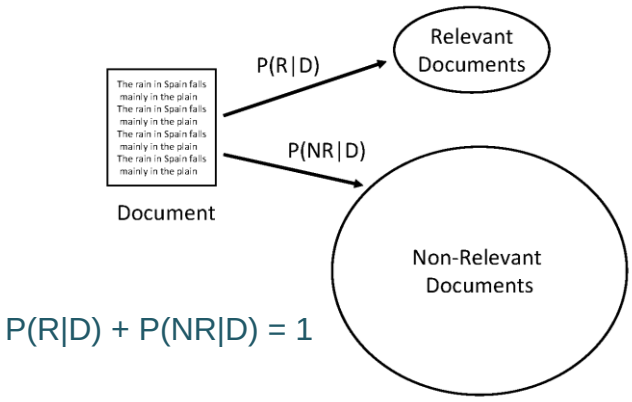
- "If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request,
- where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose,
- the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data."
- Basis for most probabilistic approaches for IR

Formulation of PRP

- Rank docs by probability of relevance
  - $P(R|D_{r1}) > P(R|D_{r2}) > P(R|D_{r3}) > P(R|D_{r4}) > \ldots$
- Estimate probability as accurate as possible
  - $P_{est}(R|D) \approx P_{true}(R|D)$
- Estimate with all possibly available data
  - $P_{est}(R \mid \text{doc, session, context, user profile, } \ldots)$
- Best possible accuracy can be achieved with that data
  - → the perfect IR system
  - Is it really doable?
- **How to estimate the probability of relevance?**

PRP Concept

- Imagine IR as a classification problem



P(R|D) + P(NR|D) = 1

- Document *D* is **relevant** if ***P(R|D) > P(NR|D)***

Probability of Relevance

- What is P_{true}(rel | doc, query, session, user, …)?
  - Isn't relevance just the user's opinion?
    - User decides relevant or not, what is the "probability" thing?
- Search algorithm cannot look into your head (yet!)
  - Relevance depends on factors that algorithm cannot observe
    - SIGIR 2016 best paper award: *Understanding Information Need: an fMRI Study*
- Different users may disagree on relevance of the same doc
  - Even similar users, doing the same task, in the same context
- P_{true} (rel | Q, D):
  - Proportion of all unseen users / context / tasks for which D would have judged relevant to Q
- Similar to: P(die=6 | even and not square)

Okapi BM25 Model

- Based on the probabilistic model
  - A document D is relevant if **P(R=1|D) > P(R=0|D)**
- Extension to the "binary independence model"
  - **Binary features**: Document represented by a vector of binary features indicating term occurrence
  - Assume **term independence** (Naïve Bayes assumption) → BOW trick
- In 1995, *Stephan Robertson* with his group came up with the **BM25** Formula as part of the **Okapi** project.
- It outperformed all other systems in TREC
- Popular and effective ranking algorithm

Okapi BM25 Ranking Function

- Let L_d be the number of terms in document d
- Let $\bar{L}$ be the average number of terms in a document

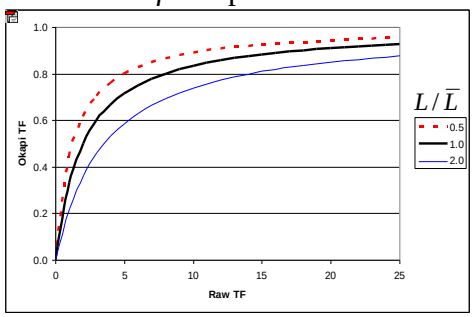
$$w_{t.d} = \frac{tf_{t,d}}{k.\frac{L_d}{\bar{L}} + tf_{t,d} + 0.5} \times log_{10}\left(\frac{N - df_t + 0.5}{df_t + 0.5}\right)$$

- Best practices: k=1.5

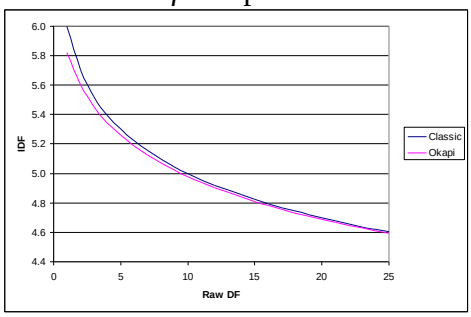
Okapi BM25 Ranking Function

$$w_{t.d} = \frac{tf_{t,d}}{1.5\frac{L_d}{\bar{L}} + tf_{t,d} + 0.5} \times log_{10}\left(\frac{N - df_t + 0.5}{df_t + 0.5}\right)$$

tf component



idf component

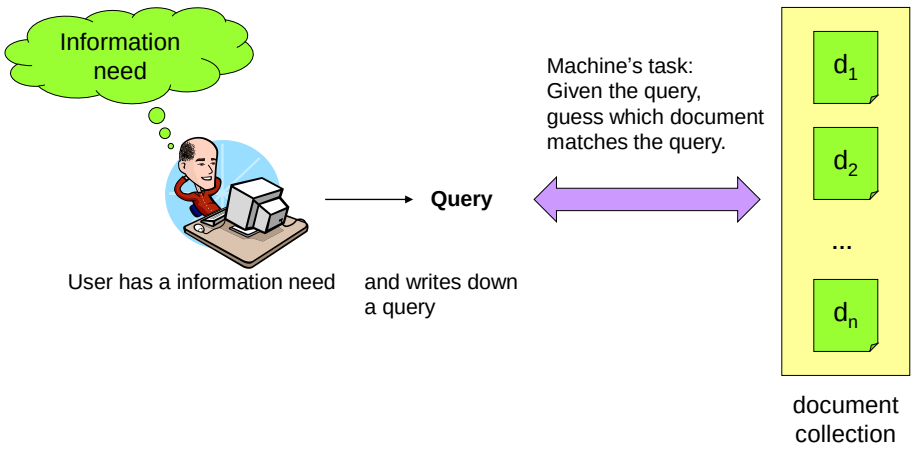


Probabilistic Model in IR

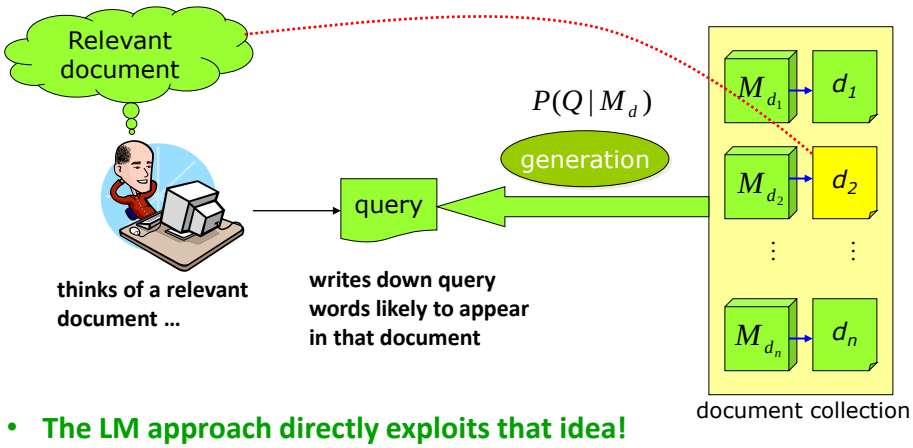
- Focuses on the probability of relevance of docs
- Could be mathematically proved
- Different ways to apply it
- BM25 is the most common formula for it

- What other models could be still used in IR?

"Noisy-Channel" Model of IR



IR based on Language Model (LM)



- **The LM approach directly exploits that idea!**
  - a document is a good match to a query if the document model is likely to generate the query

Concept

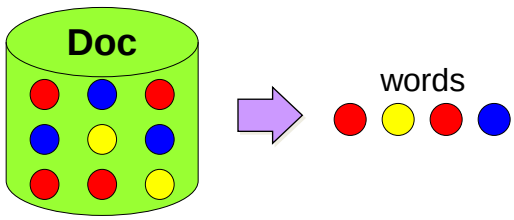
- Coming up with good queries?
  - Think of words that would likely appear in a relevant doc
  - Use those words as the query
- The language modeling approach to IR directly models that idea
  - a document is a good match to a query if the document model is likely to generate the query
    - happens if the document contains the query words often.
- Build a probabilistic language model M_d from each document d
- Rank documents based on the probability of the model generating the query: $P(q|M_d)$.

Language Model (LM)

- A language model is a probability distribution over strings drawn from some vocabulary
- A topic in a document or query can be represented as a language model
  - i.e., words that tend to occur often when discussing a topic will have high probabilities in the corresponding language model

Unigram LM

- Terms are randomly drawn from a document (with replacement)



$$P(\bullet\ \circ\ \bullet\ \bullet) = P(\bullet) \times P(\circ) \times P(\bullet) \times P(\bullet)$$
$$= (4/9) \times (2/9) \times (4/9) \times (3/9)$$

Example

w	$P(w\|q_1)$	w	$P(w\|q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
		...	...

- This is a one-state probabilistic finite-state automaton – a unigram language model.
- S = "frog said that toad likes frog STOP"
  P(S) = 0.01 × 0.03 × 0.04 × 0.01 × 0.02 × 0.01 × 0.02
  = 0.0000000000048

Comparing LMs

- **M_{d1}**
  LM generated from Doc 1
- **M_{d2}**
  LM generated from Doc 2
- Try to generate sentence **S** from **M_{d1}** & **M_{d2}**

Model M_{d1}

P(w)	w
0.2	the
0.0001	yon
0.01	class
0.0005	maiden
0.0003	sayst
0.0001	pleaseth
…	

Model M_{d2}

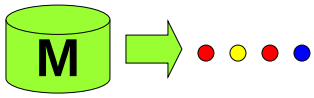
P(w)	w
0.2	the
0.1	yon
0.001	class
0.01	maiden
0.03	sayst
0.02	pleaseth
…	

text:	the	class	pleaseth	yon	maiden	P(S)
M_{d1}:	0.2	0.01	0.0001	0.0001	0.0005	0.00000000000001
M_{d2}:	0.2	0.001	0.02	0.1	0.01	0.000000004

$P(text|M_{d2}) > P(text|M_{d1})$

Stochastic Language Models

- A statistical model for generating text
  - Probability distribution over strings in a given language



P (● ○ ● ● | M) = P (● | M)
P (○ | M, ●)
P (● | M, ● ○)
P (● | M, ● ○ ●)

Unigram and Higher-order LM

P (● ○ ● ●)
= P (●) P (○ | ●) P (● | ● ○) P (● | ● ○ ●)

- **Unigram Language Models**

  P (●) P (○) P (●) P (●)

- **Bigram** (generally, *n*-gram) Language Models

  P (●) P (○ | ●) P (● | ○) P (● | ●)

LM in IR

- Each document is treated as basis for a LM.
- Given a query q, rank documents based on *P(d|q)*

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)}$$

  - *P(q)* is the same for all documents ➔ ignore
  - *P(d)* is the prior – often treated as the same for all *d*
    - But we can give a prior to "high-quality" documents, e.g., those with high PageRank (later to be discussed).
  - *P(q|d)* is the probability of *q* given *d*.
- So to rank documents according to relevance to q, ranking according to *P(q|d)* and *P(d|q)* is equivalent

LM in IR: Basic idea

- We attempt to model the query generation process.
- Then we rank documents by the probability that a query would be observed as a random sample from the respective document model.
- That is, we rank according to ***P(q|d)***.

P(q|d)

Query Likelihood Model

- We will make the conditional independence assumption.

$$P(q|M_d) = P(\langle t_1, \dots, t_{|q|} \rangle | M_d) = \prod_{1 \le k \le |q|} P(t_k|M_d)$$

$|q|$: length of q; t_k : token occurring at position k in q

- This is equivalent to:

$$P(q|M_d) = \prod_{each\ term\ t\ in\ q} P(t|M_d)^{tf_{t,q}}$$

$tf_{t,q}$: term frequency (# occurrences) of t in q

- Multinomial model (omitting constant factor)

Parameter estimation

- Probability of a term t in a LM M_d using Maximum Likelihood Estimation (MLE)

$$P(t|M_d) = \frac{tf_{t,d}}{|d|}$$

$|d|$: length of d;
$tf_{t,d}$: # occurrences of t in d

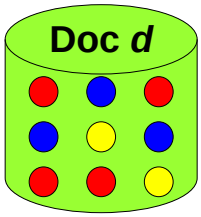
- Probability of a query q to be noticed in a LM M_d:

$$P(q|M_d) = \prod_{\forall\, t \in q} \left(\frac{tf_{t,d}}{|d|} \right)^{tf_{t,q}}$$

Example

P (● ○ ● ●) = P (●)^2 × P (○) × P (●)
= (4/9)^2 × (2/9) × (3/9) = 0.0146

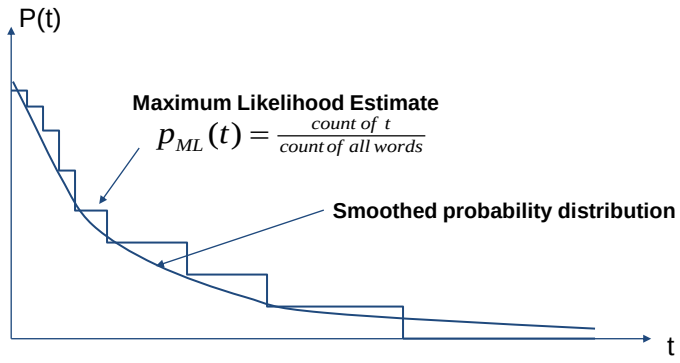
P (● ○ ● ●)



- Is that fair?
  - In VSM, S(Q,D) was summation, works more like OR in Boolean search. Missing one term reduces score only
  - In language model, S(Q,D) is P(Q|D) → Multiplication of probabilities → missing one term makes score = 0
  - Is there a better way to handle unseen terms?

Smoothing

- Problem: Zero frequency
- Solution: "Smooth" terms probability



Smoothing

- Document texts are a sample from the language model
- Missing words should not have zero probability of occurring
- A missing term is possible (even though it didn't occur)
  - but no more likely than would be expected by chance in the collection.
- A technique for estimating probabilities for missing (or unseen) words
  - Overcomes data-sparsity problem
  - lower (or discount) the probability estimates for words that are seen in the document text
  - assign that "left-over" probability to the estimates for the words that are not seen in the text (and also on the seen ones)

Mixture Model

$$P(t|d) = \lambda P(t|M_d) + (1 - \lambda)P(t|M_c)$$

- Mixes the probability from the document with the general collection frequency of the word.
- Estimate for unseen words is $(1-\lambda)\ P(t|M_c)$
  - Based on collection language model (background LM)
  - $P(t|M_c)$ is the probability for query word i in the collection language model for collection C (background probability)
  - λ is a parameter controlling probability for unseen words
- Estimate for observed words is

$$\lambda\ P(t|M_d) + (1-\lambda)\ P(t|M_c)$$

CF

Jelinek-Mercer Smoothing

$$P(t|d) = \lambda P(t|M_d) + (1 - \lambda)P(t|M_c)$$

- **High value of λ**: "conjunctive-like" search – tends to retrieve documents containing all query words.
- **Low value of λ**: more disjunctive, suitable for long queries
- Correctly setting λ is important for good performance.

- Final Ranking function:

$$P(q|M_d) \propto \prod_{1 \le k \le |q|} \left(\lambda \cdot P(t_k|M_d) + (1 - \lambda) \cdot P(t_k|M_c)\right)$$

Example

- **Collection**: d_1 and d_2
- ***d*₁** : "Jackson was one of the most talented entertainers of all time"
- ***d*₂**: "Michael Jackson anointed himself King of Pop"
- **Query *q***: Michael Jackson
- Use mixture model with λ = 1/2

- $P(q|d_1) = [(0/11 + 1/18)/2] \cdot [(1/11 + 2/18)/2] \approx 0.003$
- $P(q|d_2) = [(1/7 + 1/18)/2] \cdot [(1/7 + 2/18)/2] \approx 0.013$
- Ranking: $d_2 > d_1$

Notes on Query Likelihood Model

- It has similar effectiveness to BM25
- With more sophisticated techniques, it outperforms BM25
  - Topic models
- There are several alternative smoothing techniques
  - That was just an example

n-grams LMs

- Unigram language model
  - probability distribution over the words in a language
    - associates a probability of occurrence with every word
  - generation of text consists of pulling words out of a "bucket" according to the probability distribution and replacing them

- N-gram language model
  - some applications use bigram and trigram language models where probabilities depend on previous words
  - predicts a word based on the previous n-1 words

LMs for IR: 3 possibilities

- Probability of generating the query text from a document language model
- Probability of generating the document text from a query language model
- Comparing the language models representing the query and document topics

Summary

- **Three ways to model IR**
- VSM
  How query vector aligns with document vector?
- Probabilistic Model
  What is the relevance probability of document D given query Q?
- LM
  How likely is it possible to observe/generate sequence of terms Q in a language model of document D?

Resources

- Text book 1: Intro to IR, Chapter 12
- Text book 2: IR in Practice, Chapter 7.2, 7.3
- Readings:
  - *Robertson, Stephen E., et al.*
    *"**Okapi at TREC-3**."*
    Nist Special Publication Sp 109 (1995): 109.
  - *J. Ponte and W. B. Croft.*
    A language modeling approach toinformation retrieval.
    In Proceedings on the 21st annualinternational ACM SIGIR conference, pages 275–281, 1998

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

IR Evaluation

Instructor:
Walid Magdy

20-Oct-2021

1

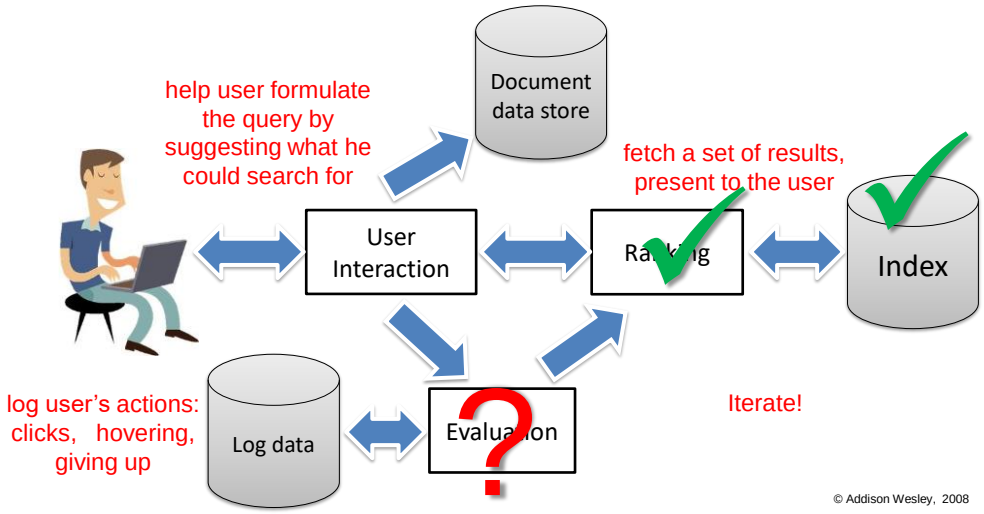
Pre-lecture

- Sorry for having last week's lectures recorded
- Discussion session about lectures 7 & 8 available on Learn
- Test collection for CW1 to be released next week
- New CW1 deadline: 31 Oct 2021
- No new lab this week (support to continue for previous labs)

Lecture Objectives

- Learn about how to evaluate IR
  - Evaluation measures
  - P, R, F
  - MAP
  - nDCG
- Implement: (as part of CW2)
  - P, R
  - MAP
  - nDCG

Search Process



IR as an Experimental Science!

- Formulate a research question: the hypothesis
- Design an experiment to answer the question
- Perform the experiment
  - Compare with a baseline "control"
- Does the experiment answer the question?
  - Are the results significant? Or is it just luck?
- Report the results!
- Iterate …
- e.g. stemming improves results? (university → univers)

Lab 3 output

Is that a good performance?

1, 65, 4.8040	2, 3549, 7.0396	3, 3354, 4.6113
1, 3533, 4.7264	2, 305, 6.8394	3, 3345, 4.5087
1, 3562, 3.5454	2, 288, 6.6742	3, 268, 3.6606
1, 3608, 3.4910	2, 223, 6.1252	3, 328, 3.4825
1, 141, 3.3262	2, 219, 4.8626	3, 21, 3.3984
1, 361, 3.3262	2, 3762, 4.8626	3, 304, 3.3722
1, 92, 3.2311	2, 3663, 4.5415	3, 313, 3.3436
1, 3829, 3.1818	2, 3766, 3.9924	3, 3790, 3.1796
1, 3420, 3.1273	2, 188, 3.8844	3, 55, 3.0462
1, 3734, 3.0561	2, 3360, 3.0988	3, 217, 2.8492
1, 3387, 2.9626	2, 3408, 3.0315	3, 361, 2.8348
1, 3599, 2.9626	2, 3390, 2.8498	3, 3789, 2.7158

Configure your system

- **About the system**:
  - Stopping? Tokenise? Stemming? n-gram char?
  - Use synonyms improve retrieval performance?
- Corresponding experiment?
  - Run your search for a set of queries with each setup and find which one will achieve the best performance
- **About the user**:
  - Is letting users weight search terms a good idea?
- Corresponding experiment?
  - Build two different interfaces, one with term weighting functionality, and one without; run a user study

Types of Evaluation Strategies

- **System-centered studies**:
  - Given documents, queries, and relevance judgments
  - Try several variations of the system
  - Measure which system returns the "best" hit list
  - Laboratory experiment
- **User-centered studies**
  - Given several users, and at least two retrieval systems
  - Have each user try the same task on both systems
  - Measure which system works the "best"

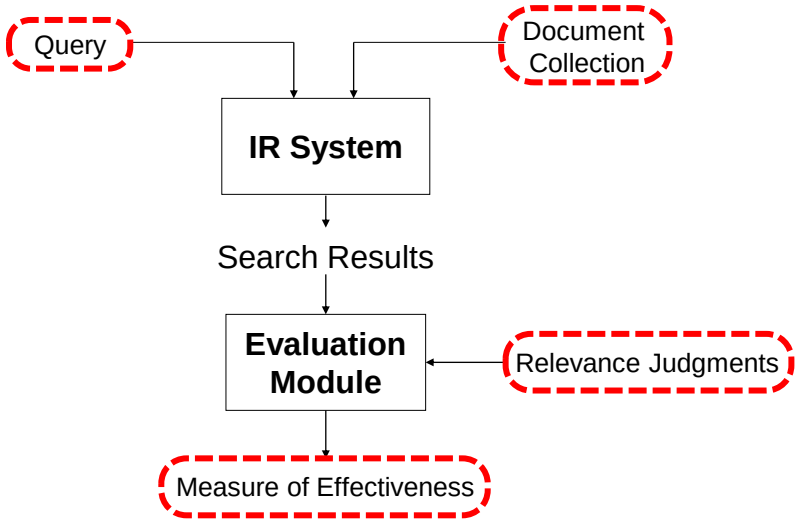
Importance of Evaluation

- The ability to measure differences underlies experimental science
  - How well do our systems work?
  - Is A better than B?
  - Is it really?
  - Under what conditions?
- Evaluation drives what to research
  - Identify techniques that work and don't work

The 3-dimensions of Evaluation

- **Effectiveness**
  - How "good" are the documents that are returned?
  - System only, human + system
- **Efficiency**
  - Retrieval time, indexing time, index size
- **Usability**
  - Learnability, flexibility
  - Novice vs. expert users

Cranfield Paradigm (Lab setting)



Reusable IR Test Collection

- **Collection of Documents**
  - Should be "representative" to a given IR task
  - Things to consider: size, sources, genre, topics, …
- **Sample of information need**
  - Should be "randomized" and "representative"
  - Usually formalized **topic** statements (query + description)
- **Known relevance judgments**
  - Assessed by humans, for each topic-document pair
  - Binary/Graded
- **Evaluation measure**

Good Effectiveness Measures

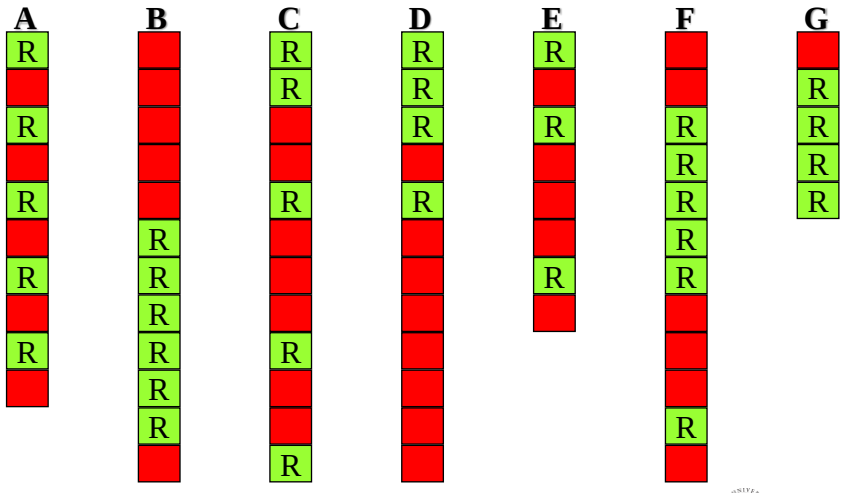
- Should capture some aspect of what the user wants
  - IR → Do the results satisfy user's information need?
- Should be easily replicated by other researchers
- Should be easily comparable
  - Optimally, expressed as a single number
    - Curves and multiple numbers are still accepted, but single numbers are much easier for comparison
- Should have predictive value for other situations
  - What happens with different queries on a different document collection?

Set Based Measures

- Assuming IR system returns sets of retrieved results without ranking
- Suitable with Boolean Search
- No certain number of results per query

Which looks the best IR system?

- For query **Q**, collection has **8 relevant documents**:

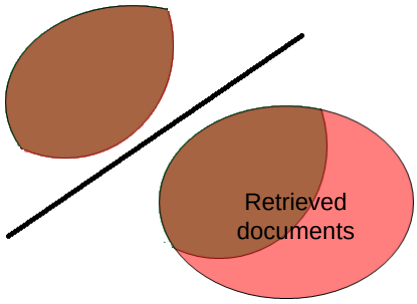


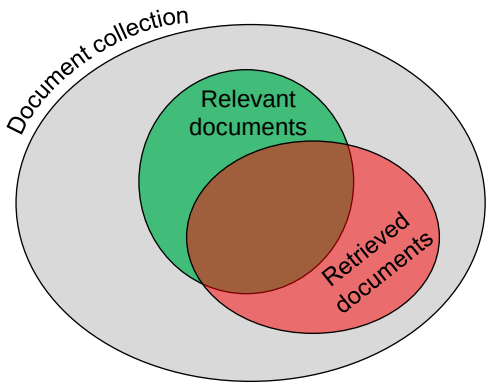
15

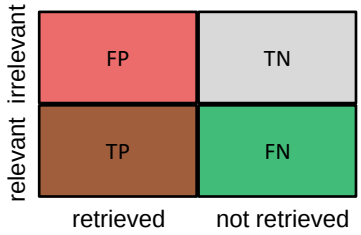
Precision and Recall

- **Precision**:
  What fraction of these retrieved docs are relevant?

$$P = \frac{rel \cap ret}{retrieved} = \frac{TP}{TP + FP}$$





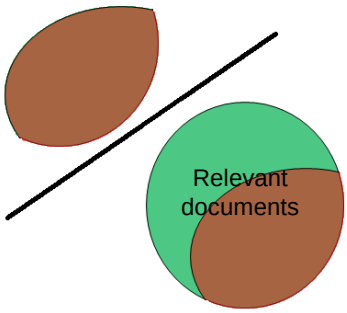


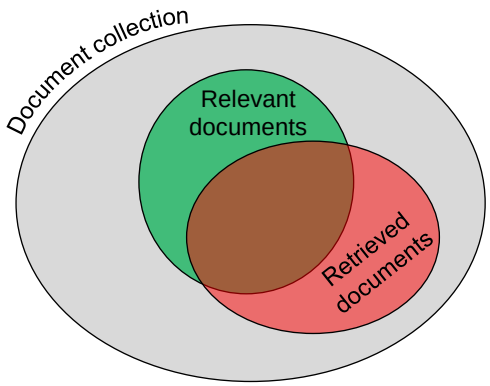
16

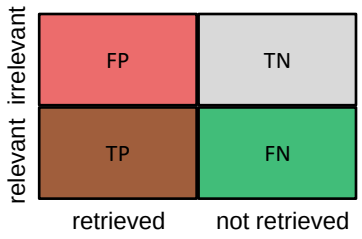
Precision and Recall

- **Recall**:
  What fraction of the relevant docs were retrieved?

$$R = \frac{rel \cap ret}{relevant} = \frac{TP}{TP + FN}$$

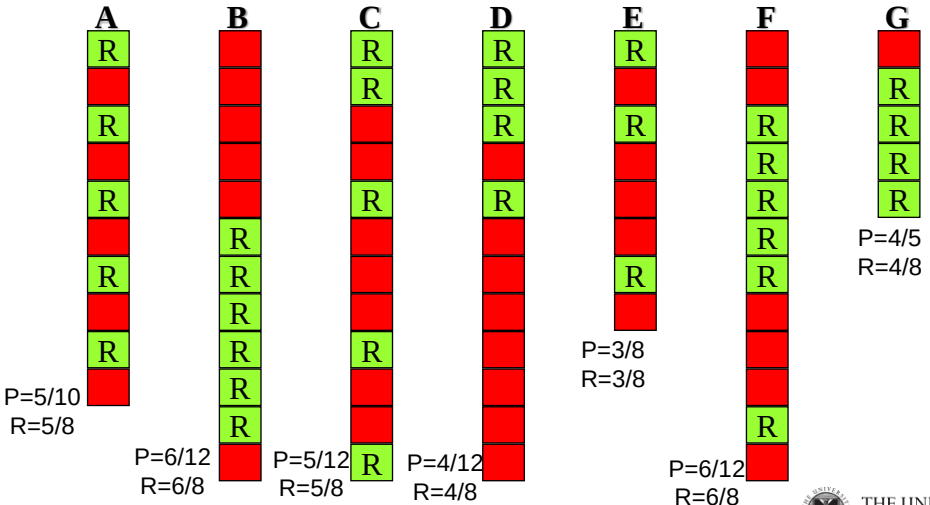






Which looks the best IR system?

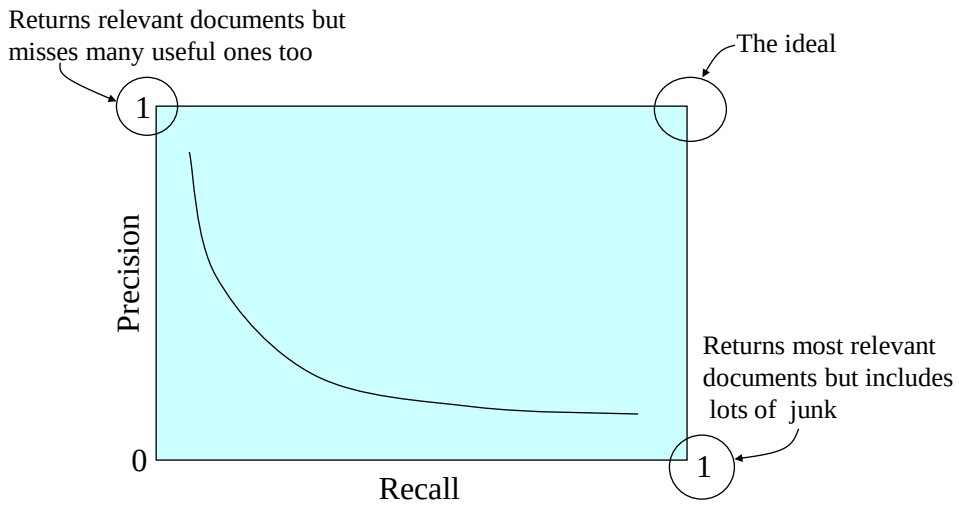
- For query **Q**, collection has **8 relevant documents**:



Trade-off between P & R

- Precision: The ability to retrieve top-ranked docs that are mostly relevant.
- Recall: The ability of the search to find all of the relevant items in the corpus.
- Retrieve more docs:
  - Higher chance to find all relevant docs → R ↑↑
  - Higher chance to find more irrelevant docs → P ↓↓

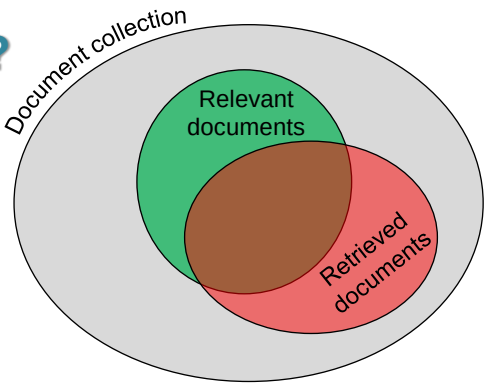
Trade-off between P & R

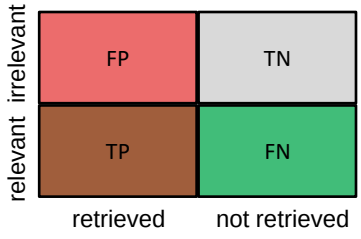


What about Accuracy?

- **Accuracy**:
  What fraction of docs was classified correctly?

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





One Measure? F-measure

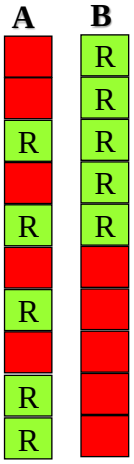
$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

$$F_\beta = \frac{(\beta^2 + 1)P \cdot R}{\beta^2 P + R}$$

- Harmonic mean of recall and precision
  - Emphasizes the importance of small values, whereas the arithmetic mean is affected more by outliers that are unusually large
- Beta (β) controls relative importance of P and R
  - β = 1, precision and recall equally important → *F*1
  - β = 5, recall five times more important than precision

Rank-based IR measures

- Consider systems A & B
  - Both retrieved 10 docs, only 5 are relevant
  - P, R & F are the same for both systems
    - Should their performances considered equal?
- Ranked IR requires taking "ranks" into consideration!
- How to do that?



Which is the best ranked list?

- For query **Q**, collection has **8 relevant documents**:

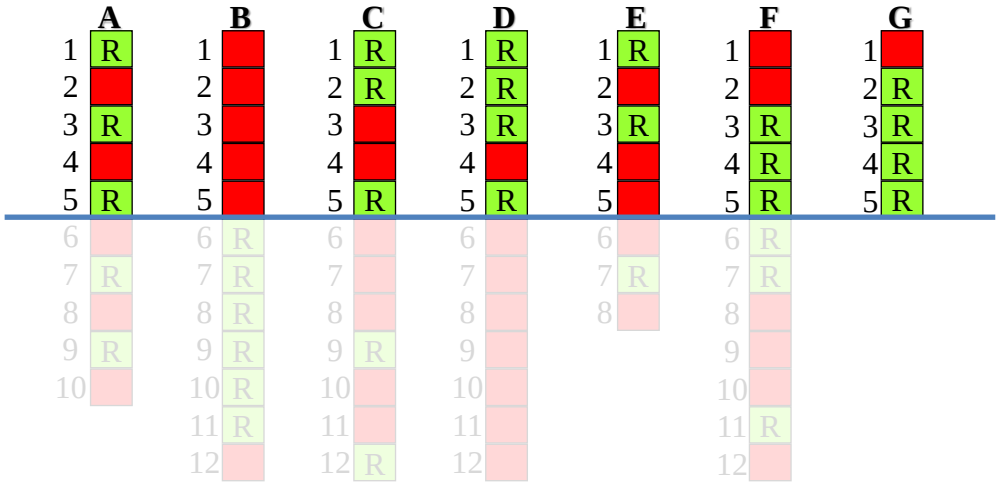
	A	B	C	D	E	F	G
1	R		R	R	R		
2			R	R			R
3	R			R	R	R	R
4						R	R
5	R		R	R		R	R
6		R				R	
7	R	R			R	R	
8		R					
9	R	R	R				
10		R					
11		R				R	
12			R				

Precision @ K

- *k* (a fixed number of documents)
- Have a cut-off on the ranked list at rank *k*, then calculate precision!
- Perhaps appropriate for most of web search: most people only check the top *k* results
- But: averages badly, Why?

P@5

- For query **Q**, collection has **8 relevant documents**:



R-Precision

- For a query with known *r* relevant documents
  → R-precision is the precision at rank *r* (P@*r*)
- *r* is different from one query to another
- Concept:
  It examines the ideal case: getting all relevant documents in the top ranks
- Is it realistic?

R-Precision

- For query **Q**, collection has **8 relevant documents**:

	A	B	C	D	E	F	G
1	R		R	R	R		
2			R	R			R
3	R			R	R	R	R
4						R	R
5	R		R	R		R	R
6		R				R	
7	R	R			R	R	
8		R					
9	R	R	R				
10		R					
11		R				R	
12			R				

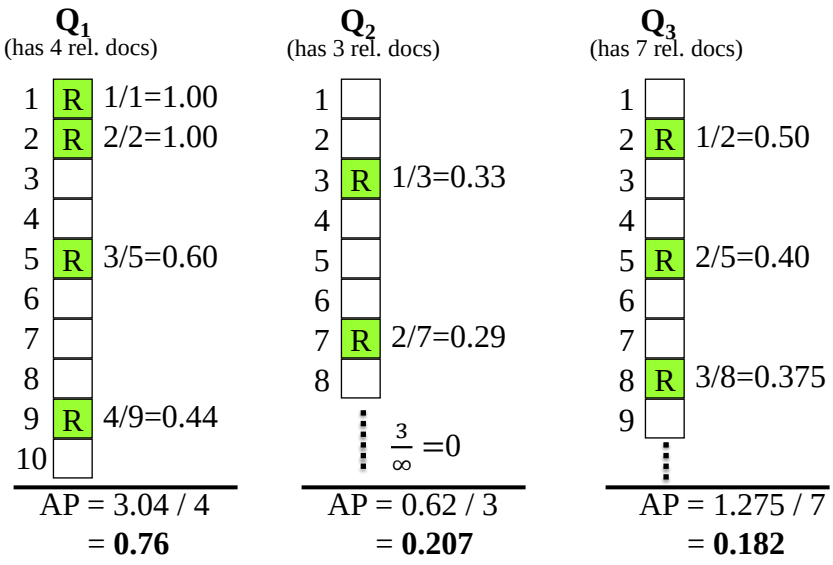
User Satisfaction??

- It is assumed that users needs to find relevant docs at the highest possible ranks
  → Precision is a good measure
- But, user would cut-off (stop inspecting results) at some point, say rank x
  →P@x
- What is the optimal x?
  When you think a user can stop?

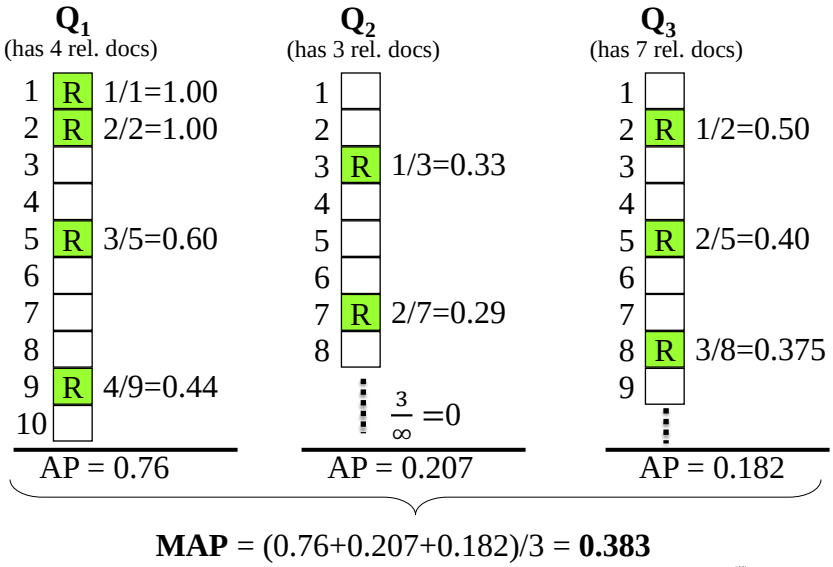
When a user can stop?

- IR objective: “satisfy user information need”
- Assumption: a user will stop once his/her information need is satisfied
- How? user will keep looking for relevant docs in the ranked list, read them, then stop once he/she feels satisfied
- P@x →x can be any rank where a relevant document appeared (*assume uniform distribution*)
- What about calculating the averages over all x’s?
  - every time you find relevant doc, calculate P@x, then take the average at the end

Average Precision (AP)



Mean Average Precision (MAP)



AP & MAP

$$AP = \frac{1}{r}\sum_{k=1}^{n} P(k) \times rel(k)$$

where, r: number of relevant docs for a given query
n: number of documents retrieved
$P(k)$ precision @ k
$rel(k)$: 1 if retrieved doc @ k is relevant, 0 otherwise.

$$MAP = \frac{1}{Q}\sum_{q=1}^{Q} AP(q)$$

where, Q: number of queries in the test collection

AP/MAP

$$AP = \frac{1}{r}\sum_{k=1}^{n} P(k) \times rel(k)$$

- A mix between precision and recall
- Highly focus on finding relevant document as early as possible
- When r=1 → MAP = MRR (mean reciprocal rank $\frac{1}{k}$)
- MAP is the most commonly used evaluation metric for most IR search tasks
- Uses binary relevance: rel = 0/1

Binary vs. Graded Relevance

- Some docs are more relevant to a query than other relevant ones!
  - We need non-binary relevance
- Binary Relevance:
  - Relevant 1
  - Irrelevant 0
- Graded Relevance:
  - Perfect 4
  - Excellent 3
  - Good 2
  - Fair 1
  - Bad 0

Binary vs. Graded Relevance

- Two assumptions:
  - Highly relevant documents are more useful than marginally relevant
  - The lower the ranked position of a relevant document, the less useful it is for the user, since it is less likely to be examined
- Discounted Cumulative Gain (DCG)
  - Uses graded relevance as a measure of the usefulness
  - The most popular for evaluating web search

Discounted Cumulative Gain (DCG)

- Gain is accumulated starting at the top of the ranking and may be reduced (*discounted*) at lower ranks
- Users care more about high-ranked documents, so we discount results by $1/\log_2(rank)$
  - the discount at rank 4 is 1/2, and at rank 8 is 1/3
- DCG_k is the total gain accumulated at a particular rank k (sum of DG up to rank k):

$$DCG_k = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{\log_2(i)}$$

0, 1, 2, 3, … (graded)

DCG

k	G	DG	DCG@k
1	3	3	3
2	2	2	5
3	3	1.89	6.89
4	0	0	6.89
5	0	0	6.89
6	1	0.39	7.28
7	2	0.71	7.99
8	2	0.67	8.66
9	3	0.95	9.61
10	0	0	9.61

Normalized DCG (nDCG)

- DCG numbers are averaged across a set of queries at specific rank values (DCG@*k*)
  - e.g., DCG at rank 5 is 6.89 and at rank 10 is 9.61
  - Can be any positive real number!
- DCG values are often normalized by comparing the DCG at each rank with the DCG value for the perfect ranking
  - makes averaging easier for queries with different numbers of relevant documents
- nDCG@*k* = DCG@*k* / iDCG@*k* (divide actual by ideal)
- nDCG ≤ 1 at any rank position
- To compare DCGs, normalize values so that a ideal ranking would have a normalized DCG of 1.0

nDCG

÷

k	G	DG	DCG@k	iG	iDG	iDCG@k	nDCG@k
1	3	3	3	3	3.00	3	1.00
2	2	2	5	3	3.00	6	0.83
3	3	1.89	6.89	3	1.89	7.89	0.87
4	0	0	6.89	2	1.00	8.89	0.78
5	0	0	6.89	2	0.86	9.75	0.71
6	1	0.39	7.28	2	0.77	10.52	0.69
7	2	0.71	7.99	1	0.36	10.88	0.73
8	2	0.67	8.66	0	0.00	10.88	0.80
9	3	0.95	9.61	0	0.00	10.88	0.88
10	0	0	9.61	0	0.00	10.88	0.88

Summary:

- IR test collection:
  - Document collection
  - Query set
  - Relevant judgements
  - IR measures
- IR measures:
  - R, P, F → not commonly used
  - P@k, R-precision → used sometimes
  - MAP → the most used IR measure
  - nDGC → the most used measure for web search

Resources

- Text book 1: Intro to IR, Chapter 8
- Text book 2: IR in Practice, Chapter 8

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

IR Evaluation (2)

Instructor:
Walid Magdy

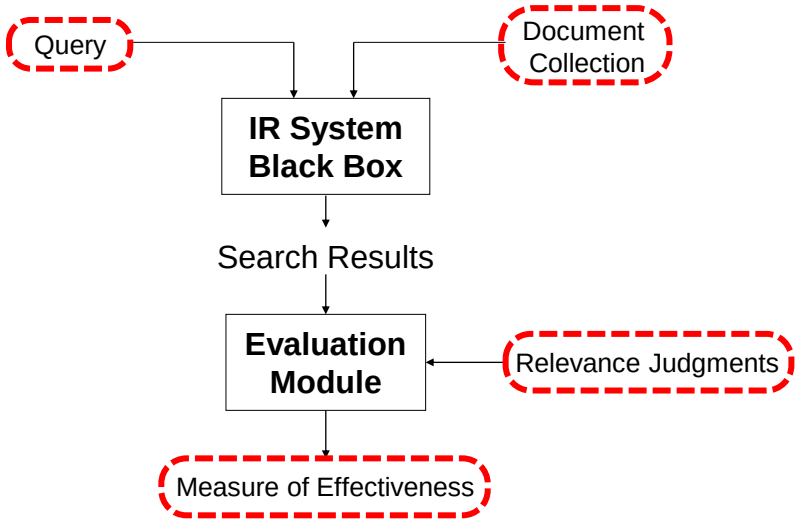
20-Oct-2021

1

Lecture Objectives

- Learn about how to evaluate IR
  - How to create a test collection?
  - Topic vs. query
  - Relevance judgements
  - Pooling

Cranfield Paradigm



Reusable IR Test Collection

- **Collection of Documents**
  - Should be "representative" to a given IR task
  - Things to consider: size, sources, genre, topics, …
- **Sample of information need**
  - Should be "randomized" and "representative"
  - Usually formalized **topic** statements (query + description)
- **Known relevance judgments**
  - Assessed by humans, for each topic-document pair
  - Binary/Graded
- **Evaluation measure**

Where Do Test Collections Come From?

- For web search, companies apply their own studies to assess the performance of their search engine.
- Web-search performance is monitored by:
  - Traffic
  - User clicks and session logs
  - Labelling results for selected users' queries
- For other search tasks:
  - Someone goes out and builds them (expensive)
  - As the by-product of large scale evaluations
- IR Evaluation Campaigns are created for this reason

IR Evaluation Campaigns

- IR test collections are provided for scientific communities to develop best IR methods
- Collections and queries are provided, relevance judgements are built during the campaign
- TREC = Text REtrieval Conferences
  - Main IR eval campaign. Sponsored by NIST (US gov)
  - Series of annual evaluations, started in 1992
  - Organized into "tracks"
- Other evaluation campaigns
  - CLEF: European version (since 2000)
  - NTCIR: Asian version (since 1999)
  - FIRE: Indian version (since 2008)

TREC Task

- It is a task for search a set of documents of given genre and domain.
- TREC (or other IR eval campaigns) are formed of a set of tracks, each track has a set of search tasks.
- Example
  - TREC Medical track
  - TREC Legal track → CLEF-IP track → NTCIR patent mining track
  - TREC Microblog track
  - Different CLIR tracks in all campaigns

TREC Collection

- 100's of collections were released in the different evaluation campaigns covering most of the domains in life
- A set of hundreds of thousands of docs
  - 1B in case of web search (TREC ClueWeb09)
- The typical format:

```
<DOC>
<DOCNO> 1234 </DOCNO>
<TEXT>
Multilines of plain text of the document
</TEXT>
</DOC>
```

TREC Topic

- Query sets are provided for each collection. Generated by experts and is associated with additional details. It is called **Topics**, and contains:
  - Query: the query text
  - Description: description of what is meant by the query
  - Narrative: what should be considered relevant

<num>189</num>
<title>Health and Computer Terminals</title>
<desc>Is it hazardous to the health of individuals to work with computer terminals on a daily basis?</desc>
<narr>Relevant documents would contain any information that expands on any physical disorder/problems that may be associated with the daily working with computer terminals. Such things as carpel tunnel, cataracts, and fatigue have been said to be associated, but how widespread are these or other problems and what is being done to alleviate any health problems</narr>

Relevance Judgements

- For each topic, set of relevant docs is required to be known for an effective evaluation!
- **Exhaustive assessment** is usually impractical
  - TREC usually has 50 topics
  - Collection usually has >1 million documents
- **Random sampling** won't work
  - If relevant docs are rare, none may be found!
- **IR systems** can help focus the sample (**Pooling**)
  - Each system finds some relevant documents
  - Different systems find different relevant documents
  - Together, enough systems will find most of them
  - Leverages cooperative evaluations

Pooling

1. Systems submit top **1000** documents per topic
2. Top **100** documents from each are judged
   - Single pool, duplicates removed, random ranking
   - Judged by the person who developed the topic
3. Treat unevaluated documents as irrelevant
4. Compute MAP (or others) down to **1000** documents

- To make pooling work:
  - Large number of reasonable systems participating
  - Systems must not all "do the same thing"

Pooling, does it work?

- Judgments can't possibly be exhaustive!
  It doesn't matter: relative rankings of different systems remain the same!
  Chris Buckley and Ellen M. Voorhees. (2004) Retrieval Evaluation with Incomplete Information. SIGIR 2004.
- This is only one person's opinion about relevance
  It doesn't matter: relative rankings remain the same!
  Ellen Voorhees. (1998) Variations in Relevance Judgments and the Measurement of Retrieval Effectiveness. SIGIR 1998.
- What about hits 101 to 1000?
  It doesn't matter: relative rankings remain the same!
- We can't possibly use judgments to evaluate a system that didn't participate in the evaluation!
  Actually, we can!
  Justin Zobel. (1998) How Reliable Are the Results of Large-Scale Information Retrieval Experiments? SIGIR 1998.

Who decides a doc is relevant or not?

- The same doc can be seen relevant by me, but not you
- Sometimes, it would be useful to have multiple judgements on relevance on the same document
- How to measure agreement among different assessors?
- Cohen's *kappa*

$$\varkappa = \frac{P(A) - P(E)}{1 - P(E)}$$

$P(A)$ – proportion of time judges agree (*inter-annotator agreement*)
$P(E)$ – what agreement would be by chance

Cohen's *kappa*

- Two judges (J_1 & J_2) annotating 50 docs for relevance
- $P(A) = \frac{20+15}{50} = 0.7$
- $P(E) = P(J_1,J_2|rel) + P(J_1,J_2|irrel)$
  - $P(rel) = P(J_1|rel) \cdot P(J_2|rel) = \frac{20+10}{50} \cdot \frac{20+5}{50} = 0.6 \times 0.5 = 0.3$
  - $P(irrel) = P(J_1|irrel) \cdot P(J_2|irrel) = \frac{20}{50} \cdot \frac{25}{50} = 0.4 \times 0.5 = 0.2$
- $\varkappa = \frac{P(A)-P(E)}{1-P(E)}$
- $= \frac{0.7-0.5}{1-0.5} = \frac{0.2}{0.5} = 0.4$

		J_1	
		Relevant	Irrelevant
J_2	Relevant	20	5
	Irrelevant	10	15

Cohen's *kappa* - meaning

- Kappa = 0, for chance agreement,
  = 1, for total agreement.
  < 0, for worse than random!
- Kappa > 0.8 ➔
  good agreement
- 0.67 < Kappa < 0.8 ➔
  "fair" agreement
- Kappa < 0.67 ➔
  seen as data providing a suspicious basis for an evaluation

Web Search Engines Evaluation

- Search engines have test collections of queries and hand-ranked results
- Recall is difficult to measure on the web – why?
- Search engines often use
  - precision at top k, e.g., k = 10
  - measures that reward you more for getting rank 1 right than for getting rank 10 right (nDCG)
  - non-relevance-based measures:
    - Clickthrough on first result
      not very reliable if you look at a single clickthrough … but pretty reliable in the aggregate.
    - Studies of user behaviour in the lab
    - A/B testing

Web Search Engines: A/B testing

- Purpose: Test a single innovation
- Prerequisite: You have a large search engine up & running.
- Have most users use old system
- Divert a small proportion of traffic (e.g., 1%) to the new system that includes the innovation
- Evaluate with an "automatic" measure like clickthrough on first result
- Now we can directly see if the innovation does improve user satisfaction.
- Probably the evaluation methodology that large search engines trust most

Is system B really better than A?

- Given the results from a number of queries, B achieved better score than A. How can we conclude that ranking algorithm B is really better than algorithm A?

	Experiment 1			Experiment 2	
Query	System A	System B	Query	System A	System B
1	0.20	0.40	1	0.02	0.76
2	0.21	0.41	2	0.39	0.07
3	0.22	0.42	3	0.16	0.37
4	0.19	0.39	4	0.58	0.21
5	0.17	0.37	5	0.04	0.02
6	0.20	0.40	6	0.09	0.91
7	0.21	0.41	7	0.12	0.46
Average	0.20	0.40	Average	0.20	0.40

Significance Test

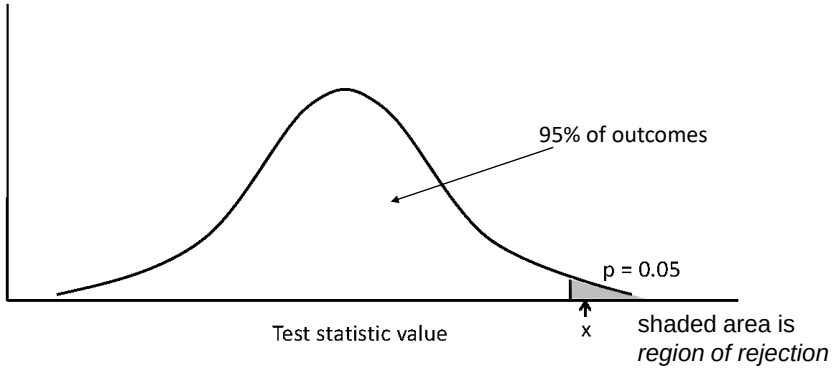
- **Null Hypothesis**:
  No relationship between two observed phenomena
  - Rejecting null hypothesis: observation has a meaning
- A **significance test** enables the rejection of *null hypothesis* (no difference) in favor of the *alternative hypothesis* (B is better than A).
- The power of a test is the probability that the test will reject the *null hypothesis* correctly.
  - increasing the number of queries in the experiment increases the power of test.

Significance Test: Steps

- Compute the effectiveness measure for every query for both retrieval systems (note: AP not MAP).
- Compute a **test statistic** based on a comparison of the effectiveness measures for each query.
  - depends on the significance test
- Test statistic is used to compute a ***p-value***: reflects the probability that the *null hypothesis* is true.
  - Small p-values suggest that the null hypothesis may be false.
- The null hypothesis (no difference) is rejected in favor of the alternate hypothesis (B is more effective than A) if **p-value ≤ α**, where **α** is the significance level.
  - Values for **α** are small, typically **0.05** or less, to reduce the chance of incorrect rejection.

One-sided Test Static

- Distribution for the possible values of a test statistic assuming the null hypothesis



t-test

- Assumption is that the difference between the effectiveness values is a sample from a normal distribution
- Null hypothesis is that the mean of the distribution of differences is zero
- Test statistic

$$t = \frac{\overline{B-A}}{\sigma_{B-A}} \cdot \sqrt{N}$$

- t-value to *p*-value
  http://www.socscistatistics.com/pvalues/tdistribution.aspx

Query	A	B	B-A
1	25	35	10
2	43	84	41
3	39	15	-24
4	75	75	0
5	43	68	25
6	15	85	70
7	20	80	60
8	52	50	-2
9	49	58	9
10	50	75	25

$\overline{B-A} = 21.4$, $\sigma_{B-A} = 29.1$, $t = 2.33$, p-value=.02

Significance Test

- It is not enough to show that system B achieves better score than system A
  - Significance test is essential
- Two-tailed t-test is highly accepted, with α=0.05
  - Sometimes it is required to use others
    Wilcoxon test: does not assume normal distribution
- Meaning of significance test for IR system
  - When a user uses system B that is significantly better than system A, he/she will feel the difference in performance
  - If system B is better than A but not significantly, the user won't notice a difference between the two systems

Now, is system B better than A?

Experiment 1

Query	System A	System B
1	0.20	0.40
2	0.21	0.41
3	0.22	0.42
4	0.19	0.39
5	0.17	0.37
6	0.20	0.40
7	0.21	0.41
Average	0.20	0.40

t-test p-value = 0

B is statistically significantly better than A

Experiment 2

Query	System A	System B
1	0.02	0.76
2	0.39	0.07
3	0.16	0.37
4	0.58	0.21
5	0.04	0.02
6	0.09	0.91
7	0.12	0.46
Average	0.20	0.40

t-test p-value = 0.306

B and A are statistically indistinguishable

Summary

- IR test-collection for automatic evaluation
  - Collection of documents
  - Set of topics
    - Topic = query + details on what is meant and what is relevant
    - Recommended minimum number of **25** topics
  - Relevance judgements
    - Pooling is the most common approach for creating judgements
    - Large number of diverse systems are required
  - Evaluation measure
    - Select the proper measure according to the IR task
    - Significance test is essential to confirm that improvement has real meaning
- Web-search uses different evaluation methods that relies on user experience and click-through data

Resources

- Text book 1: Intro to IR, Chapter 8
- Text book 2: IR in Practice, Chapter 8
- Pooling:
  Chris Buckley and Ellen M. Voorhees. (2004) Retrieval Evaluation with Incomplete Information. SIGIR 2004

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Query Expansion

Instructor:
Walid Magdy

27-Oct-2021

1

Lecture Objectives

- Learn about Query Expansion
  - Query expansion methods
  - Relevance feedback in IR
  - Rocchio's algorithm
  - PRF
- Implement:
  - PRF

Query Expansion

- Query: representation of user's information need
  - Many times it can be suboptimal
- Different words can have the same meaning
  - replacement, replace, replacing, replaced → Stemming
  - go, gone, went → Lemmatisation (NLP)
  - car, vehicle, automobile → ??
  - US, USA, the states, united states of America → ??
- Stemming/Lemmatisation → could be applied to normalise document and queries
  - Research shows that no significant difference between both
- Query Expansion (QE) → add more words of the same meaning to your query for better retrieval

Query Expansion: Methods

- Thesaurus
  - Group words into sets of synonyms (synsets)
  - Typically grouping is on the word level (neglects context)
  - Manually built: e.g. WordNet
    - NLTK wordnet: http://www.nltk.org/howto/wordnet.html
  - Automatically built:
    - Words co-occurence
    - Parallel corpus of translations
- Retrieved documents-based expansion
  - Relevance feedback
  - Pseudo (Blind) relevance feedback
- Query logs

Automatic Thesaurus: co-occurence

- Words co-occurring in a document/paragraph are likely to be (*in some sense*) similar or related in meaning
- Built using collection matrix (term-document matrix)
- For a collection matrix $\boldsymbol{A}$, where $\boldsymbol{A_{t,d}}$ is the normalised weight of term $\boldsymbol{t}$ in document $\boldsymbol{d}$, similarity matrix could be calculated as follows:

$$\boldsymbol{C = A.A^T}$$

where, $\boldsymbol{C_{u,v}}$ is the similarity score between terms $\boldsymbol{u}$ and $\boldsymbol{v}$. The higher the score, the more similar the terms

- Advantage: unsupervised
  Disadvantage: related words more than real synonyms

Automatic Thesaurus: co-occurence

- Example

Word	Nearest neighbors
absolutely	absurd, whatsoever, totally, exactly, nothing
bottomed	dip, copper, drops, topped, slide, trimmed
captivating	shimmer, stunningly, superbly, plucky, witty
doghouse	dog, porch, crawling, beside, downstairs
makeup	repellent, lotion, glossy, sunscreen, skin, gel
mediating	reconciliation, negotiate, case, conciliation
keeping	hoping, bring, wiping, could, some, would
lithographs	drawings, Picasso, Dali, sculptures, Gauguin
pathogens	toxins, bacteria, organisms, bacterial, parasite
senses	grasp, psyche, truly, clumsy, naive, innate

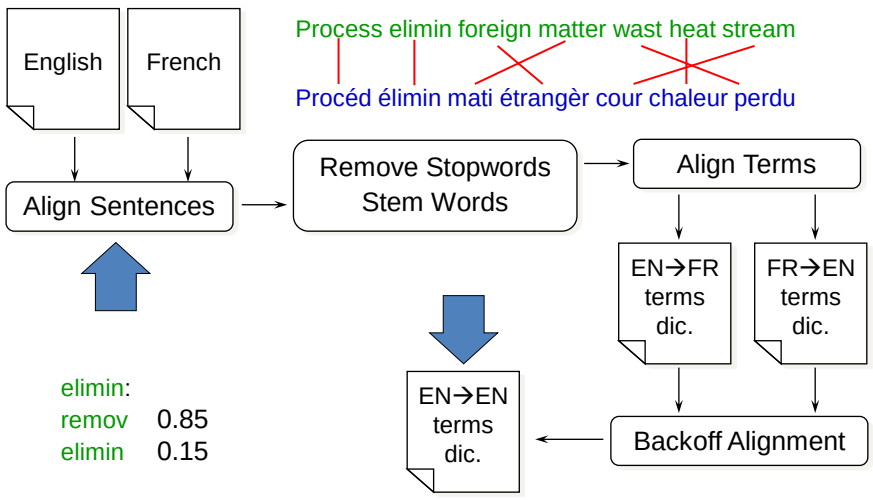
► **Figure 9.4** An example of an automatically generated thesaurus. This example is based on the work in Schütze (1998), which employs latent semantic indexing (see Chapter 18).

https://nlp.stanford.edu/IR-book/html/htmledition/automatic-thesaurus-generation-1.html#fig:autothesaurus

Automatic Thesaurus: parallel corpus

- Parallel corpus are the main training resource for machine translation systems
- Nature: sets of two parallel sentences in two different languages (source and target language)
- Idea:
  - More than one word in language X can be translated into the same word in language Y
    → these words in language X could be considered synsets
- Requirement: the presence of parallel corpus (training data) → supervised method

Automatic Thesaurus: parallel corpus



Automatic Thesaurus: parallel corpus

- Example

motor		weight		travel		color		link	
motor	0.63	weight	0.86	travel	0.67	color	0.56	link	0.4
engin	0.36	wt	0.14	move	0.19	colour	0.25	connect	0.18
				displac	0.14	dye	0.19	bond	0.17
								crosslink	0.13
								bind	0.12

cloth		tube		area		game		play	
fabric	0.36	tube	0.88	area	0.4	set	0.6	set	0.3
cloth	0.3	pipe	0.12	zone	0.23	game	0.4	play	0.24
garment	0.2			region	0.2			read	0.17
tissu	0.14			surfac	0.17			game	0.16
								reproduc	0.1

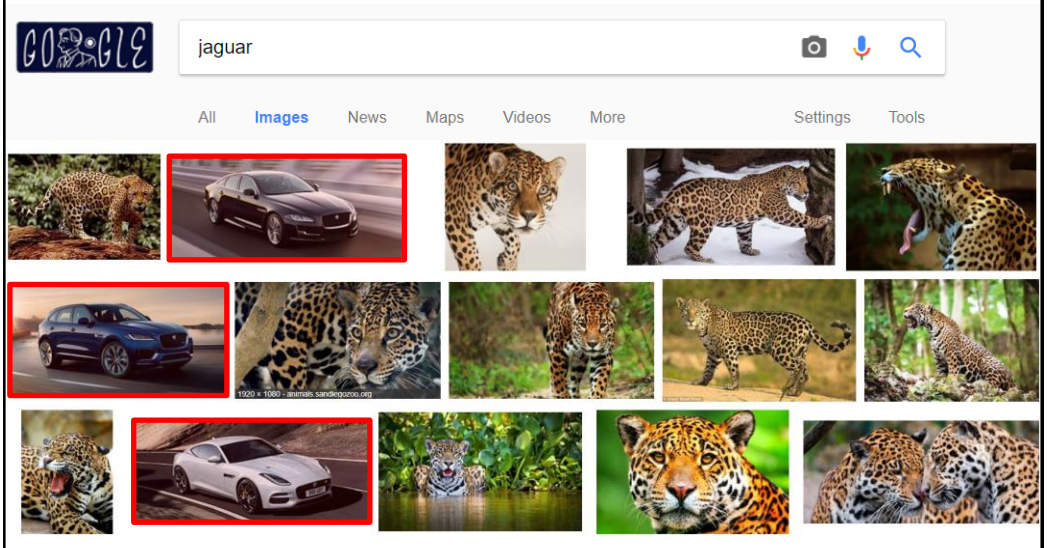
Thesaurus-based QE

- Works for very specific applications (e.g. medical domain)
- Many times fails to improve retrieval
  - Sometimes reduces both precision and recall
  - How?
- When it works, it is hard to get a consistent performance over all queries:
  - Improves some, and reduces others. Significant?
- Why it fails?
  - Lack of context
- Current research: word embeddings / BERT
  - No consistent improvement still

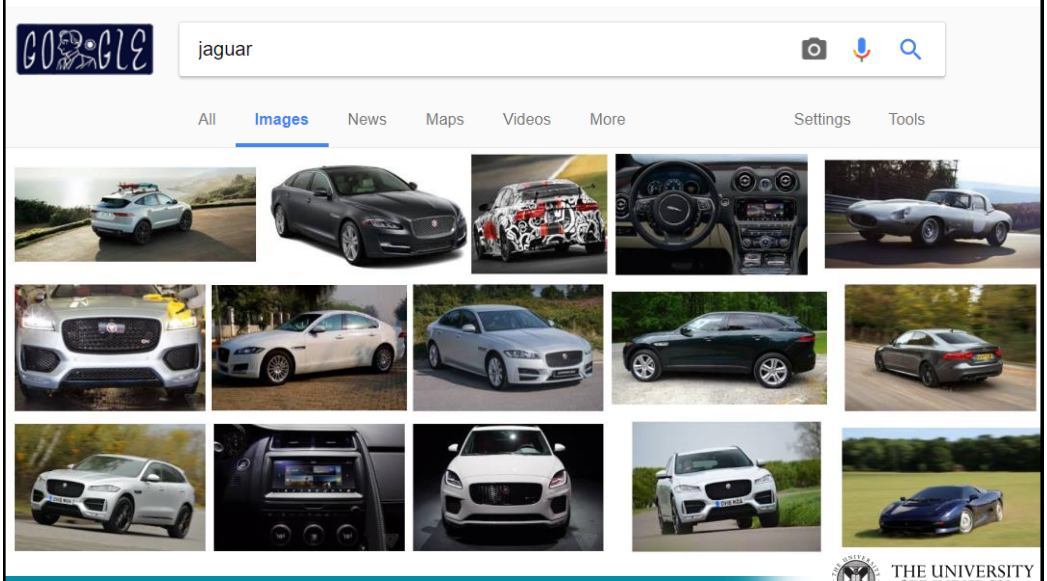
Relevance Feedback

- Idea: let user give feedback to the IR system about samples of what is relevant and what is not.
- User feedback on relevance of docs in initial results
  - User issues a (short, simple) query
  - The user marks some results as relevant or non-relevant.
  - The system computes a better representation of the information need based on feedback.
  - Relevance feedback can go through one or more iterations
- From user perspective: it may be difficult to formulate a good query when you don't know the collection well, BUT easier to judge particular documents

Example 1: Image Search



Example 1: Image Search



13

Example 2: Text Search

- Initial query: ***New space satellite applications***
- ***Initial Results***
  1. NASA Hasn't Scrapped Imaging Spectrometer
  2. NASA Scratches Environment Gear From Satellite Plan
  3. Science Panel Backs NASA Satellite Plan, But Urges Launches of Smaller Probes
  4. A NASA Satellite Project Accomplishes Incredible Feat: Staying Within Budget
  5. Scientist Who Exposed Global Warming Proposes Satellites for Climate Research
  6. Report Provides Support for the Critics Of Using Big Satellites to Study Climate
  7. Arianespace Receives Satellite Launch Pact From Telesat Canada
  8. Telecommunications Tale of Two Companies
- User then marks relevant documents with "+"
- System learns new terms

14

New terms common in selected docs

2.074	new	15.10	space
30.81	satellite	5.660	application
5.991	**nasa**	5.196	**eos**
4.196	**launch**	3.972	**aster**
3.516	instrument	3.446	rianespace
3.004	bundespost	2.806	ss
2.790	rocket	2.053	scientist
2.003	broadcast	1.172	earth
0.836	oil	0.646	measure

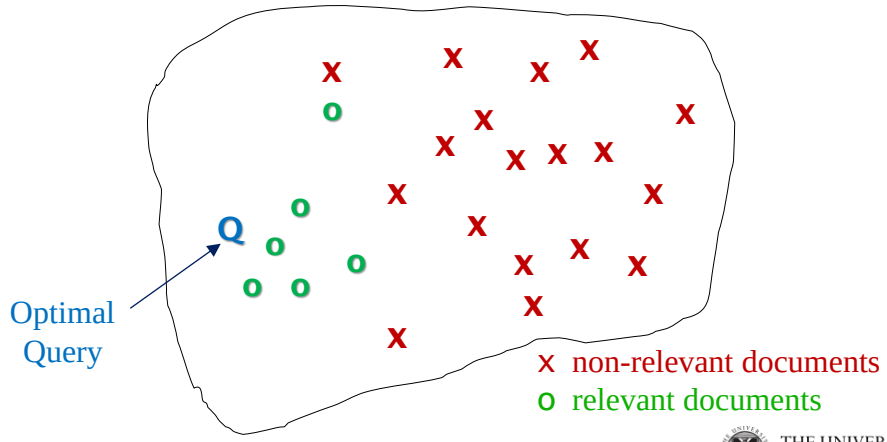
Adding new terms to the query

1. NASA Scratches Environment Gear From Satellite Plan
2. NASA Hasn't Scrapped Imaging Spectrometer
3. When the Pentagon Launches a Secret Satellite, Space Sleuths Do Some Spy Work of Their Own
4. NASA Uses 'Warm' Superconductors For Fast Circuit
5. Telecommunications Tale of Two Companies
6. Soviets May Adapt Parts of SS-20 Missile For Commercial Use
7. Gaping Gap: Pentagon Lags in Race To Match the Soviets In Rocket Launchers
8. Rescue of Satellite By Space Agency To Cost $90 Million

Hopefully better results!

Theoretical Optimal Query

- Found closer to *rel* docs and away from *irrel* ones.
- Challenge: we don't know the truly relevant docs



Rocchio's Algorithm

- Key Concept: Vector Centroid
- Recall that, in VSM, we represent documents as points in a high-dimensional space
- The centroid is the centre mass of a set of points

$$\vec{\mu}(C) = \frac{1}{|C|}\sum_{\vec{d}\epsilon C}\vec{d}$$

where C is a set of documents.

- Introduced **1963**

Rocchio Algorithm: theory

- Rocchio seeks the query $\vec{q}_{opt}$ that maximizes

$$\vec{q}_{opt} = \underset{\vec{q}}{\text{argmax}}[sim(\vec{q}, Crel) - sim(\vec{q}, Cirre_l)]$$

- For Cosine similarity

$$\vec{q}_{opt} = \frac{1}{|Crel|} \sum_{\vec{d_j} \in C_{rel}} \vec{d_j} - \frac{1}{|C_{irrel}|} \sum_{\vec{d_j} \notin C_{rel}} \vec{d_j}$$

$$\vec{q}_{opt} = \vec{\mu}(C_{rel}) - \vec{\mu}(C_{irrel})$$

Rocchio Algorithm: in practice

- Only small set of docs are known to be *rel* or *irrel*

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_{rel}|} \sum_{\vec{d_j} \in D_{rel}} \vec{d_j} - \gamma \frac{1}{|D_{irrel}|} \sum_{\vec{d_j} \in D_{irrel}} \vec{d_j}$$

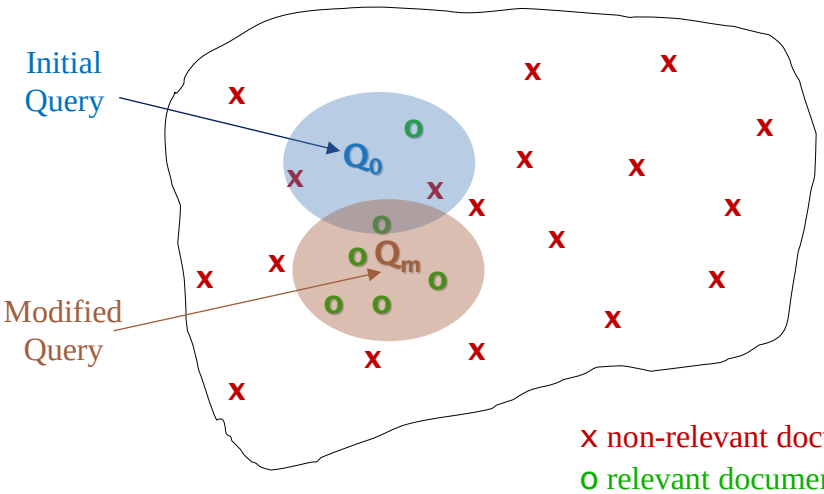
$\vec{q}_0$ = original query vector
D_{rel} = set of known relevant doc vectors
D_{irrel} = set of known non-relevant doc vectors
$\vec{q}_m$ = modified query vector
α = original query weights (hand-chosen or set empirically)
β = positive feedback weight
γ = negative feedback weight

- New query moves toward relevant documents and away from non-relevant documents

Notes about setting weights: α, β, γ

- Values of β, γ compared to α are set high when large judged documents are available.
- In practice, +ve feedback is more valuable than -ve feedback (usually, set $\beta > \gamma$)
  - Many systems only allow positive feedback (γ=0).
  - Or, use only highest-ranked negative document.
- When γ>0, some weights in query vector can go -ve.
  - "Jaguar" $\xrightarrow{feedback}$ jaguar + car + model - animal - jungle
- In practice, top n_t terms in $\vec{d_j} \in Drel$ are only selected
  - n = 5 → 50
  - Top n_t are identified using e.g. TFIDF

Effect of Relevance Feedback on Query



Effect of Relevance Feedback on Retrieval

- Relevance feedback can improve recall and precision
- In practice, relevance feedback is most useful for increasing recall in situations where recall is important.
- Empirically, one round of relevance feedback is often very useful. Two rounds is sometimes marginally useful.

Relevance Feedback: Issues

- Long queries are inefficient for typical IR engine.
  - High cost for retrieval system. (why?)
  - Long response times for user.
- It's often harder to understand why a particular document was retrieved after applying relevance feedback
- Users are often reluctant to provide explicit feedback → not practical!

Relevance Feedback: Practicality

- User revises and resubmits query
  - Users may prefer revision/resubmission to having to judge relevance of documents.
  - Useful for query suggestion to other users
- Is there a way to apply relevance feedback without user's input?

Pseudo (Blind) Relevance Feedback

- Solves the problem of users hate to provide feedback
- Feedback is applied blindly (PRF)
  - Automates the "manual" part of true relevance feedback.
- Algorithm:
  - Retrieve a ranked list of hits for the user's query
  - Assume that the top *k* documents are relevant
  - Do relevance feedback (e.g. Rocchio)
  - Typically applies only positive relevance feedback (γ=0)
- Mostly works
  - Still can go horribly wrong for some queries (when top *k* docs are not relevant)
  - Several iterations can lead to query drift

PRF (BRF)

- Was proven to be useful for many IR applications
  - News search (learn names and entities)
  - Social media search (learn hashtags)
  - Web search (implicit feedback is used more = clicks)
- Some domains are more challenging
  - Patent search
    - Top documents are usually not relevant
    - Patent text in general is unclear/confusing
- PRF is the most basic QE method for IR
  - Unsupervised
  - Language independent
  - Does not require any kind of language resources

PRF (BRF): Evaluation

- In practice, different number of feedback docs (n_d) and terms (n_t) are usually tested for PRF
  - n_d: 1 → 50
  - n_t: 5 → 50
- Results of PRF are directly compared to baseline (with no PRF)
  - It is not considered cheating.
  - It is essential to show that improvement is significant, and preferred to show the % of queries improved vs degraded.

Summary

- QE: automatically add more terms to user's query to better match relevant docs
- QE via thesaurus
  - Manual/automatic thesaurus: useful for specific applications
  - Fail when context is important
- Relevance feedback
  - Get samples of *rel*/*irrel* docs for extracting QE useful terms
  - Rocchio's is one of the most common algorithms for query modification
- PRF
  - Skips user's input for the feedback process
  - Found to be useful in many applications

Resources

- Text book 1: Intro to IR, Chapter 9
- Text book 2: IR in Practice, Chapter 6.2, 6.3
- Reading:
  Magdy W. and G. J. F. Jones.
  A Study on Query Expansion Methods for Patent Retrieval.
  PAIR 2011 - CIKM 2011 (link)
- Lab 5

THE UNIVERSITY of EDINBURGH

Search is not only the Web
IR Applications

Walid Magdy

School of Informatics
University o Edinburgh

27 Oct 2021

1

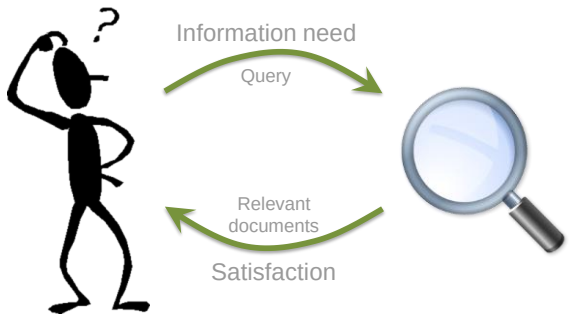
Objectives

- Main objective of IR
- Two search tasks
  - Printed documents search
  - Patent search
- Possible ideas for Group Project ☺

Information Retrieval Objective

- IR is finding material of an unstructured nature that satisfies an information need from within large collections.

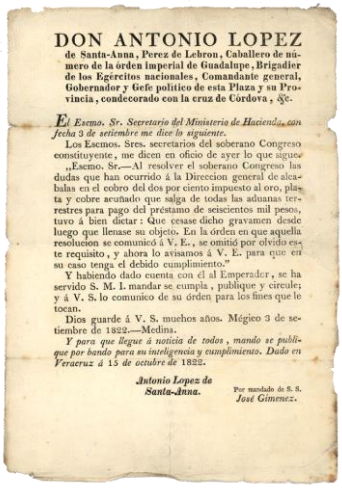
- **Information need**
  - Expected search scenario?
  - Modeling the task?
- **Data nature**
  - Approach?
  - Scalable? Fast?
- **User Satisfaction**
  - More relevant documents?
  - Effective evaluation?



Printed Documents Retrieval

Printed Documents Retrieval

- **Documents:**
  text on printed papers (books)
- **Information need:**
  Information within these books
- **Challenge:**
  It is an image of text
- **Common Approach:**
  OCR → Recognized text ← Search
- **Challenges in Common Approach:**
  OCR → Text with mistakes ($WER_{Ar} \approx 40\%$)
  OCR → Not available for all languages



DON ANTONIO LOPEZ
de Santa-Anna, Perez de Lebron, Caballero de número de la orden imperial de Guadalupe, Brigadier de los Egércitos nacionales, Comandante general, Gobernador y Gefe politico de esta Plaza y su Provincia, condecorado con la cruz de Córdova, &c.

El Escmo. Sr. Secretario del Ministerio de Hacienda, con fecha 3 de setiembre me dice lo siguiente.

Los Escmos. Sres. secretarios del soberano Congreso constituyente, me dicen en oficio de ayer lo que sigue: „Escmo. Sr.—Al resolver el soberano Congreso las dudas que han ocurrido á la Direccion general de alcabalas en el cobro del dos por ciento impuesto al oro, plata y cobre acuñado que salga de todas las aduanas terrestres para pago del préstamo de seiscientos mil pesos, tuvo á bien dictar: Que cesase dicho gravamen desde luego que llenase su objeto. En la órden en que aquella resolucion se comunicó á V. E., se omitió por olvido este requisito, y ahora lo avisamos á V. E. para que en su caso tenga el debido cumplimiento."

Y habiendo dado cuenta con él al Emperador, se ha servido S. M. I. mandar se cumpla, publique y circule; y á V. S. lo comunico de su órden para los fines que le tocan.

Dios guarde á V. S. muchos años. Mégico 3 de setiembre de 1822.—Medina.

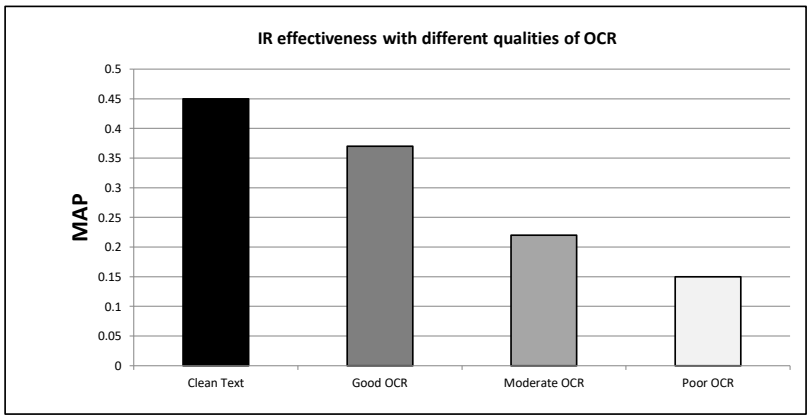
Y para que llegue á noticia de todos, mando se publique por bando para su inteligencia y cumplimiento. Dado en Veracruz á 15 de octubre de 1822.

Antonio Lopez de Santa-Anna.

Por mandado de S. S.
José Gimenez.

Problem

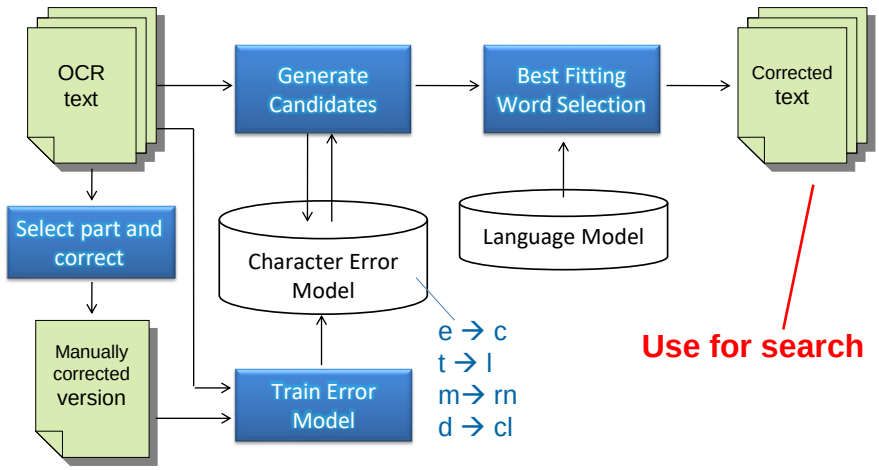
- Text with errors (sometime many errors)



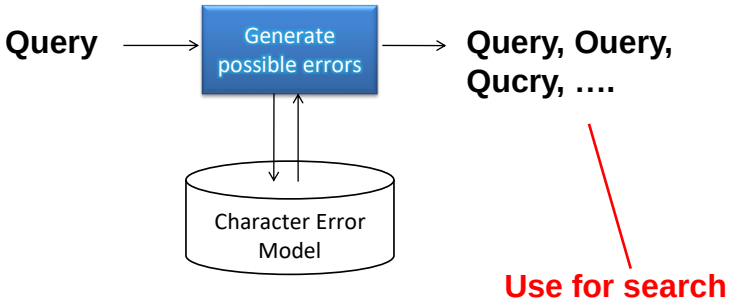
n-gram Char Representation of OCR

- **Original:**
  example sentence
- **OCR output:**
  exarnple senlcnce
- **3-gram char representation:**
  $ex exa xar arn rnp npl ple le$ $se sen enl nlc lcn cnc nce ce$
- **Query:**
  example sentence →
  $ex exa xam amp mpl ple le$ $se sen ent nte ten enc nce ce$
- **Matching:**
  $ex exa xar arn rnp npl ple le$ $se sen enl nlc lcn cnc nce ce$
  $ex exa xam amp mpl ple le$ $se sen ent nte ten enc nce ce$

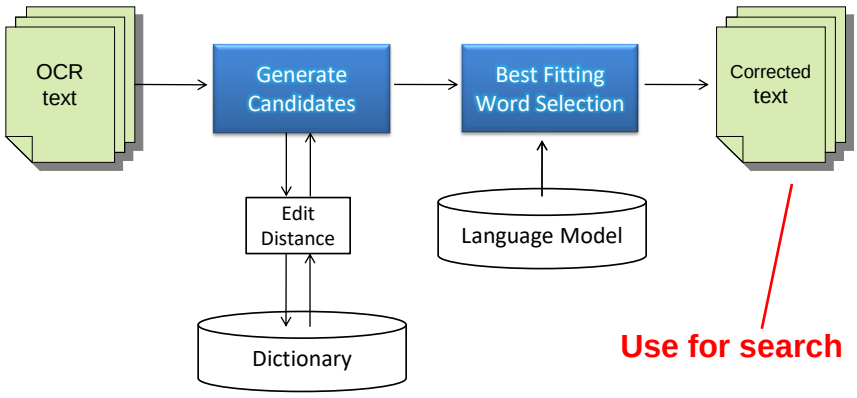
OCR Correction using Error Model



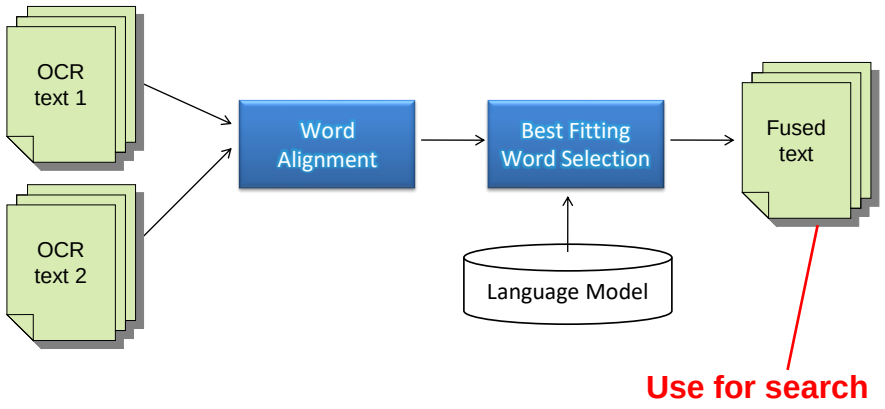
Query Garbling using Error Model



OCR Correction using Edit Distance



Multi-OCR Text Fusion

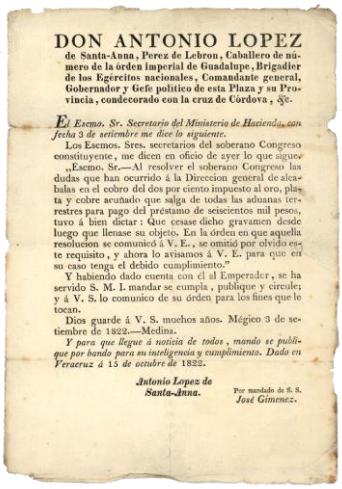


OCR Search

- Recognition errors in OCR text degrades retrieval
- Different methods of text processing can overcome the negative effect on retrieval and improves search
- n-gram character representation improves retrieval, but not that much
- Some training and resources are needed which can be manual correction, trained language model, or both
- Previous methods fail when errors are large (WER>50%)

Solution – back to Information Need

- **Information need:**
  the printed papers
- **Question:**
  Why convert image to text?
- **Related work:**
  Word Spotting



DON ANTONIO LOPEZ
de Santa-Anna, Perez de Lebron, Caballero de número de la orden imperial de Guadalupe, Brigadier de los Egércitos nacionales, Comandante general, Gobernador y Gefe politico de esta Plaza y su Provincia, condecorado con la cruz de Córdova, &c.

El Escmo. Sr. Secretario del Ministerio de Hacienda, con fecha 3 de setiembre me dice lo siguiente.

Los Escmos. Sres. secretarios del soberano Congreso constituyente, me dicen en oficio de ayer lo que sigue:

„Escmo. Sr.—Al resolver el soberano Congreso las dudas que han ocurrido á la Direccion general de alcabalas en el cobro del dos por ciento impuesto al oro, plata y cobre acuñado que salga de todas las aduanas terrestres para pago del préstamo de seiscientos mil pesos, tuvo á bien dictar: Que cesase dicho gravamen desde luego que llenase su objeto. En la órden en que aquella resolucion se comunicó á V. E., se omitió por olvido este requisito, y ahora lo avisamos á V. E. para que en su caso tenga el debido cumplimiento."

Y habiendo dado cuenta con él al Emperador, se ha servido S. M. I. mandar se cumpla, publique y circule; y á V. S. lo comunico de su órden para los fines que le tocan.

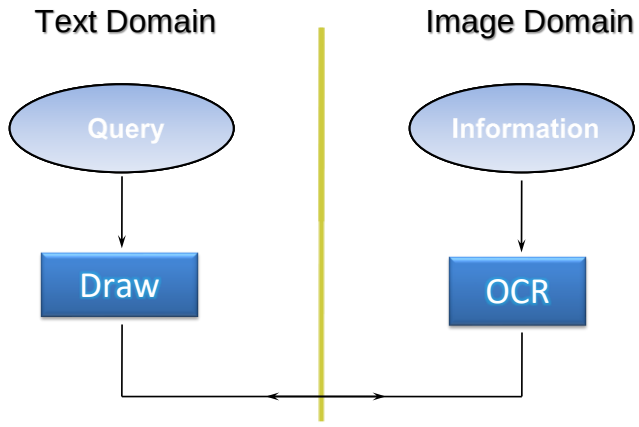
Dios guarde á V. S. muchos años. Mégico 3 de setiembre de 1822.—Medina.

Y para que llegue á noticia de todos, mando se publique por bando para su inteligencia y cumplimiento. Dado en Veracruz á 15 de octubre de 1822.

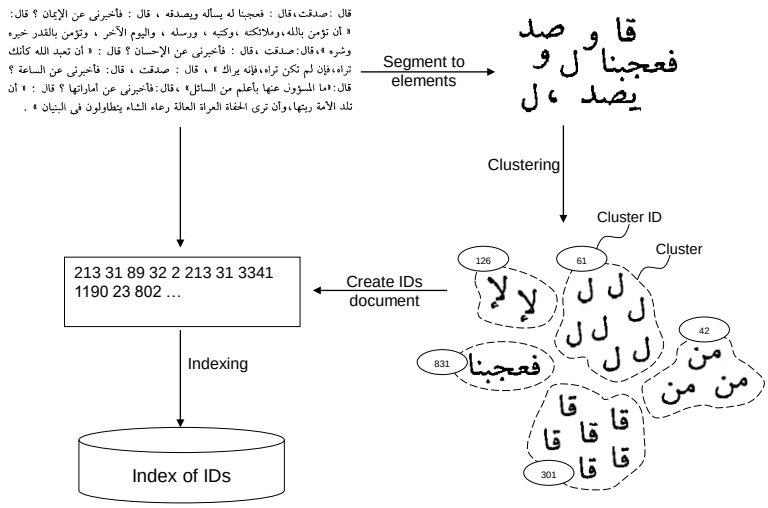
Antonio Lopez de Santa-Anna.

Por mandado de S. S.
José Gimenez.

Modeling the Problem

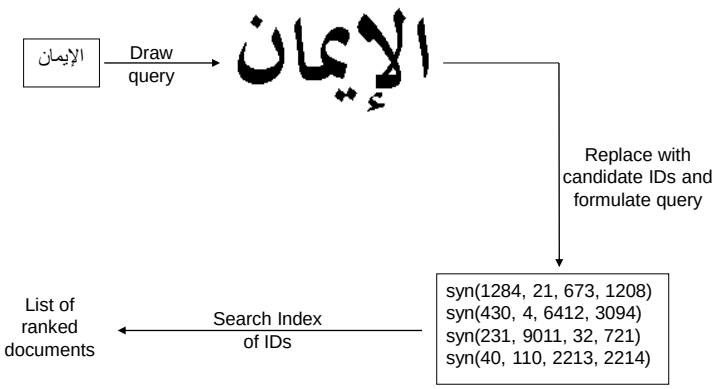


OCRless Search



15

Solution – OCRless Search



16

Solution – OCRless Search

- **Effective and fast**
- **Robust to OCR errors** (*v1de0*)
- **No training resources required**
- **Language independent**

- **Microsoft TechFest Demo**
  The same engine for searching printed documents in:
  Arabic, English, Chinese, Hebrew, and Hieroglyphic

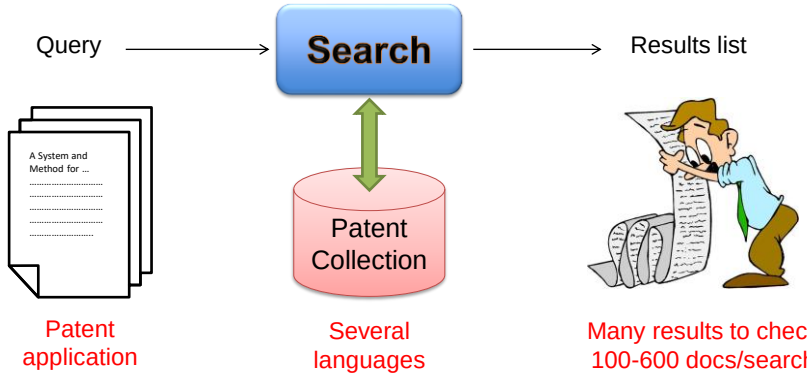
Printed Documents Retrieval

- **Text-based solutions: correction**
- **Image-based: clustering**
- **Current State-of-the-art?**
- **Information need → Approach**

Patent Search

Patent Search

- Given a patent application, check if the invention described is novel



Patent Search – User Satisfaction

- NTCIR, CLEF, TREC
- Recall-oriented → Try not to miss a relevant document
  - Recall is the objective
- Precision is also important
- Huge # documents checked (100-600 documents)
- Evaluation: average precision (AP)!!
  - Focuses on finding relevant docs early in ranked list
  - Less focus on recall

Example

For a topic with 4 relevant docs and 1st 100 docs to be examined:

System1: relevant ranks = {1}

System2: relevant ranks = {50, 51, 53, 54}

System3: relevant ranks = {1, 2, 3, 4}

$AP_{system1} = 0.25$	$R_{system1} = 0.25$
$AP_{system2} = 0.0481$	$R_{system2} = 1$
$AP_{system3} = 1$	$R_{system3} = 1$

- We need a metric that reflects recall and ranking quality in one measure

PRES: Patent Retrieval Evaluation Score

$$PRES = 1 - \frac{\frac{\sum r_i}{n} - \frac{n+1}{2}}{N_{\max}}$$

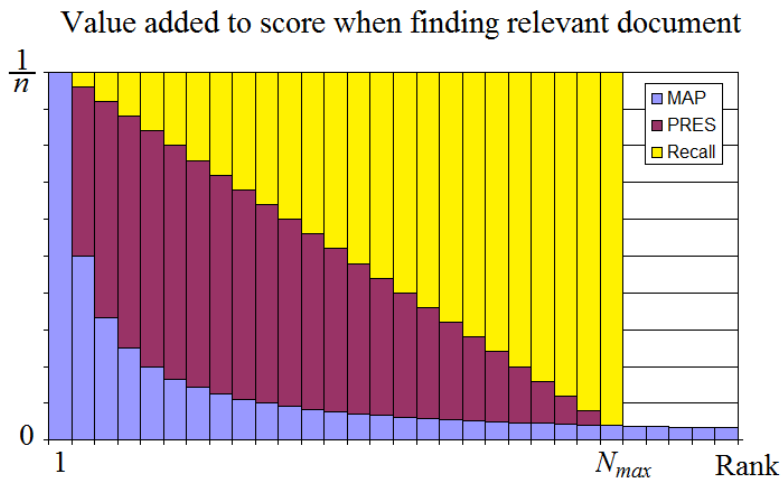
n: number of relevant docs

r_i: rank of the i^{th} relevant document

N_{max}: max number of checked docs

- **Derived from R_{norm} (Rocchio, 1964)**
- **Gives higher score for systems achieving higher recall and better average relative ranking**
- **Dependent on user's potential/effort (N_{max})**
- **Robust to incomplete relevance judgements**

PRES: as a cumulative gain



Patent Search – CLIR

- **Query: Full patent application**
- **Common approach: MT** (the best)
- **Challenge: training recourses + speed!**
- **Ideal: Query + Document translation**

Patent Search – CLIR – Objective?

- **Manual translation**

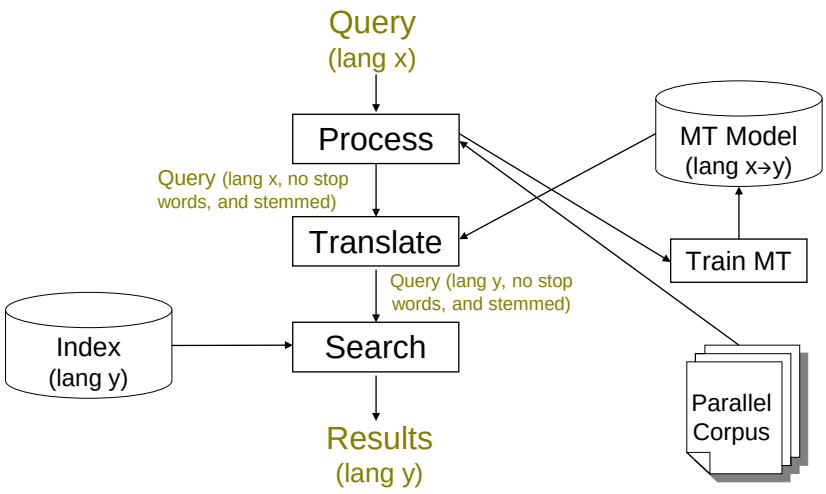
It is a great idea to apply stemming in information retrieval

- **MT output**

he are an great ideas to applied stem by information retrieving

- **MT evaluation: MT sucks**
- **IR evaluation: MT rocks ☺**
- **MT4IR:** An efficient MT that neglects morphological and syntactic features of output

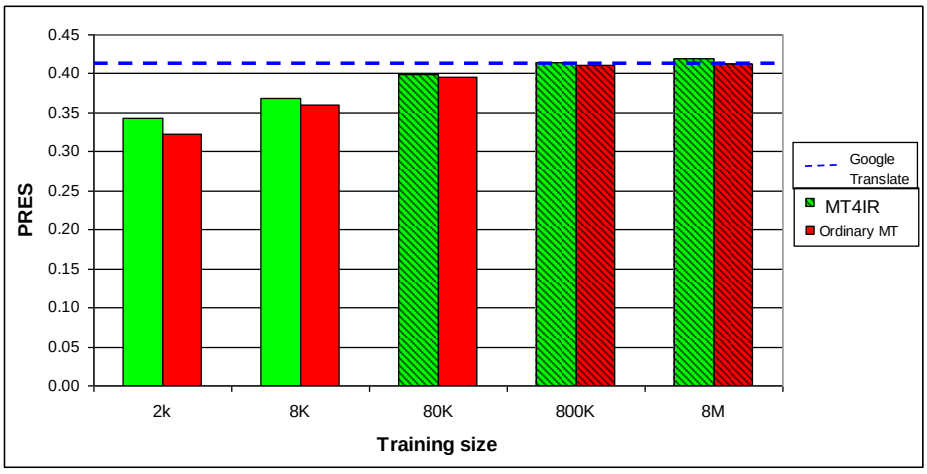
Ordinary MT vs. MT4IR



27

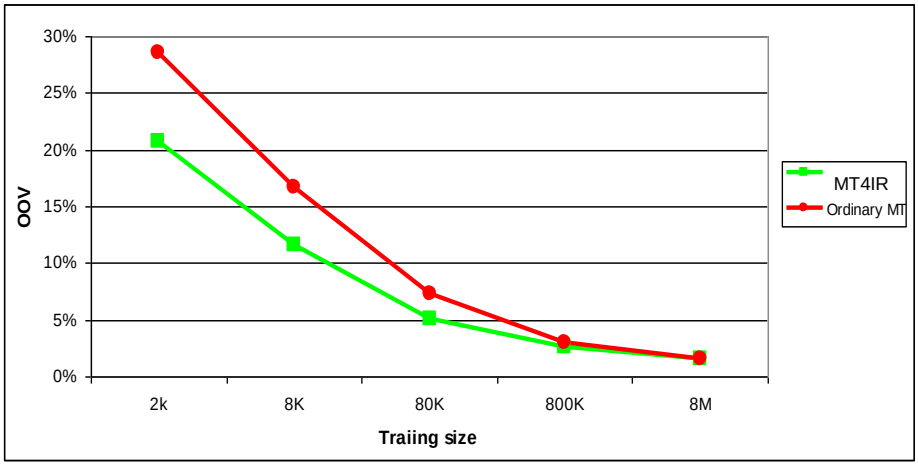
Patent Search – MT4IR

Retrieval effectiveness for a Patent CLIR **En-Fr** task



28

Patent Search – MT4IR

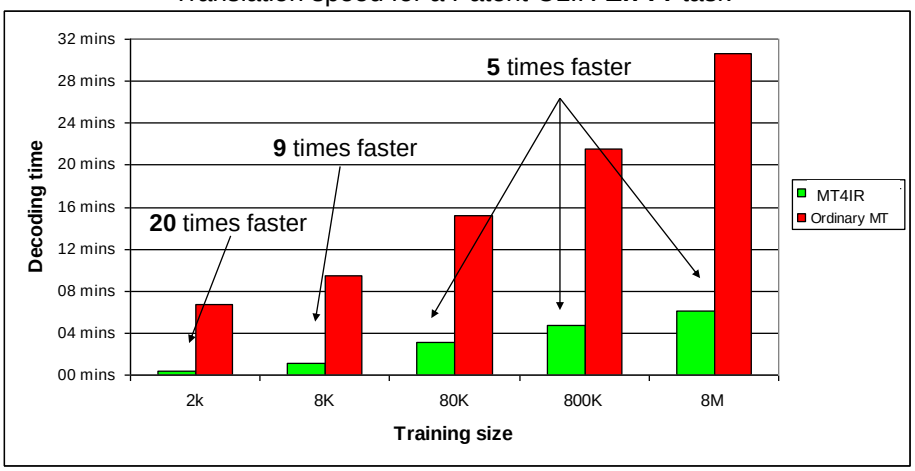


E.g. play, plays, played, playing

29

Patent Search – MT4IR

Translation speed for a Patent CLIR **En-Fr** task



30

Summary

- The objective is IR is "User Satisfaction"
- Understand the user needs well
- Design the IR task carefully
- You do not have to stick to the path in the literature
- Are you sure performance is measured correctly?

Readings

- Magdy W. and G. J. F. Jones. Studying Machine Translation Technologies for Large-Data CLIR Tasks: A Patent Prior-Art Search Case Study. *Springer, Information Retrieval, 2013*
- Magdy W. and G. J. F. Jones. PRES: A Score Metric for Evaluating Recall-Oriented Information Retrieval Applications. *SIGIR 2010*
- Magdy W. , K. Darwish, and M. El-Saban. Efficient Language-Independent Retrieval of Printed Documents without OCR. *SPIRE 2009*
- Magdy W. and K. Darwish. Effect of OCR Error Correction on Arabic Retrieval. *Springer, Information Retrieval, 2008*

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Web Search

Instructor:
Walid Magdy

3-Nov-2021

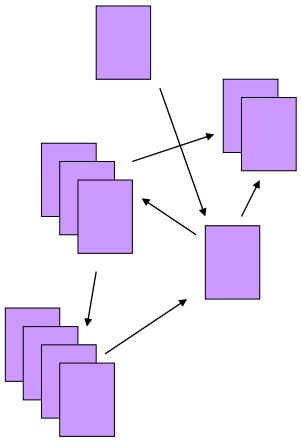
1

Lecture Objectives

- Learn about:
  - Working with Massive data
  - Link analysis (PageRank)
  - Anchor text

The Web Document Collection

- Huge / Massive
- Graph / Connected
- No design/co-ordination
- Distributed content publishing
- Content includes truth, lies, obsolete information, contradictions …
- Unstructured (text, html, …), semi-structured (XML, annotated photos), structured (DB) …
- Growth – slowed down from initial "volume doubling every few months" but still expanding
- Content can be dynamically generated



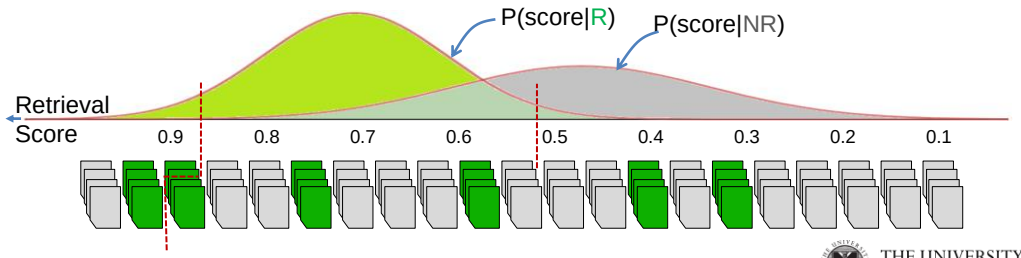
The Web

Effect of Massive data

- Web search engines work with huge amount of data
  - 20 PB/day in 2008 → 160 PB/day in 2013 → now??
  - 1 PB = 1,000 TB = 1,000,000 GB
- How this would affect a search engine?
  - Very challenging (storage, processing, networking, …)
  - Very useful still (makes stuff easier), how?
- Assume two good search engines the collects two sub-sets of the web
  - Search engine A collected N docs → precision@10 = 40%
  - Search engine B collected 4N docs → precision@10??

Effect of Massive data on Precision

- Assume two good search engines that collect two sub-sets of the web
  - Search engine A collected N docs → precision@10 = 40%
  - Search engine B collected 4N docs → precision@10??
    - Distribution of positive/negative scores stays the same
    - Precision/Recall at a given score stays the same
    - In any decent IR system: more relevant docs exist at the top → P@n ↑↑ ➔ precision@10 = 60% (increases)



Big Data or Clever Algorithm?

- For Web search, larger index usually would beat a better retrieval algorithm
  - Google Index vs Bing Index
- Similar to other applications
  - Google MT vs IBM MT
    - Statistical methods trained over 10x training data beat deep NLP methods with 1x training data
  - In general ML, the more data, the better the results
    - Tweets classification: using 100x of noisy training data beats 1x of well prepared training data, even with absence of stemming & stopping
  - Question answering task:
    - IBM Watson vs Microsoft experiment

Big Data or Clever Algorithm?

- Question answering task:
  - **Q**: Who created the character of Scrooge?
  - **A**: Scrooge, introduced by Charles Dickens in "A Chrismas Carol"
  - Requires heavy linguistic analysis, lots of research in TREC
- 2002, Microsoft
  - Identify (subj verb obj), rewrite as queries:
    - Q1: "created the character of Scrooge"
    - Q2: "the character of Scrooge was created by"
  - Search the web for exact phrase, get top 500 results
  - Extract phrase: ■Q1 or Q2■ , get most frequent ■
  - Very naive approach, ignores most answers patterns
  - Who cares!! Web is huge, you will find matches anyway

117	Dickens
78	Christmas Carol
75	Charles Dickens
72	Disney
54	Carl Banks
	…

Search "Microsoft"

Doc1

Microsoft.com

"Microsoft" mentioned 5 times

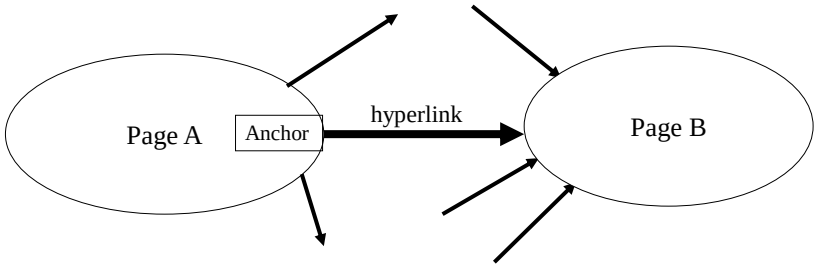
Doc2

Tutorial.com

Tutorial on MS word

"Microsoft" mentioned 35 times

The Web as a Directed Graph



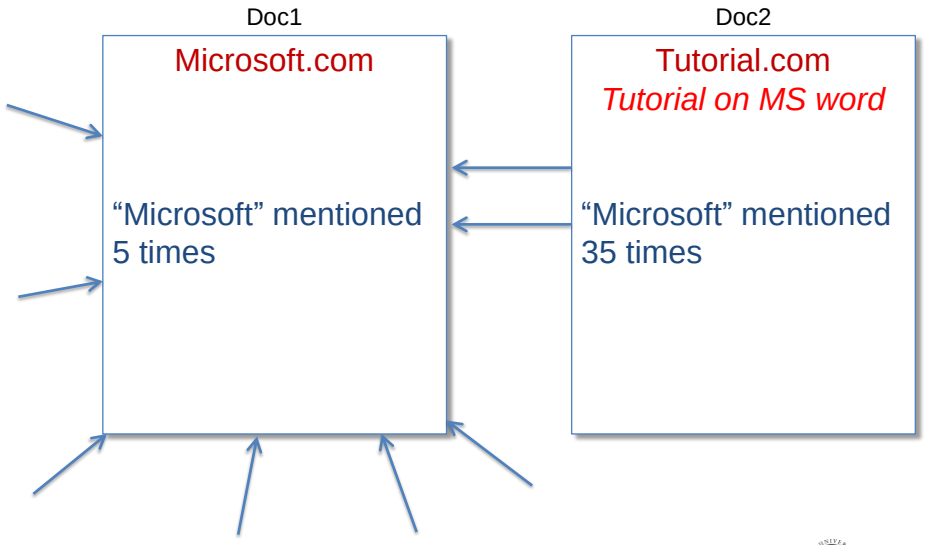
Assumption 1: A hyperlink between pages denotes author perceived relevance (quality signal)

Assumption 2: The text in the anchor of the hyperlink describes the target page (textual context)

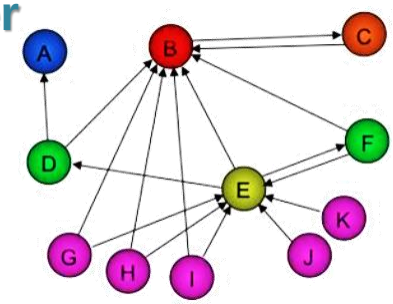
Links between Pages

- Google Description of **PageRank**:
  - Relies on the "uniquely democratic" nature of the web
  - Interprets a link from page A to page B as "a vote"
- A → B: means A thinks B worth something
  - "wisdom of the crowds": many links means B must be good
  - Content-independent measure of quality of B
- Use as ranking feature, combined with content
  - But not all pages that link to B are of equal importance!
    - Importance of a link from CNN >>> link from blog page
- Google PageRank, 1998
  - How many "good" pages link to B?

Search "Microsoft"

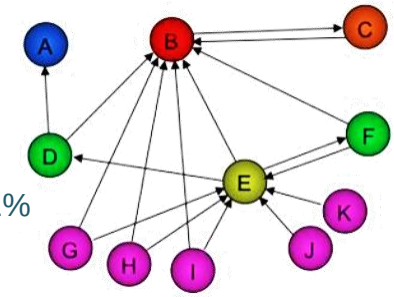


PageRank: Random Surfer



- Analogy:
  - User starts browsing at a random page
  - Pick a random outgoing link → goes there → repeat forever
  - Example: G → E → F → E → D → B → C
  - With probability 1-λ jump to a random page
    - Otherwise, can get stuck forever A, or B ↔ C
- **PageRank** of page x
  - Probability of being at page x at a random moment in time

PageRank: Algorithm



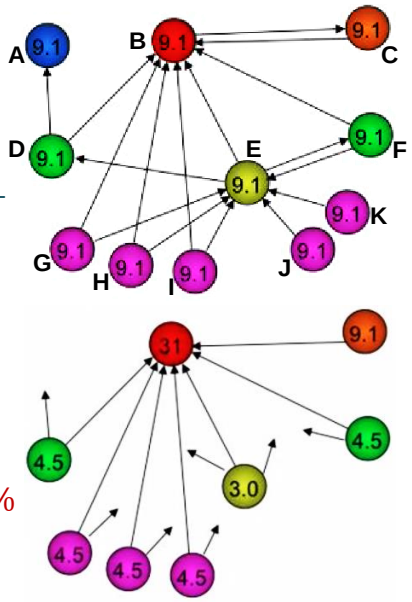
- Initialize $PR_0(x) = \frac{100\%}{N}$
  - *N*: total number of pages
  - $PR_0(A) = .. = PR_0(K) = \frac{100\%}{11} = 9.1\%$
- For every page x

$$PR_{t+1}(x) = \frac{1-\lambda}{N} + \lambda \sum_{y \to x} \frac{PR_t(y)}{L_{out}(y)}$$

- $y \longrightarrow x$ contributes part of its PR to x
- Spread PR equally among out-links
- Iterate till converge → PR scores should sum to 100%

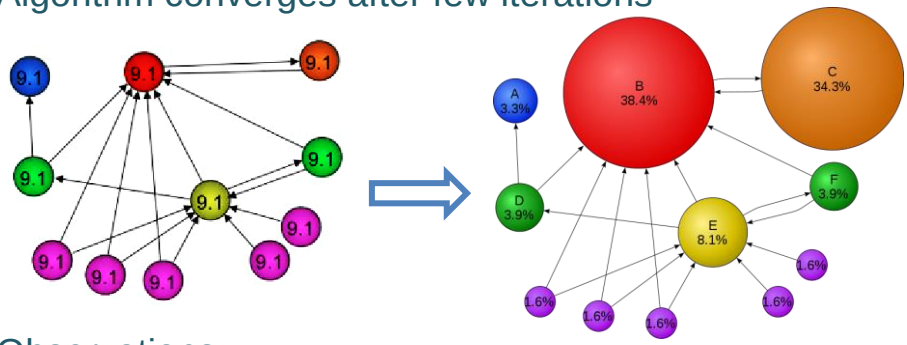
PageRank: Example

- Let $\lambda = 0.82$
- $PR(B) = \frac{0.18}{11} + 0.82 \times [PR(C) + \frac{1}{2}PR(D) + \frac{1}{3}PR(E) + \frac{1}{2}PR(F) + \frac{1}{2}PR(G) + \frac{1}{2}PR(H) + \frac{1}{2}PR(I)$
  $\approx 0.31 = 31\%$
- $PR(C) = \frac{0.18}{11} + 0.82 \times PR(B)$
  $= 0.18 \times 9.1\% + 0.82 \times 9.1\%$
  $= 9.1\%$
- $PR_{t+1}(C) = 0.18 \times 9.1\% + 0.82 \times 31\%$
  $\approx 26\%$



PageRank: Example result

- Algorithm converges after few iterations



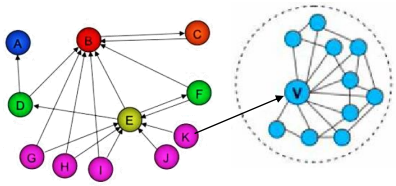
- Observations
  - Pages with no inlinks: PR = $(1-\lambda)/N$= 0.18/11 = 1.6%
  - Same (or symmetric) inlinks → same PR (e.g. **D** and **F**)
  - One inlink from high PR >> many from low PR (e.g. **C** vs **E**)

Anchor Text

- Anchor Text (text of a link):
  - Description of destination page
  - Short, descriptive like a query
  - Re-formulated in different ways
    - Human "query expansion"
- Used when indexing page content
  - Add text of all anchor text linking the page
  - Different weights for different anchor text
    - Weighted according to PR of linking page
- Significantly improves retrieval

Link Spam

- Trackback links (blogs that link to me)
  - Based on `$HTTP_REFERER`
  - Artificial feedback loops
    - Similar to "*follow back*" in Twitter
- Links from comments on sites with high PR
  - Links in comments on CNN
  - One solution: insert `rel=nofollow` into links
    - Link ignored when computing PR
- Link farms
  - Fake densely-connected graph
  - Hundreds of web domains / IPs can be hosted on one machine



The Reality

- **PageRank** is used in Google, but is hardly the full story of ranking
  - A big hit when initially proposed, but just one feature now
  - Many sophisticated features are used
  - Machine-learned ranking heavily used
    - Learning to Rank (L2R)
    - Many features are used, including PR
  - Still counted as a very useful feature

Summary

- Web data is massive
  - Challenging for efficiency, but useful for effectiveness
- PageRank:
  - Probability than random surfer is currently on page x
  - The more powerful pages linking to x, the higher the PR
- Anchor text:
  - Short concise description of target page content
  - Very useful for retrieval
- Link Spam
  - Trackable links, link farms

Resources

- Text book 1: Intro to IR, Section 21.1
- Text Book 2: IR in Practice: 4.5, 10.3
- Page Rank Paper:
  Page, L., Brin, S., Motwani, R., & Winograd, T. (1999).
  The PageRank citation ranking: Bringing order to the web.
  Stanford InfoLab.
- Additional reading:
  Dumais, S., Banko, M., Brill, E., Lin, J., & Ng, A. (2002)
  Web question answering: Is more always better?.
  SIGIR 2002.
- YouTube Video: How Search Works
  https://www.youtube.com/watch?v=BNHR6IQJGZs

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Web Search (2)

Instructor:
Walid Magdy

03-Nov-2021

1

Lecture Objectives

- Learn about:
  - Basics of Web search
  - Brief History of web search
  - SEOs
  - Web Crawling (intro)

Brief History

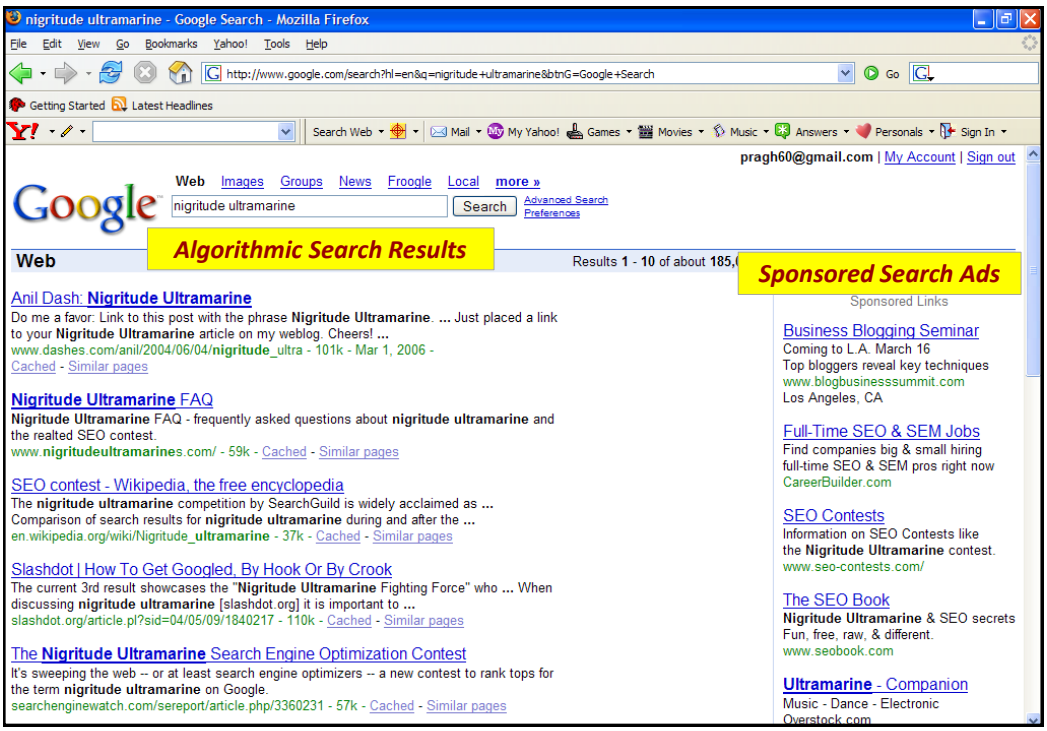
- Early keyword-based engines (1995-1997)
  - Altavista, Excite, Infoseek, Lycos, AOL
  - Traditional IR techniques
  - Scalability is an issue
- Paid search ranking: Goto (morphed into Overture.com → Yahoo!)
  - Your search ranking depended on how much you paid
  - Auction for keywords
  - Called "sponsored search"
    - CPC (Cost Per Click)
    - CPM (Cost Per Thousand Impressions)

CPC / CPM / RPM

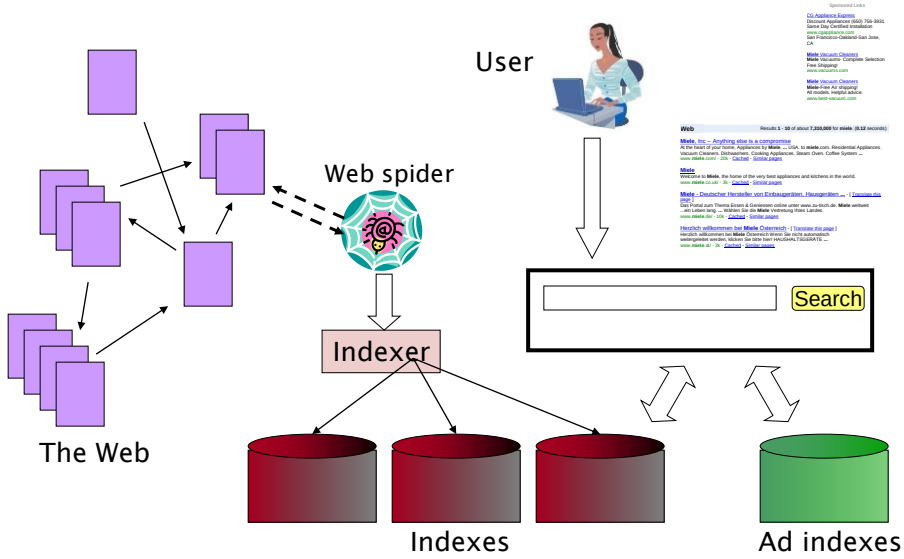
- With new services on the web → RPM
- RPM: Revenue per 1000 video views
- Read more:
  Understand ad revenue analytics
  https://support.google.com/youtube/answer/9314357

Brief (non-technical) History

- 1998+: Link-based ranking pioneered by Google
  - Blew away all early engines
  - Great user experience in search of a business model
  - Meanwhile Goto/Overture's annual revenues: ~ $1 billion
- Result: Google added paid search "ads" to the side, independent of search results
  - Yahoo followed, acquiring Overture (for paid placement) and Inktomi (for search)
- 2005+: Google gains search share, dominating in Europe and very strong in North America
  - 2009: Yahoo! and Microsoft combined paid search offering



Web Search Basics



User Need on Web Search

- **Informational** – want to learn about something (~40% / 65%)
  `Information Retrieval`
- **Navigational** – want to go to that page (~25% / 15%)
  `United Airlines`
- **Transactional** – want to do something (web-mediated) (~35% / 20%)
  - Access a service `Seattle weather`
  - Downloads `Mars surface images`
  - Shop `Canon S410`
- **Gray areas**
  - Exploratory search "see what's there"

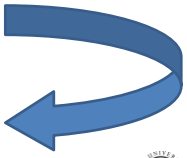
Search Engine Optimization (SEO)

- The Trouble with Paid Search Ads:
  It costs money. What's the alternative?
- ***Search Engine Optimization (SEO):***
  - "Tuning" your web page to rank highly in the algorithmic search results for selected keywords
  - Alternative to paying for placement
  - Thus, intrinsically a marketing function
- Performed by companies, webmasters and consultants ("Search engine optimizers") for their clients
- Some perfectly legitimate, some very shady

SEO: Simplest Form

- First generation engines relied heavily on *tf/idf*
  - The top-ranked pages for the query **`maui resort`** were the ones containing the most **`maui`**'s and **`resort`**'s
- SEOs responded with dense repetitions of chosen terms
  - e.g., **`maui resort maui resort maui resort`**
  - Misleading meta-tags, excessive repetition
  - Often, the repetitions would be in the same color as the background of the web page
    - Repeated terms got indexed by crawlers
    - But not visible to humans on browsers

Pure word density cannot be trusted as an IR signal

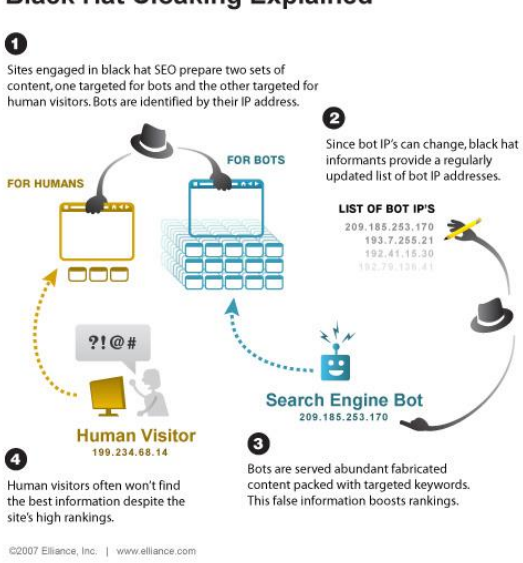


SEO word manipulating examples

- XYZ Hotel in ABC city
  - Accommodation, hotel, room, flat, travel, sights, attractions, vacation, holiday, in ABC ABC ABC
- XYZ for family advices
  - Family, couples, parents, spouse, wife, husband, fights, relationship, cheating, communication, kids, children
- XYZ Umbrellas
  - Raining, rainy, wet, weather, day

SEO: Cloaking

- Serve fake content to search engine spider
- Famous technique: **Black Hat**
- Kind of a spam!



Duplicate Detection

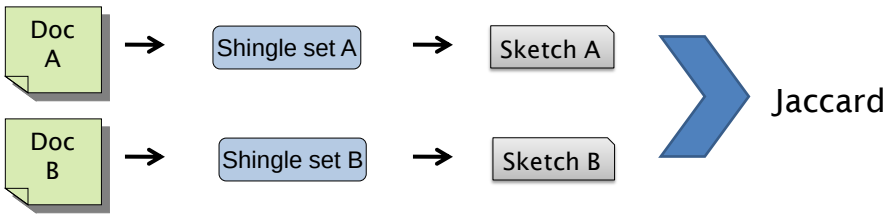
- The web is full of duplicated content
- Strict duplicate detection = exact match
  - Not as common
  - can be detected with fingerprints
- But many, many cases of **near duplicates**
  - e.g., last modified date the only difference between two copies of a page
- *Near-Duplication*: Approximate match
  - Use similarity threshold to detect near-duplicates
    - e.g., Similarity > 80% => Documents are "near duplicates"
    - Not transitive though sometimes used transitively
      - A ≈ B & B ≈ C → doesn't have to mean A ≈ C

Duplicate Detection: MiniHash

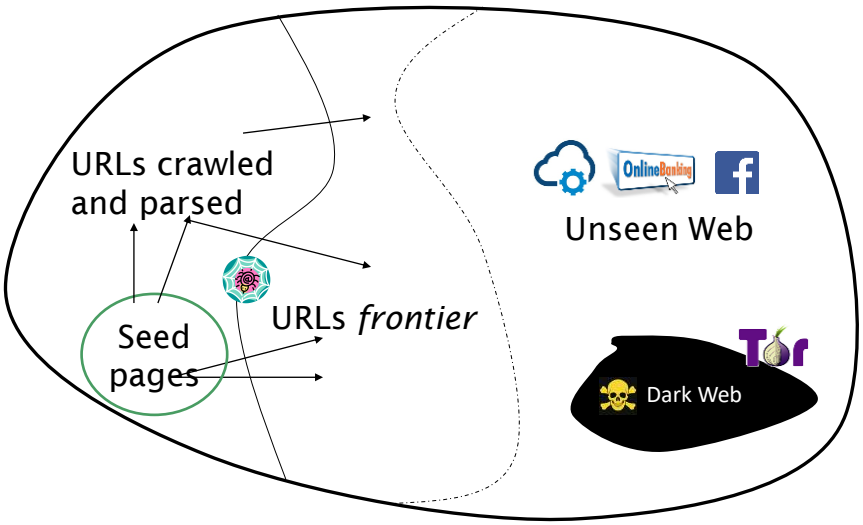
- Features of similarity:
  - Segments of a document (natural or artificial breakpoints)
  - **Shingles** (word n-grams)
  - ***a rose is a rose is a rose*** →
    a_rose_is_a
    rose_is_a_rose
    is_a_rose_is
    a_rose_is_a
- Similarity measure between two docs (= sets of shingles)
  - Set intersection
  - Specifically (Size_of_Intersection / Size_of_Union)

Shingles + Set Intersection

- Computing exact set intersection of shingles between all pairs of documents is expensive/intractable
- Approximate using a cleverly chosen subset of shingles from each (a sketch)
- Estimate $\frac{\textit{size of intersection}}{\textit{size of union}}$ based on a short sketch

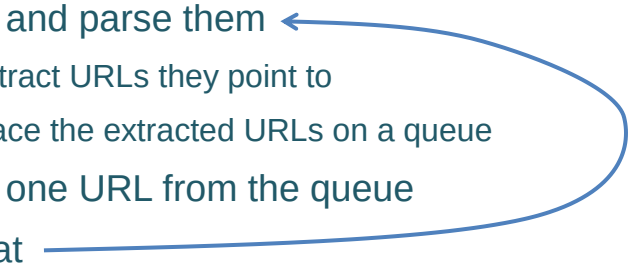


Web Crawling



Basic Crawler Operation

- Begin with known "seed" URLs
- Fetch and parse them
  - Extract URLs they point to
  - Place the extracted URLs on a queue
- Fetch one URL from the queue
- Repeat



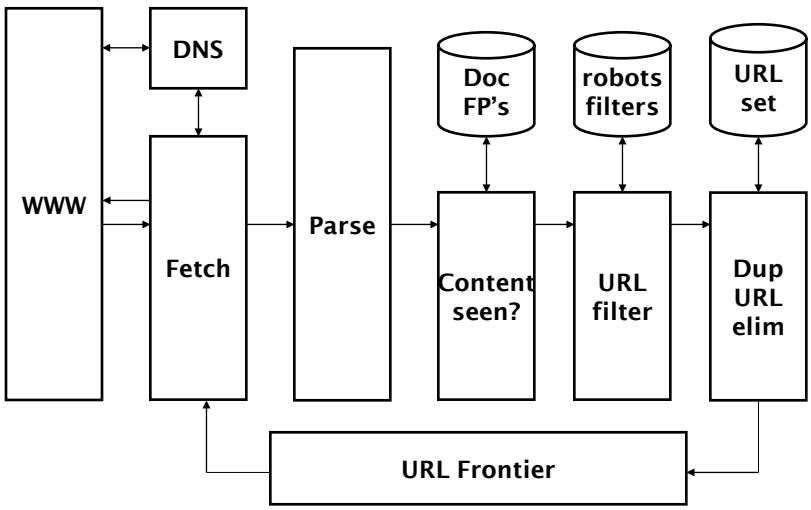
What Any Crawler Must Do

- Be Polite: Respect implicit and explicit politeness considerations
  - Only crawl allowed pages
    - respect `robots.txt`
  - Avoid hitting any site too often
- Be Robust: Be immune to spider traps and other malicious behaviour from web servers
  - Be careful to spams (link farms)

What Any Crawler Should Do

- Be capable of distributed operation
  - designed to run on multiple distributed machines
- Be scalable: designed to increase the crawl rate by adding more machines
- Performance/efficiency: permit full use of available processing and network resources
- Fetch pages of "higher quality" first
- Freshness/Continuous operation: Continue fetching fresh copies of a previously fetched page
- Extensible: Adapt to new data formats, protocols

Basic Crawler Architecture



Processing Steps in Crawling

1. Pick a URL from the frontier
2. Fetch the document at the URL
3. Parse the document
   1. Extract links from it to other docs (URLs)
4. Check if document has content already seen
   1. If not, add to indexes
5. For each extracted URL
   1. Ensure it passes certain URL filter tests
   2. Check if it is already in the frontier (duplicate URL elimination)

URL Frontier

- Can include multiple pages from the same host
- Must avoid trying to fetch them all at the same time
- Must try to keep all crawling threads busy

Explicit and Implicit Politeness

- Explicit politeness: specifications from webmasters on what portions of site can be crawled
  - `robots.txt`
- Implicit politeness: even with no specification, avoid hitting any site too often

```
User-agent: *
Disallow: /yoursite/temp/

User-agent: searchengine
Disallow:
```

- No robot should visit any URL starting with "/yoursite/temp/", except the robot called "searchengine"

URL Frontier: 2 Main Considerations

- Politeness: do not hit a web server too frequently
- Priority/Freshness: crawl some pages more often than others
  - Pages whose content changes often (e.g. News sites)
- These goals may conflict each other.
  - e.g., simple priority queue fails – many links out of a page go to its own site, creating a burst of accesses to that site.
- Even if we restrict only one thread to fetch from a host, can hit it repeatedly
- Common heuristic: insert time gap between successive requests to a host that is >> time taken in most recent fetch from that host

Summary

- History of Web search
- Basics of web search
- Usage of web search
- SEO
- Web crawling

Resources

- Text book 1: Intro to IR, Chapter 19
- Text Book 2: IR in Practice: Chapter 3
- YouTube Videos (nice to watch)
  - How Search Works. Google
    https://www.youtube.com/watch?v=BNHR6IQJGZs
  - The Evolution of Search. Google
    https://www.youtube.com/watch?v=mTBShTwCnD4
  - What Is The Deep Web?. Mashable
    https://www.youtube.com/watch?v=_UOK7aRmUtw
  - Most popular search engines over time
    https://www.youtube.com/watch?v=1a3WL1iOvnE
  - This is How Much YouTube Pays Me
    https://www.youtube.com/watch?v=I3MeCEwVxB0

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Comparing Text Corpora

Instructor:
Björn Ross

10-Nov-2021

1

Initial Text Analysis

- Scenario: you are given access to a new dataset
  - 2 corpora, each contains thousands of plain text files
  - You want to understand and quantify:
    - What is the *content* of these documents? What are they *about*?
    - How does the content of these corpora *differ*?
- What are some things you might try first?

Lecture Objectives

- Analyze text corpora
  - Content analysis background
  - Word-level differences
  - Dictionaries and Lexicons
  - Topic modeling
  - Annotation + classification

Content Analysis

- Goal: given some documents determine
  - What are the types of content present? (themes/topics)
  - Which documents contain which topics?
- Traditionally a manual process
  1. Read a subset of documents, define themes/topics
  2. Determine consistent coding* methodology
  3. Read all documents and label them according to codes
  4. Check agreement between human coders
  5. Settle disagreements via a third-party
  6. Analyze resulting annotations

Content Analysis

- Can this process be automated?
  - Yes, to an extent
- *Should* this process be automated?
  - Humans are better than machines at this task (for now?)
  - Computers are *much*, *much* faster
    - Avg. human reading speed: 250 wpm
    - Assume 1K words/document, 50K documents…
      - Average person needs > 4 months to read
      - This is a ***relatively small*** corpus for modern NLP
    - Modern computers can process millions of words/second

Automated Content Analysis

- Single corpus/class
  - Word frequency analysis
  - Dictionaries & Lexicons
  - Topic modelling

- Multiple corpora/classes
  - Word-level differences
  - Dominance Scores
  - Topic-level differences

Word Level Analysis

Word frequency analysis

- Very simple starting point

1. Preprocess as usual (lowercasing? stemming?...)
2. Count words
3. Normalize by document length
4. Average across all documents

Word-level Differences

- Which words best characterize set of documents (such as a corpus or class)?
  - Need a reference corpus
- Some methods to do this:
  - Mutual information
  - Chi squared
- Can also be used for *feature selection*

Mutual Information

- I(X;Y)
  - How much can I learn about Y by observing X?
  - Is the same as *information gain*
  - Is **not** the same as *pointwise mutual information*
- We want to learn about important words in our class
- What should X and Y be?
  - X = U = document contains term t (Boolean)
  - Y = C = class is the target class (Boolean)

$$I(U;C) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U = e_t, C = e_c) \log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(C = e_c)}$$

Mutual Information

$$I(U;C) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U = e_t, C = e_c) \log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(C = e_c)}$$

- Given a corpus and a term, how do we estimate the probability of this term appearing in a random document in the corpus?

Source: Manning, Raghavan, and Schütze, 2008

Mutual Information

$$I(U;C) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U = e_t, C = e_c) \log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(C = e_c)}$$

- Given count data for 2 classes, can be computed as:

$$I(U;C) = \frac{N_{11}}{N} \log_2 \frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N} \log_2 \frac{NN_{01}}{N_{0.}N_{.1}} + \frac{N_{10}}{N} \log_2 \frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N} \log_2 \frac{NN_{00}}{N_{0.}N_{.0}}$$

Source: Manning, Raghavan, and Schütze, 2008

Mutual Information

$$I(U;C) = \frac{N_{11}}{N}\log_2\frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N}\log_2\frac{NN_{01}}{N_{0.}N_{.1}} + \frac{N_{10}}{N}\log_2\frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N}\log_2\frac{NN_{00}}{N_{0.}N_{.0}}$$

- Example:
  - What is I(U;C) given these values?

	$e_c = e_{poultry} = 1$	$e_c = e_{poultry} = 0$
$e_t = e_{export} = 1$	$N_{11} = 49$	$N_{10} = 27{,}652$
$e_t = e_{export} = 0$	$N_{01} = 141$	$N_{00} = 774{,}106$

Example: Manning, Raghavan, and Schütze, 2008

Mutual Information for News Data

UK	
london	0.1925
uk	0.0755
british	0.0596
stg	0.0555
britain	0.0469
plc	0.0357
england	0.0238
pence	0.0212
pounds	0.0149
english	0.0126

China	
china	0.0997
chinese	0.0523
beijing	0.0444
yuan	0.0344
shanghai	0.0292
hong	0.0198
kong	0.0195
xinhua	0.0155
province	0.0117
taiwan	0.0108

poultry	
poultry	0.0013
meat	0.0008
chicken	0.0006
agriculture	0.0005
avian	0.0004
broiler	0.0003
veterinary	0.0003
birds	0.0003
inspection	0.0003
pathogenic	0.0003

coffee	
coffee	0.0111
bags	0.0042
growers	0.0025
kg	0.0019
colombia	0.0018
brazil	0.0016
export	0.0014
exporters	0.0013
exports	0.0013
crop	0.0012

elections	
election	0.0519
elections	0.0342
polls	0.0339
voters	0.0315
party	0.0303
vote	0.0299
poll	0.0225
candidate	0.0202
campaign	0.0202
democratic	0.0198

sports	
soccer	0.0681
cup	0.0515
match	0.0441
matches	0.0408
played	0.0388
league	0.0386
beat	0.0301
game	0.0299
games	0.0284
team	0.0264

Example: Manning, Raghavan, and Schütze, 2008

Chi-squared

- Hypothesis testing approach
- H_0: Term appearance is independent from a document's class
  - i.e., $P(U = 1, C = 1) = P(U = 1)P(C = 1)$
- Compute:

$$X^2(\mathbb{D}, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}}$$

- Or to directly plug in values like before:

$$X^2(\mathbb{D}, t, c) = \frac{(N_{11} + N_{10} + N_{01} + N_{00}) \times (N_{11}N_{00} - N_{10}N_{01})^2}{(N_{11} + N_{01}) \times (N_{11} + N_{10}) \times (N_{10} + N_{00}) \times (N_{01} + N_{00})}$$

Chi-squared

$$X^2(\mathbb{D}, t, c) = \frac{(N_{11} + N_{10} + N_{01} + N_{00}) \times (N_{11}N_{00} - N_{10}N_{01})^2}{(N_{11} + N_{01}) \times (N_{11} + N_{10}) \times (N_{10} + N_{00}) \times (N_{01} + N_{00})}$$

- Example
  - What is the value of X^2 given the example data?

	$e_c = e_{poultry} = 1$	$e_c = e_{poultry} = 0$
$e_t = e_{export} = 1$	$N_{11} = 49$	$N_{10} = 27{,}652$
$e_t = e_{export} = 0$	$N_{01} = 141$	$N_{00} = 774{,}106$

Dictionaries and Lexicons

Dictionaries and Lexicons

- What if we know what we are looking for?
- Dictionaries (lexicons) are prebuilt mappings
  - Category -> word list
  - E.g., a tiny sentiment lexicon:
    - Positive: good, great, happy, amazing, wonderful, best, incredible
    - Negative: terrible, horrible, bad, awful, nasty, gross, worst, poor
- Domain can be important
  - "***unpredictable*** movie plot" ✓
  - "***unpredictable*** coffee pot" ✗

Dictionaries and Lexicons

- How to get a score per category?

$$\frac{num\_dictionary\_words\_in\_document}{num\_total\_words\_in\_document}$$

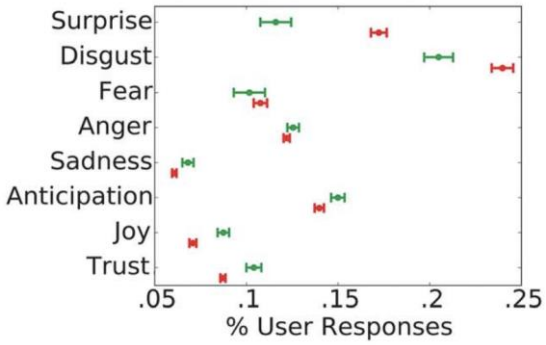
- That's it!
- Can also be used as machine learning features

- A more advanced approaches to quantifying categories (optional reading)
  - https://www.ncbi.nlm.nih.gov/pubmed/28364281

Some Dictionaries

- LIWC (Pennebaker et al. 2015)
- General Inquirer (Stone 1997)
- Roget's Thesaurus Categories
- VADER (Hutto and Gilbert, 2014)
- Sentiwordnet (Esuli and Sebastiani 2006)
- Wordnet Domains (Magnini and Cavaglia, 2000)
- EmoLex (Mohammad and Turney, 2010)
- Empath (Fast et al., 2016)
- Personal Values Lexicon (Wilson et al., 2018)
- …

Reactions to Rumor Tweets with EmoLex



Red = reactions to false rumors
Green = reactions to true rumors

Vosoughi, Roy, and Aral, 2018

Dominance Scores

- The dominance score for a category w.r.t. a corpus:

$$\frac{category\_score\_in\_target\_corpus}{category\_score\_in\_background\_corpus}$$

  - From Mihalcea and Pulman, 2009

LIWC category dominance scores

Truthful			
Interviews		Trials	
Class	Score	Class	Score
Metaphor	2.98	You	3.99
Money	2.74	Family	3.07
Inhibition	2.74	Home	2.45
Home	2.13	Humans	1.87
Humans	2.02	Posemo	1.81
Family	1.96	Insight	1.64

Deceptive			
Interviews		Trials	
Class	Score	Class	Score
Assent	4.81	Anger	2.61
Past	2.59	Anxiety	2.61
Sexual	2.00	Certain	2.28
Other	1.87	Death	1.96
Motion	1.68	Physical	1.77
Negemo	1.44	Negemo	1.52

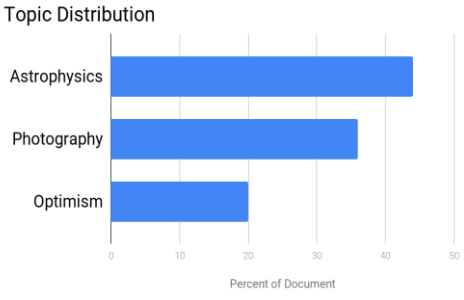
Pérez-Rosas et al, 2015

Topic Level Analysis

Intro to Topic Modelling

- Goals are similar to traditional content analysis:
  - What are the main themes/topics in this corpus?
  - Which documents contain which topics?

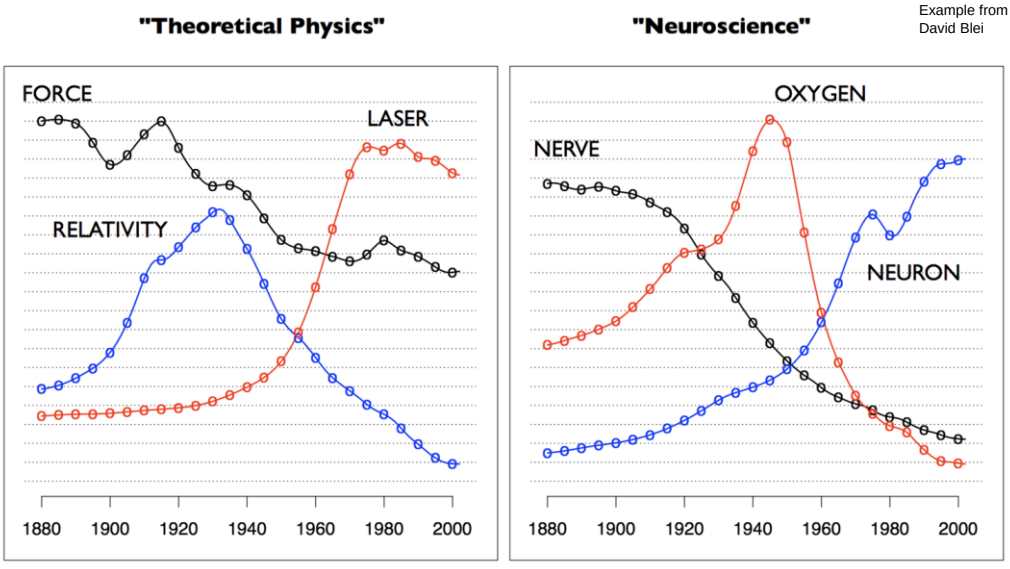
Topic Models



30

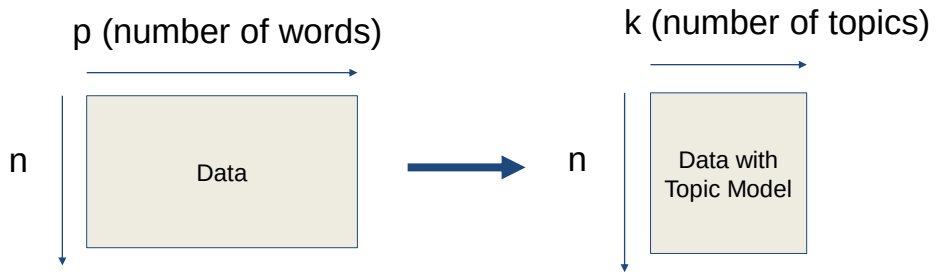
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

Example from David Blei



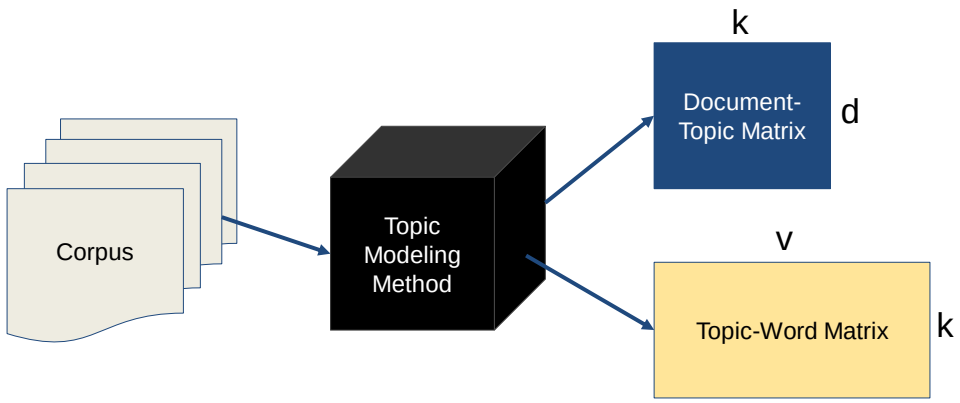
Example from David Blei

Dimensionality Reduction

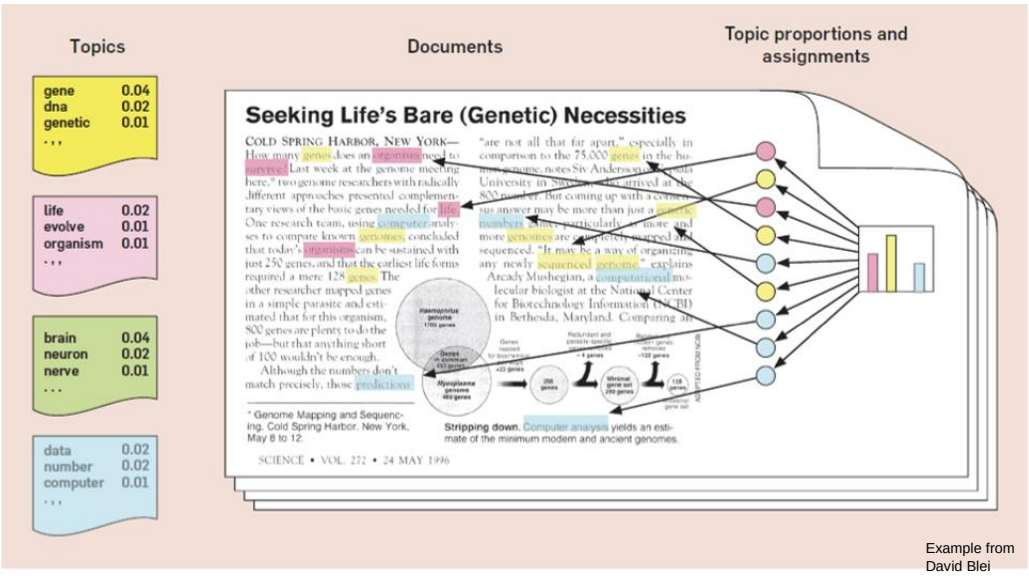


33

Topic Modeling



34



Example from David Blei

Topic Models

- Most often used for text data, but can also be applied in other settings:
  - Bioinformatics (Liu et al. 2016)
  - Computer code (McBurney et al. 2014)
  - Music (Hu and Saul 2009)
  - Network data (Cha and Cho 2014)

Topic Modeling Methods

- Most popular: Latent Dirichlet Allocation (LDA)
  - Introduced by David Blei, Andrew Ng, and Michael Jordan (2003)
- Other methods include
  - pLSI
  - PCA-based methods
  - Non-negative matrix factorization
  - Deep learning based topic modeling
  - ...

37

Topic Modeling Methods

- Most popular: Latent Dirichlet Allocation (LDA)
  - Introduced by David Blei, Andrew Ng, and Michael Jordan (2003)
- Other methods include
  - pLSI
  - PCA-based methods
  - Non-negative matrix factorization
  - Deep learning based topic modeling
  - ...

38

Latent Dirichlet Allocation (LDA)

- More details coming up in next lecture…

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Comparing Text Corpora (2)

Instructor:
Björn Ross

10-Nov-2021

1

LDA Overview

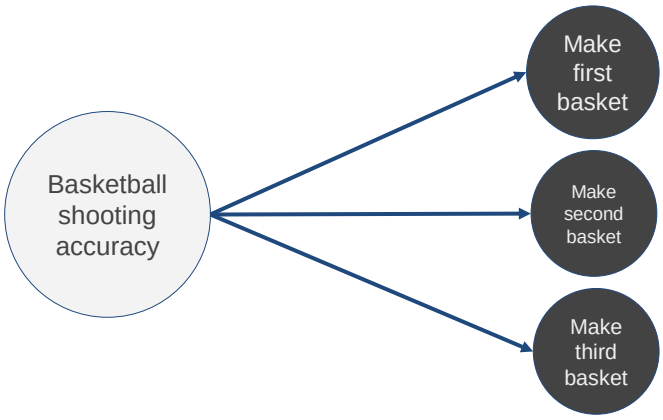
Background: Plate Notation

Make a basket

Background: Plate Notation

Basketball shooting accuracy → Make a basket

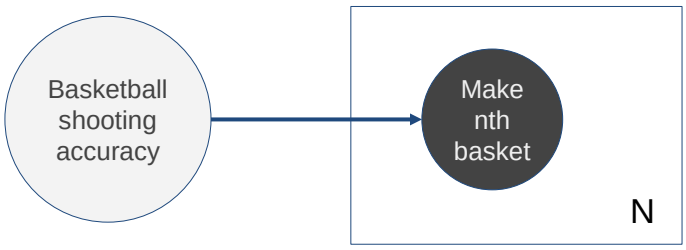
Background: Plate Notation



5

5

Background: Plate Notation



6

6

Latent Dirichlet Allocation

- Let's start with a very simple model
- We will work our way up to the full LDA model

Unigram Model

w is a word
N words in a document

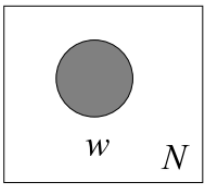


Figure from
Blei et al 2003

8

Unigram Model

w is a word
N words in a document
M documents in a corpus

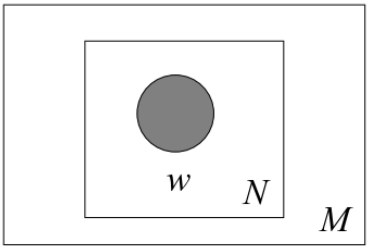
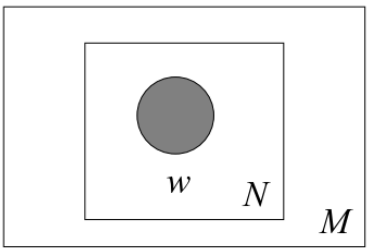


Figure from
Blei et al 2003

9

Unigram Model

w is a word
N words in a document
M documents in a corpus
w is a vector of words (i.e. doc)



$$p(\mathbf{w}) = \prod_{n=1}^{N} p(w_n)$$

Figure from
Blei et al 2003

10

Probability with a Unigram Model

$$p(\mathbf{w}) = \prod_{n=1}^{N} p(w_n)$$

What is the probability of the example sentence?

"My dog barked at another dog."

word	my	at	dog	another	barked
probability	.10	.10	.05	.04	.03

11

Unigram Model…

- What is the point of making these models more complex?
- Why not just use the basic unigram model for everything?

- Remember:
  - Higher text probability ***doesn't imply a better model***
  - We want to ***accurately describe*** the data
    - → higher probability for *real* documents, lower probability for noise

Mixture of Unigrams Model

z is the topic of a document

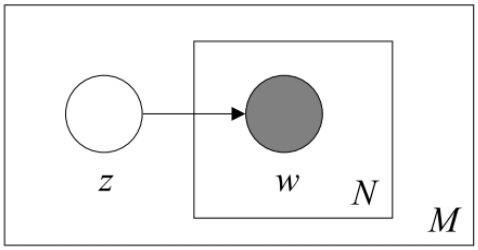


Figure from
Blei et al 2003

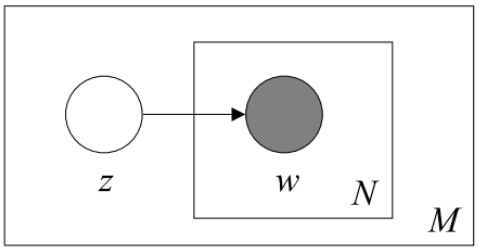
14

THE UNIVERSITY of EDINBURGH

14

Mixture of Unigrams Model

z is the topic of a document



$$p(\mathbf{w}) = \sum_{z} p(z) \prod_{n=1}^{N} p(w_n \mid z)$$

Figure from
Blei et al 2003

15

THE UNIVERSITY of EDINBURGH

15

Probability with Mixture of Unigrams

$$p(\mathbf{w}) = \sum_z p(z) \prod_{n=1}^{N} p(w_n | z).$$

What is the probability of the sentence?
Ignore stopwords: “my”, “after”, “the”

“My dog chased after the bus.”

w_i	cat	dog	chased	car	bus
$P(w_i \mid$ z = pets)	.20	.30	.10	.01	.01
$P(w_i \mid$ z = vehicles)	.01	.01	.10	.30	.20

$p(z = pets) = 0.6,$
$p(z = vehicles) = 0.4$

16

Probabilistic Latent Semantic Indexing

d is a document ID

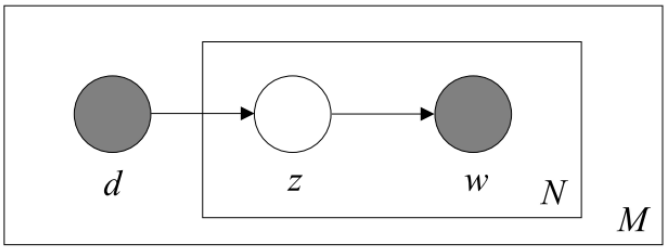
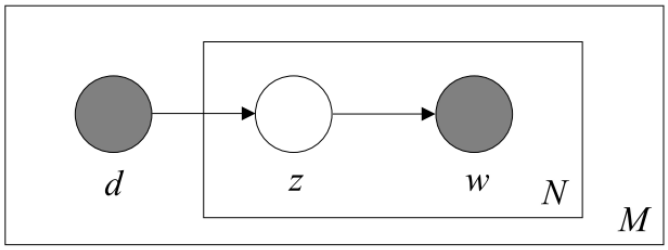


Figure from Blei et al 2003

18

Probabilistic Latent Semantic Indexing

d is a document ID



$$p(d, w_n) = p(d) \sum_z p(w_n | z) p(z | d)$$

Figure from Blei et al 2003

19

Probability with pLSI

w_i	cat	sat	down	car	broke
$p(w_i \mid z = t_1)$	.2	.1	.05	.01	.1
$p(w_i \mid z = t_2)$	.01	.05	.1	.3	.1

d_1 "The **cat** sat down."

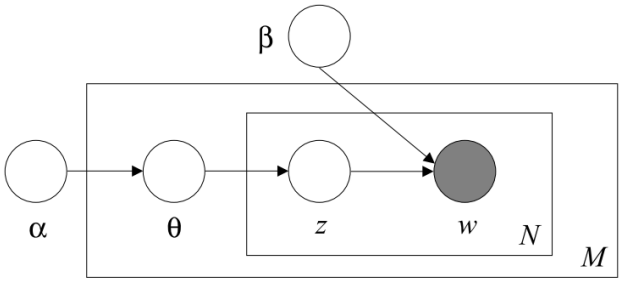
$p(d = d_1)$	.01
$p(z = t_1 \mid d = d_1)$	.6
$p(z = t_2 \mid d = d_1)$	.4

w_i	cat	sat	down	car	broke
$p(w_i \mid z = t_1)$	.2	.1	.05	.01	.1
$p(w_i \mid z = t_2)$	.01	.05	.1	.3	.1

What is the joint probability of the document and the word "cat"?

20

Latent Dirichlet Allocation

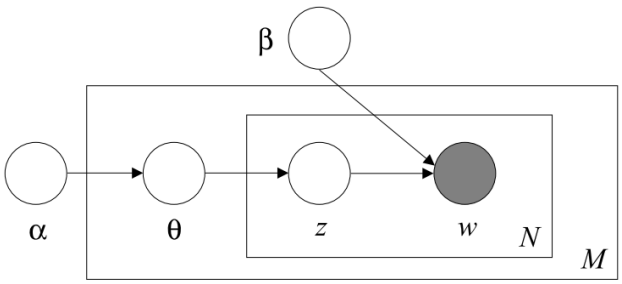


θ is the distribution over topics in a document
α is the parameter of a Dirichlet distribution giving possible topic distributions within documents
β gives word distributions within topics

Figure from Blei et al 2003

22

Latent Dirichlet Allocation



$$p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta) = p(\theta \mid \alpha) \prod_{n=1}^{N} p(z_n \mid \theta) p(w_n \mid z_n, \beta)$$

Figure from Blei et al 2003

23

Latent Dirichlet Allocation

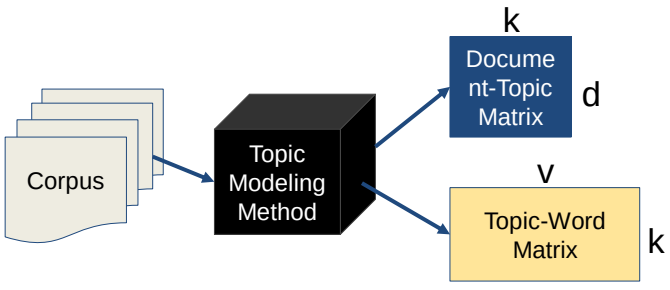
$$p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta) = p(\theta \mid \alpha) \prod_{n=1}^{N} p(z_n \mid \theta) p(w_n \mid z_n, \beta)$$

$$p(\mathbf{w} \mid \alpha, \beta) = \int p(\theta \mid \alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n \mid \theta) p(w_n \mid z_n, \beta) \right) d\theta$$

$$p(D \mid \alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d \mid \alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn} \mid \theta_d) p(w_{dn} \mid z_{dn}, \beta) \right) d\theta_d$$

Model Inference

- Want to learn the model parameters
- Exact inference becomes intractable



Model Inference

- Instead, use an approximate method such as:
  - Gibbs sampling
  - Variational Inference

28

Gibbs Sampling for LDA

Goal: Learn Φ, θ given a set of documents D

Φ = topic-word probabilities
θ = document-topic probabilities

Known:
corpus, α, β and the probability that a word is from a topic conditional on the assignments of all other words to topics

$$P(z_i = j \mid \mathbf{z}_{-i}, w_i, d_i, \cdot) \propto \frac{C^{WT}_{w_i j} + \beta}{\sum_{w=1}^{W} C^{WT}_{wj} + W\beta} \frac{C^{DT}_{d_i j} + \alpha}{\sum_{t=1}^{T} C^{DT}_{d_i t} + T\alpha}$$

29

Gibbs Sampling for LDA

Want to learn Φ, θ given a set of documents D

1. Assign each word a topic randomly
2. Calculate count matrices
3. Repeat until convergence:
   - For every document d
     - For every word i
       - Decrement count matrices C^{WT} and C^{DT} for current topic assignment
       - Sample a new topic assignment
       - Increment count matrices C^{WT} and C^{DT} for new topic assignment
4. Calculate Φ and θ

Gibbs Sampling for LDA

d1 Green eggs and ham.
d2 Ham and green peppers.
d3 Ham and cheese.

Green eggs and ham.
Ham and green peppers.
Ham and cheese.

Random initialization.

Gibbs Sampling for LDA

C^{WT}	green	eggs	and	ham	peppers	cheese
t1	1	1	1	1	1	1
t2	1	0	2	2	0	0

Green eggs and ham.
Ham and green peppers.
Ham and cheese.

C^{DT}	d1	d2	d3
t1	1	1	1
t2	1	1	1

32

Gibbs Sampling for LDA

Assume (for the moment) $\alpha = \beta = 0$

θ	green	eggs	and	ham	peppers	cheese
t1	0.17	0.17	0.17	0.17	0.17	0.17
t2	0.20	0.00	0.40	0.40	0.00	0.00

Green eggs and ham.
Ham and green peppers.
Ham and cheese.

Φ	d1	d2	d3
t1	0.50	0.50	0.66
t2	0.50	0.50	0.33

33

Gibbs Sampling for LDA

C^{WT}	green	eggs	and	ham	peppers	cheese
t1	1	1	1	1	1	1
t2	1	0	2	2	0	0

Green eggs and ham.
Ham and green peppers.
Ham and cheese.

C^{DT}	d1	d2	d3
t1	2	2	2
t2	2	2	1

34

34

Gibbs Sampling for LDA

C^{WT}	green	eggs	and	ham	peppers	cheese
t1	1	1	1	1	1	1
t2	1	0	2	2	0	0

Green eggs and ham.
Ham and green peppers.
Ham and cheese.

C^{DT}	d1	d2	d3
t1	2	2	2
t2	2	2	1

35

35

Gibbs Sampling for LDA

$$\frac{C^{WT}_{w_i,j}+\beta}{\sum_{w=1}^{W} C^{WT}_{wj}+W\beta}\;\frac{C^{DT}_{d_i,j}+\alpha}{\sum_{t=1}^{T} C^{DT}_{d_i t}+T\alpha}$$

Assume (for the moment) $\alpha = \beta = 0$

C^{WT}	green	eggs	and	ham	peppers	cheese
t1	0	1	1	1	1	1
t2	1	0	2	2	0	0

Green eggs and ham.
Ham and green peppers.
Ham and cheese.

C^{DT}	d1	d2	d3
t1	1	2	2
t2	2	2	1

36

Gibbs Sampling for LDA

C^{WT}	green	eggs	and	ham	peppers	cheese
t1	0	1	1	1	1	1
t2	2	0	2	2	0	0

Green eggs and ham.
Ham and green peppers.
Ham and cheese.

C^{DT}	d1	d2	d3
t1	1	2	2
t2	3	2	1

37

Gibbs Sampling for LDA

C^{WT}	green	eggs	and	ham	peppers	cheese
t1	0	1	1	1	1	1
t2	2	0	2	2	0	0

Green eggs and ham.
Ham and green peppers.
Ham and cheese.

C^{DT}	d1	d2	d3
t1	1	2	2
t2	3	2	1

38

Gibbs Sampling for LDA

$$\frac{C^{WT}_{w_i j} + \beta}{\sum_{w=1}^{W} C^{WT}_{wj} + W\beta} \frac{C^{DT}_{d_i j} + \alpha}{\sum_{t=1}^{T} C^{DT}_{d_i t} + T\alpha}$$

Assume (for the moment) $\alpha = \beta = 0$

C^{WT}	green	eggs	and	ham	peppers	cheese
t1	0	0	1	1	1	1
t2	2	0	2	2	0	0

Green eggs and ham.
Ham and green peppers.
Ham and cheese.

C^{DT}	d1	d2	d3
t1	0	2	2
t2	3	2	1

39

Gibbs Sampling for LDA

$$\frac{C^{WT}_{w_i,j} + \beta}{\sum_{w=1}^{W} C^{WT}_{wj} + W\beta} \frac{C^{DT}_{d_i,j} + \alpha}{\sum_{t=1}^{T} C^{DT}_{d_i t} + T\alpha}$$

$C^{WT} + \alpha$	green	eggs	and	ham	peppers	cheese
t1	0.01	0.01	1.01	1.01	1.01	1.01
t2	2.01	0.01	2.01	2.01	0.01	0.01

Green eggs and ham.
Ham and green peppers.
Ham and cheese.

$C^{DT} + \beta$	d1	d2	d3
t1	0.01	2.01	2.01
t2	3.01	2.01	1.01

40

Gibbs Sampling for LDA

- Repeat until convergence
- Probabilistic algorithm – results depend on random initialisation and random samples!

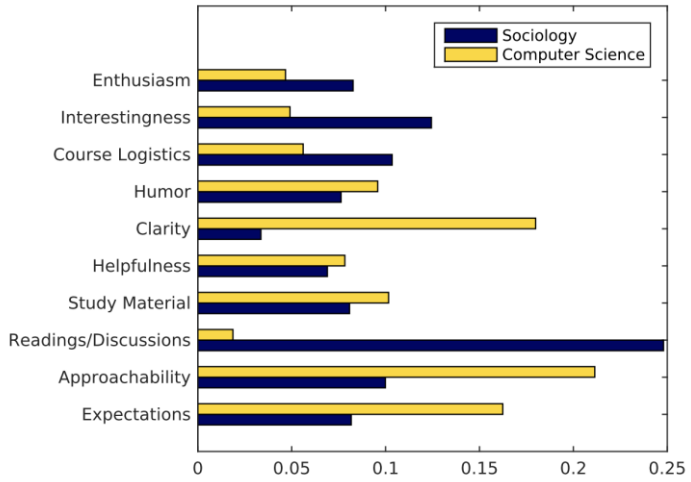
Topic Modeling Examples

What do students look for in a professor?

Topic	Sample words
Approachability	prof, fair, clear, helpful, teaching, approachable, nice, organized, extremely, friendly, super, amazing
Clarity	understand, hard, homework, office, material, clear, helpful, problems, explains, accent, questions, extremely
Course Logistics	book, study, boring, extra, nice, credit, lot, hard, attendance, make, fine, attention, pay, mandatory
Enthusiasm	teaching, passionate, awesome, enthusiastic, professors, loves, cares, wonderful, fantastic, passion
Expectations	hard, work, time, lot, comments, tough, expects, worst, stuff, avoid, horrible, classes
Helpfulness	helpful, nice, recommend, cares, super, understanding, kind, extremely, effort, sweet, friendly, approachable
Humor	guy, funny, fun, awesome, cool, entertaining, humor, hilarious, jokes, stories, love, hot, enjoyable
Interestingness	interesting, material, recommend, lecturer, engaging, classes, knowledgeable, enjoyed, loved, topics
Readings/ Discussions	readings, papers, writing, ta, interesting, discussions, grader, essays, boring, books, participation
Study Material	exams, notes, questions, material, textbook, hard, slides, study, answer, clear, tricky, attend, long, understand

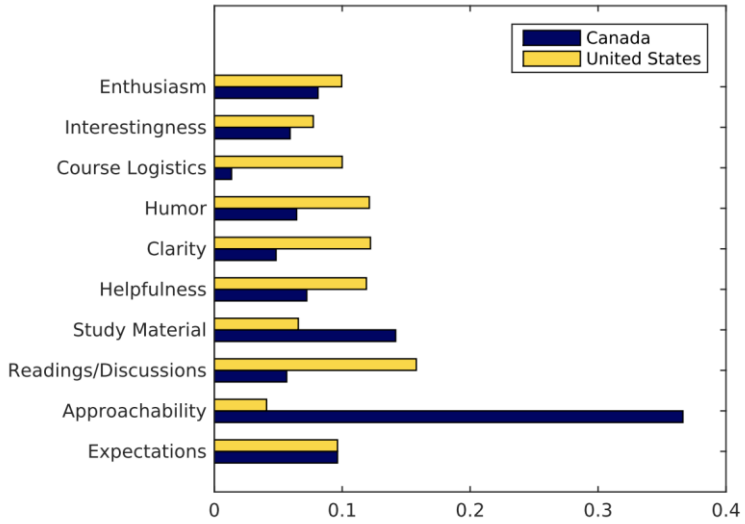
Azab, Mihalcea, and Abernathy, 2016

What do students look for in a professor?



Azab, Mihalcea, and Abernathy, 2016

What do students look for in a professor?

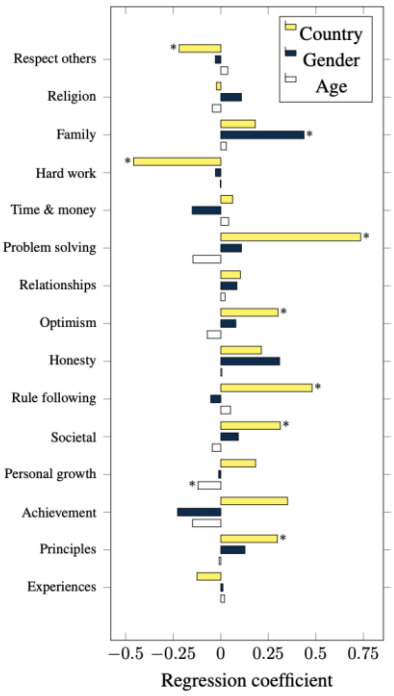


Azab, Mihalcea, and Abernathy, 2016

How do personal attributes relate to values?

Theme	Example Words
Respect others	people, respect, care, human, treat
Religion	god, heart, belief, religion, right
Family	family, parent, child, husband, mother
Hard Work	hard, work, better, honest, best
Time & Money	money, work, time, day, year
Problem solving	consider, decision, situation, problem
Relationships	family, friend, relationship, love
Optimism	enjoy, happy, positive, future, grow
Honesty	honest, truth, lie, trust, true
Rule following	moral, rule, principle, follow
Societal	society, person, feel, thought, quality
Personal Growth	personal, grow, best, decision, mind
Achievement	heart, achieve, complete, goal
Principles	important, guide, principle, central
Experiences	look, see, experience, choose, feel

Wilson, Mihalcea, Boyd, and Pennebaker 2016



46

Annotation + Classification

47

Annotation + Classification

- Method 1: Traditional Supervised Learning
  - Annotate representative samples
  - Train a classifier
  - Apply to rest of data
- Method 2: Transfer Learning
  - Find another large, but similar dataset
  - Train a classifier on that dataset
  - *Optionally: fine-tune classifier to your smaller dataset*
  - Apply to rest of your data

After Classification

- Which features are most relevant for each class?
- What are common words/topics for each class?
- How do predicted classes relate to other variables?

- *More about text classification coming up next week!*

Wrap-up

- Content analysis background
- Word-level differences
- Dictionaries and Lexica
- Topic modeling
- Annotation + classification

Readings

- Manning: IR book section 13.5
- "Probabilistic Topic Models" by David Blei
- "Latent Dirichlet Allocation" by David Blei, Andrew Y. Ng, and Michael I. Jordan
- "Probabilistic Topic Models" by Mark Steyvers and Tom Griffiths

To watch:

- Guest lecture (2017) by David Blei at University of Edinburgh School of Informatics

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Text Classification

Instructor:
Björn Ross

17-Nov-2021

1

Lecture Objectives

- Learn about text basics of text classification
  - Definition
  - Types
  - Methods
  - Evaluation

Text Classification

- **Text classification** is the process of classifying documents into predefined categories based on their content.

- Input: Text (document, article, sentence)
- Task: Classify into predefined one/multiple categories
- Categories:
  - Binary: relevant/irrelevant, spam .. etc.
  - Few: sports/politics/comedy/technology
  - Hierarchical: patents

Classification is and is not

- Classification (a.k.a. "categorization"): a common technology in data science; studied within pattern recognition, statistics, and machine learning.
- Definition:
  the activity of predicting to which among a predefined finite set of groups ("classes", or "categories") a data item belongs to
- Formulated as the task of generating a hypothesis (or "classifier", or "model")

$$h : D \rightarrow C$$

where $D = \{\mathbf{x}_1, \mathbf{x}_2, ...\}$ is a domain of data items and
$C = \{c_1, ..., c_n\}$ is a finite set of classes (the classification scheme)

Classification is and is not

- Different from clustering, where the groups ("clusters") and their number are not known in advance
- The membership of a data item into a class must not be determinable with certainty
  - e.g., predicting whether a natural number belongs to *Prime* or *Non-Prime* is not classification
- In text classification, data items are
  - **Textual**: e.g., news articles, emails, sentences, queries, etc.
  - **Partly textual**: e.g., Web pages

Types of Classification

- **Binary**:
  item to be classified into one of two classes
  $h : D \rightarrow C$, $C = \{c_1, c_2\}$
  - e.g., Spam/not spam, offensive/not offensive, rel/irrel
- **Single-Label Multi-Class** (SLMC)
  item to be classified into only one of *n* possible classes.
  $h : D \rightarrow C$, $C = \{c_1 \ldots. c_n\}$, where n>2
  - e.g., Sports/politics/entertainment, positive/negative/neutral
- **Multi-Label Multi-Class** (MLMC)
  item to be classified into none, one, two, or more classes
  $h : D \rightarrow 2^C$, $C = \{c_1 \ldots. c_n\}$, where n>1
  - e.g., Assigning CS articles to classes in the ACM Classification System
  - Usually be solved as *n* independent binary classification problems

Dimension of Classification

- Text classification may be performed according to several dimensions ("axes") orthogonal to each other
- by topic; by far the most frequent case, its applications are global
- by sentiment; useful in market research, online reputation management, social science and political science
- by language (a.k.a. "language identification"); useful, e.g., in query processing within search engines
- by genre; e.g., AutomotiveNews vs. AutomotiveBlogs, useful in website classification and others;
- by author (a.k.a. "authorship attribution"), by native language ("native language identification"), or by gender; useful in forensics and cybersecurity
- by usefulness; e.g., product reviews
- ……

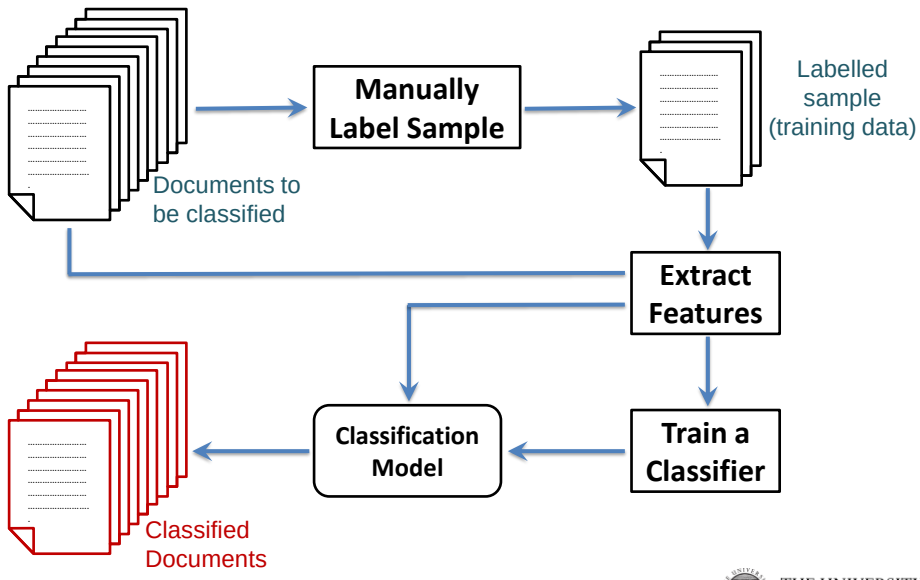
Rule-based classification

- An old-fashioned way to build text classifiers was via knowledge engineering, i.e., manually building classification rules
  - E.g., (Viagra or Sildenafil or Cialis) → Spam
  - E.g. (#MAGA or America great again) → support Trump
- Common type: dictionary-based classification
- Disadvantages:
  - Expensive to setup and to maintain
  - Depends on few keywords → bad coverage (recall)

Supervised-learning classification

- A generic (task-independent) learning algorithm is used to train a classifier from a set of manually classified examples
- The classifier learns, from these training examples, the characteristics a new text should have in order to be assigned to class *c*
- Advantages:
  - Generating training examples cheaper than writing classification rules
  - Easy update to changing conditions (e.g., addition of new classes, deletion of existing classes, shifted meaning of existing classes, etc.)

Supervised-learning classification



Extract Features

- In order to be input to a learning algorithm (or a classifier), all training (or unlabeled) documents are converted into vectors in a common vector space
- The dimensions of the vector space are called features
- In order to generate a vector-based representation for a set of documents *D*, the following steps need to be taken
  1. Feature Extraction
  2. Feature Selection or Feature Synthesis (optional)
  3. Feature Weighting

Step 1: Feature Extraction

- What are the features that should be different from one class to another?
- Simplest form: Bag-of-words (BOW)
  - Each term in a document is a feature
  - Feature space size = vocabulary in all docs
  - Standard IR preprocessing steps are usually applied
    - Tokenisation, stopping, stemming
- Other simple features forms:
  - Word n-grams (bigrams, trigrams, ….)
    - Much larger + more sparse
  - Sometimes char n-grams are used
    - Especially for degraded text (OCR or ASR outputs)

Step 1: Feature Extraction

- What other text features could be used?
- Sentence structure:
  - POS (part-of-speech tags)
  - Syntactic tree structure
- Topic-based features:
  - LDA topics
  - NEs (named entities) in text
  - Links / Linked terms
- Non-textual features:
  - Average doc\sentence\word length
  - % of words start with upper-case letter
  - % of links/hashtags/emojis in text

Step 1: Feature Extraction

- What preprocessing to apply?
  - Case-folding? really vs Really vs REALLY
  - Punctuation? "?", "!", "@", "#"
  - Stopping? "he", "she", "what", "but"
  - Stemming? "replaced" vs "replacement"
- Other Features:
  - Starts with capital letter, all caps
  - Repeated characters "congraaaaaats" "help!!!!!!!!"
  - Scores from dictionaries and lexicons (e.g. LIWC)
- Which to choose?
  - Classification task/application

Step 2: Feature Selection

- Number of distinctive features = length of feature vector
- Vector can be of length in the order of 10^6, and might be sparse
  → High computational cost
  → Overfitting
- What are the most important features among those?
  - e.g. Reduce from 10^6 to 10^4
- For each class, find the top representative *k* features for it → get the Union over all classes → reduced feature space

Step 2: Feature Selection Functions

- Document frequency
  - % of docs in class c_i that contain the term t_k
  - Very basic measure. Will select stop words as features

$$\#(t_k, c_i) = P(t_k|c_i)$$

- Mutual Information
  - How much we learn from the presence or absence of term t_k about whether or not a document is in class c_i
  - Often used in feature selection in text classification

$$MI(t_k, c_i) = \sum_{c \in \{c_i, \bar{c}_i\}} \sum_{t \in \{t_k, \bar{t}_k\}} P(t, c) \cdot log_2 \frac{P(t, c)}{P(t) \cdot P(c)}$$

- Pearson's Chi-squared (x^2)
  - used more in comparisons between classes

Step 2: Feature Selection Functions

Function	Denoted by	Mathematical form
Document frequency	$\#(t_k, c_i)$	$P(t_k \mid c_i)$
DIA association factor	$z(t_k, c_i)$	$P(c_i \mid t_k)$
Information gain	$IG(t_k, c_i)$	$\sum_{c \in \{c_i, \overline{c}_i\}} \sum_{t \in \{t_k, \overline{t}_k\}} P(t, c) \cdot \log \frac{P(t, c)}{P(t) \cdot P(c)}$
Mutual information	$MI(t_k, c_i)$	$\log \frac{P(t_k, c_i)}{P(t_k) \cdot P(c_i)}$
Chi-square	$\chi^2(t_k, c_i)$	$\frac{\lvert Tr\rvert \cdot [P(t_k, c_i) \cdot P(\overline{t}_k, \overline{c}_i) - P(t_k, \overline{c}_i) \cdot P(\overline{t}_k, c_i)]^2}{P(t_k) \cdot P(\overline{t}_k) \cdot P(c_i) \cdot P(\overline{c}_i)}$
NGL coefficient	$NGL(t_k, c_i)$	$\frac{\sqrt{\lvert Tr\rvert} \cdot [P(t_k, c_i) \cdot P(\overline{t}_k, \overline{c}_i) - P(t_k, \overline{c}_i) \cdot P(\overline{t}_k, c_i)]}{\sqrt{P(t_k) \cdot P(\overline{t}_k) \cdot P(c_i) \cdot P(\overline{c}_i)}}$
Relevancy score	$RS(t_k, c_i)$	$\log \frac{P(t_k \mid c_i) + d}{P(\overline{t}_k \mid \overline{c}_i) + d}$
Odds Ratio	$OR(t_k, c_i)$	$\frac{P(t_k \mid c_i) \cdot (1 - P(t_k \mid \overline{c}_i))}{(1 - P(t_k \mid c_i)) \cdot P(t_k \mid \overline{c}_i)}$
GSS coefficient	$GSS(t_k, c_i)$	$P(t_k, c_i) \cdot P(\overline{t}_k, \overline{c}_i) - P(t_k, \overline{c}_i) \cdot P(\overline{t}_k, c_i)$

Step 2: Feature Synthesis

- Matrix decomposition techniques (e.g., PCA, SVD, LSI) can be used to synthesize new features that replace the features discussed above
- Principle of distributional semantics: semantics of a word "is" the words it co-occurs with
  - **Pros**: the synthetic features in the new vector representation do not suffer from problems such as polysemy and synonymy
  - **Cons**: computationally expensive
- Word embeddings: the new wave of distributional semantics, modern approaches are based on neural networks

- PCA: Principle component analysis
- SVD: Singular value decomposition
- LSA: latent semantic analysis

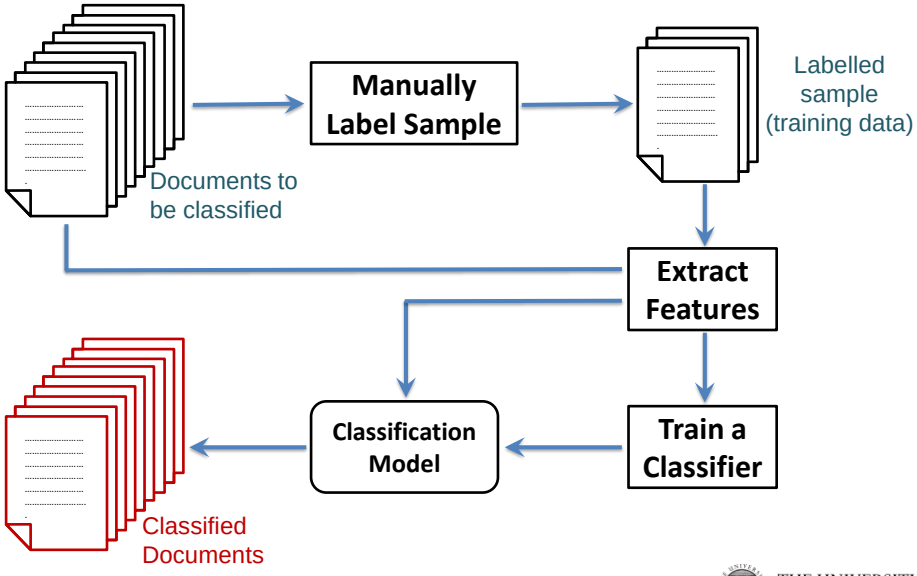
Step 2: Feature Synthesis

- Deep learning?
- Language modelling "features"
  - Tokenize text and pass to neural network layer
    - E.g., recurrent layer, convolutional layer, self-attention layer
  - Stack on 3+ more layers
  - Train a model to predict the next word (or a missing word) given previous words
  - Penultimate layer of network can be used to generate features for other language-based tasks
  - Basis for many state-of-the-art text classifiers
    - BERT, GPT, Electra, XLNet, etc.

Step 3: Feature Weighting

- Attributing a value to feature t_k in document d_i
  This value may be
  - binary (representing presence/absence of t_k in d_i);
  - numeric (representing the importance of t_k for d_i); obtained via feature weighting functions in the following two classes:
    - unsupervised: e.g., tfidf or BM25,
    - supervised: e.g., $tf * MI$, $tf * x^2$
- Similarity between two vectors may be computed e.g. via cosine similarity
- Scaling can be important!

Supervised-learning classification



Training a Classifier

- For binary classification, essentially any supervised learning algorithm can be used for training a classifier; classical choices include
  - Support vector machines (SVMs)
  - Random forests
  - Naïve Bayesian methods
  - Lazy learning methods (e.g., k-NN)
  - Logistic Regression
  - ….
- The "No-free-lunch principle" (Wolpert, 1996) → *there is no learning algorithm that can outperform all others in all contexts*
- Implementations need to cater for
  - the very high dimensionality
  - the sparse nature of the representations involved

Training a Classifier

- For Multiclass classification, some learning algorithms for binary classification are “SLMC-ready”; e.g.
  - Decision trees
  - Random forests
  - Naive Bayesian methods
  - Lazy learning methods (e.g., k-NN)
  - Neural networks
- For other learners (notably: SVMs) to be used for SLMC classification, combinations / cascades of the binary versions need to be used
  - e.g. multi-class classification SVM
  - Could be directly used for MLMC as well

Holding out test data

- It’s important to avoid overfitting
- Labelled data could be split into two parts
  - Training: used to train the classifier (e.g. **80%** of the data)
  - Test: used to test the performance of the trained classifier on unseen data (e.g. **20%** of the data)

Hyperparameter optimisation

- Most classifiers have some hyperparameters to be optimized (we will usually refer to the ones we set manually as “hyperparameters” to distinguish from the ”learned” parameters/weights of the model)
  - The *C* parameter in soft-margin SVMs
  - The *r*, *d* parameters of non-linear kernels
  - Decision threshold for binary SVM
- Usually labelled data is split into three parts
  - Training: used to train the classifier (typically **80%** of the data)
  - Development: used to optimise hyperparameters. Apply the classifier on this data with different values of the hyperparameters and report the one that achieves the highest results (usually **10%** of the data)
  - Test: used to test the performance of the trained classifier with the optimal hyperparameters on these unseen data (usually **10%** of the data)
- Optimising the hyperparameters on test data is cheating!

Evaluation

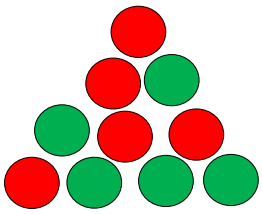
- Effectiveness (e.g. accuracy, precision, recall, F1):
  - Global effectiveness measures
  - Per class effectiveness measures
- Efficiency:
  - Speed in learning
    - SVM with linear kernel is known to be fast
    - DNNs are known to be much slower (specially with large # layers)
  - Speed in classification
    - K-NNs are known to be one of the slowest
  - Speed in feature extraction
    - BOW vs POS vs Link analysis features
- Importance of baselines

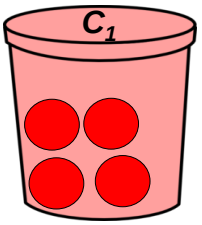
Evaluation: Baselines

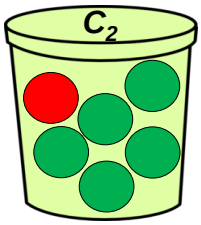
- There are standard methods for creating baselines in text classification to compare your classifier with
- Most popular/simplest baselines
  - Random classification
    - Classes are assigned randomly
    - How much better is the classifier doing than random?
  - Majority class baseline
    - Assign all elements to the class that appears the most
    - How much better you are doing than if you always picked the same thing output regardless of input?
  - Simple algorithm, e.g. BOW
    - Usually used when you introduce new interesting features

Evaluation: Binary Classification

- Accuracy:
  - How many of the samples are classified correctly?
- A = (4+5)/10 = 0.9

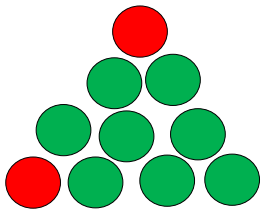


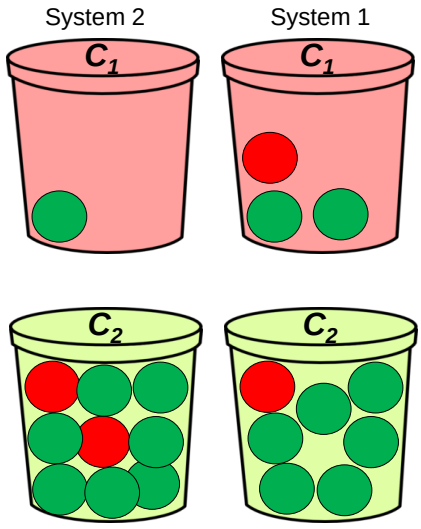




Evaluation: Binary Classification

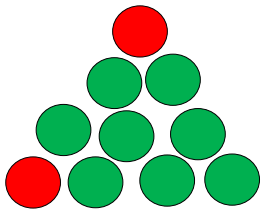
- A = (1+6)/10 = 0.7 System 1
- A = (0+7)/10 = 0.7 System 2
- When classes are highly unbalanced
  - Precision/recall/F1 for the **rare class**
  - e.g. Spam classification (detection)

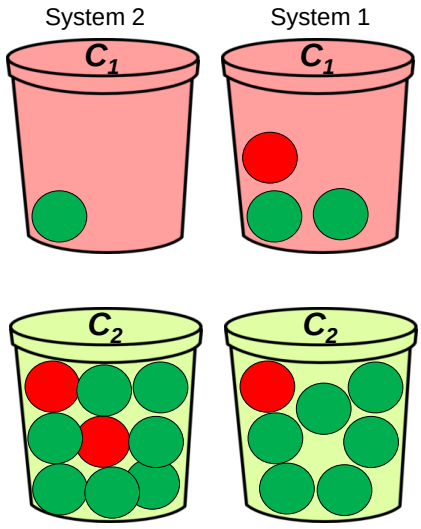




Evaluation: Binary Classification

	System 1	System 2
Precision	1/3 = 0.33	0/1 = 0
Recall	1/2 = 0.5	0/2 = 0
F1	0.4	0



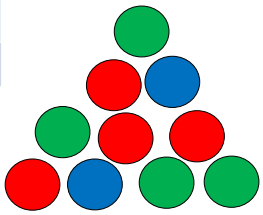


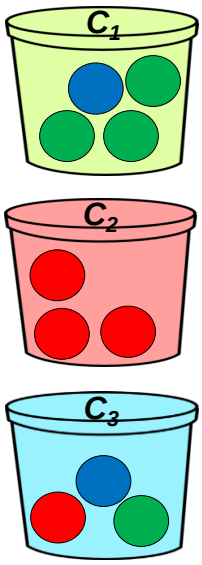
Evaluation: Multi-class

- Accuracy = (3+3+1)/10 = 0.7
- Good measure when
  - Classes are nearly balanced
- Preferred:
  - Precision/recall/F1 for each class

			
P	0.75	1	0.333
R	0.75	0.75	0.5
F1	0.75	0.86	0.4

- **Macro-F1**
  = (0.75+0.86+0.4)/3
  = **0.67**

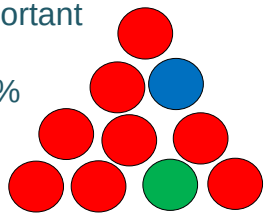


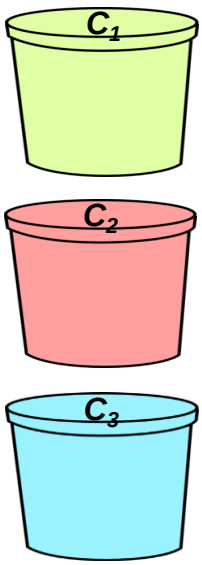


Evaluation: Multi-class

- Majority class baseline
- Accuracy = 0.8
- Macro-F1 = 0.296

- Macro-F1:
  - Should be used in binary classification when two classes are important
  - e.g.: males/females while distribution is 80/20%

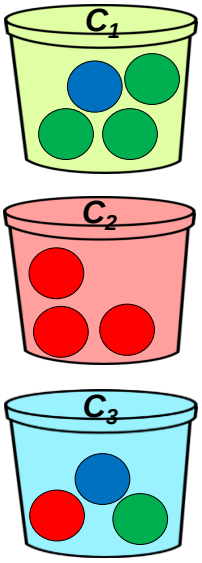




Error Analysis

- **Confusion Matrix**

Actual class \ Predicted class			
	3	0	1
	0	3	1
	1	0	1



- Useful:
  - Find classes that are confused with others
  - Develop better features to solve the problem

Summary

- Text Classification tasks
- Feature extraction/selection/synthesis/weighting
- Learning algorithms
- Cross-validation
- Baselines
- Evaluation measures
  - Accuracy/precision/recall/Macro-F1

Resources

- *Fabrizio Sebastiani*
  Machine Learning in Automated Text Categorization
  ACM Computing Surveys, 2002
  Link: https://arxiv.org/pdf/cs/0110053
- *Yoav Goldberg*
  A Primer on Neural Network Models for Natural Language Processing
  Link: https://arxiv.org/abs/1510.00726

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Text Classification (2)

Instructor:
Björn Ross

17-Nov-2021

1

Lecture Objectives

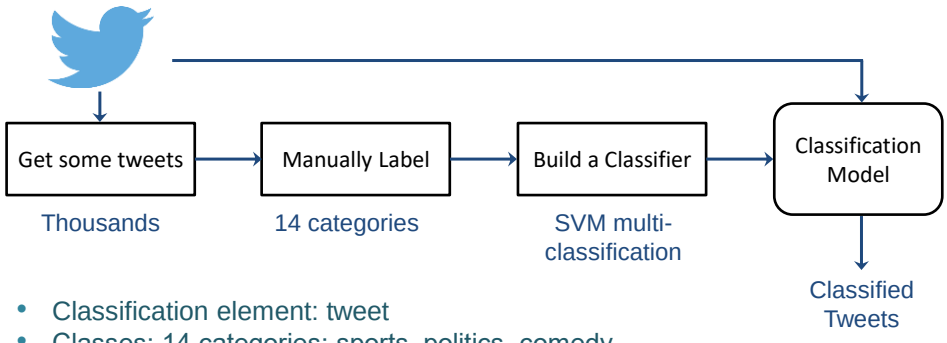
- Implement your first text classifier easy steps
- This is practical lecture
  No equations this time ☺

My first text classifier: Ingredients

- Text elements to be classified
  - Document, paragraph, sentence
- Set of predefined classes (classification task)
  - At least two (binary)
  - Topical, spam, relevance, sentiment, …
- Training set
  - Enough samples of text elements for each class
- Test set (+ possible validation set)
  - Some samples of each class that not used in training
- Features set
  - A set of features extracted from the text to train the classifier
- Classifier
  - The ML module that learns a classification model

My first text classifier: Application

- Classifying tweets into general-purpose categories



- Classification element: tweet
- Classes: 14 categories: sports, politics, comedy, …
- Training/test set: 3129 tweets → 80/20% for train/test
- Features: BOW
- Classifier: SVM multiclass classifier

My first text classifier: Steps

1. Prepare training data
   required: piece of text (tweet) + label to class
2. Extract features
   1. Pre-process text: lowercase, tokenise, remove useless strings
   2. Create a list of all unique terms in the training data. Give each term a unique ID
   3. Convert the text into features, by replacing each term with its corresponding feature ID. Add value to the feature (simplest: value "1" if exists, or count of occurrences)
3. Prepare test file
   Convert test file text into features using the same mapping from the training data. For terms that are not in the features list, it could be neglected, or assigned to an ID representing OOV.
4. Run the learning process on the training data features to create a model
5. Run the classification on the features of the test data and get predictions
6. Evaluate performance

Examples

- Tweet + Label

  Kobe passes Wilt for 4th on all-time scoring list Sports

- Learned features (BOW) from training data
- After converting text to feature vectors

	0	…	2943	2944	2945	2946	…	8330	8331	…	10000
0	0	…	0	1	0	0	…	1	0	…	0
1	0	…	1	0	0	1	…	0	0	…	0
…							…			…	

Feature ID | Corresponding word

2944	kobe
2945	rapping
..	
4525	4th
4526	trevi
..	
8330	passes
8331	ducks
..	
9929	17
9930	wilt
…	

- SVM prediction output

  7 Predicted Class ID

Practical

Possible Improvements

- Feature Extraction
  - Apply stemming & stopping
  - Duplicate hashtags words (#car → #car car)
  - Expand tweet text that has link with the page title of that link
  - Add new set of features to the terms appearing in the profile description of the author of the tweet
    - E.g. tweets terms features: ID range: 1 → 12000
      profile terms features: ID range: 12001 → 20000
    - If a term appeared in the tweet and in the profile description, these are two different features with two different IDs
  - Try non-textual features
    - Tweet length, presence of hashtags, links, emojis …

Possible Improvements

- Feature weighting
  - Using tfidf, BM25 as the feature value instead of binary
- Learning method
  - Test other ML learning methods other than SVM
    - Random forest
    - Decision trees
    - Naive Bayesian
  - Test DNNs with word embeddings
    - Google Collab: you can experiment with using GPUs for free! https://colab.research.google.com/
- Add more training data
  - Think about a way to create more training data

Resources

- Magdy W., H. Sajjad, T. El-Ganainy and F. Sebastiani. (2015) Bridging Social Media via Distant Supervision.
  Springer SNAM 2015 link, arXiv
- Additional reading:
  Nguyen, D. P., Gravel, R., Trieschnigg, R. B., & Meder, T. How old do you think I am? A study of language and age in Twitter.
  ICWSM 2013

THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Learning to Rank

Instructor:
Walid Magdy

01-Dec-2021

1

Lecture Objectives

- Learn about:
  - IR as a classification task
  - Learning to Rank approaches

2

Classical Models vs. ML in IR

- Classical Models:
  - Features (factors): only a few, e.g., TF, IDF, |D|, P(t|corpus) etc.
  - Structure: optimized for the a few particular features
  - Parameter & training
    - Often 1-2; not every factor has a parameter controlling its influence
    - Hand-tuning or data-based; can exhaustive since just 1-2 parameters
  - *tfidf* or BM25 or LMIR? PRF? What n_d, n_t?
- ML in IR
  - Features: can include up to hundreds, thousands, or even more
  - Define the basic structure of a model
  - Quite generic: such as a weighted linear combination of all features
  - Parameters & training
    - Many; control the influence of each feature and their combinations
    - Impossible to tune by hand; Must be data-driven
  - Let the ML decide what is better!

Text Classification in IR

- Text Classification:
  - Classify a document into one of two or more classes
  - Different features could be used, e.g. BOW
- Can we model IR as classification?
  - Classify document to C1: R or C2: NR
  - Challenges?
    - Training data?
    - Features? BOW?
- BOW features cannot work
  - Spam? Viagra, @ed.ac.uk
  - Sentiment? happy, sad
  - Relevant? Trump, hurricane
  - Relevance is a query-dependent class

Getting Classification to IR

- Transforming features
  - Text classification: Input (D) → output (yes/no)
  - Information Filtering: Input (D|Q) → output (yes/no)
- Features set:
  - Independent of absolute words
  - More on relation between doc and query
  - Mostly are numbers (formulas, frequencies, …)
  - Consistent as much as possible among different Q,D pairs
  - e.g.:
    - TFIDF, BM25
    - Query in page title? Heading?
    - Query in anchor text linking pages
    - PageRank of doc
    - Number of times page clicked for the same query

Popular Features

Column in Output	Description	Column in Output	Description
1	TF(Term frequency) of body	24	LMIR.JM of body
2	TF of anchor	25	BM25 of anchor
3	TF of title	26	LMIR.ABS of anchor
4	TF of URL	27	LMIR.DIR of anchor
5	TF of whole document	28	LMIR.JM of anchor
6	IDF(Inverse document frequency) of body	29	BM25 of title
7	IDF of anchor	30	LMIR.ABS of title
8	IDF of title	31	LMIR.DIR of title
9	IDF of URL	32	LMIR.JM of title
10	IDF of whole document	33	BM25 of URL
11	TF*IDF of body	34	LMIR.ABS of URL
12	TF*IDF of anchor	35	LMIR.DIR of URL
13	TF*IDF of title	36	LMIR.JM of URL
14	TF*IDF of URL	37	BM25 of whole document
15	TF*IDF of whole document	38	LMIR.ABS of whole document
16	DL(Document length) of body	39	LMIR.DIR of whole document
17	DL of anchor	40	LMIR.JM of whole document
18	DL of title	41	PageRank
19	DL of URL	42	Inlink number
20	DL of whole document	43	Outlink number
21	BM25 of body	44	Number of slash in URL
22	LMIR.ABS of body	45	Length of URL
23	LMIR.DIR of body	46	Number of child page

Training Data

- Training data: {R,X}
  - X: feature representation of (D,Q) pairs
  - R = {-1,+1} … is D relevant to Q or no
- Samples:
  - Large set of (D,Q) pairs
  - Wide range of Q's (long/short, frequent/rare, …)
  - Wide range of D's for each Q (top/deep ranked, recent/old pages, …)
- Labels:
  - Manually labelled: assessors judge relevance of docs to queries (similar to standard IR)
  - Automatically labelled: click-through data

Classification or Ranking?

- Click-through data
  - User clicks can give indication of relevance
  - What about non-relevance?
  - A list of ranked results: D1 → D2 → D3
    user clicked on D3 and neglected D1 & D2
    what does it mean?
    - D3 is relevant and D1 & D2 are not relevant?
    - Relevance: D3 > D1 & D2?
- It might be better to model the problem as ranking
  - Label→ Ranking preference (e.g. gain={4,3,2,1,0})
  - Learning→ to optimize $Doc_X > Doc_Y$
    not to classify them to R/NR
  - Input: features for set of docs for a given query
    Objective: rank them (sort by relevance)

ML & IR: History

- Considerable interaction between these fields
  - Rocchio algorithm (60s) is a simple learning approach
  - 80s, 90s: learning ranking algorithms based on user feedback
  - 2000s: text categorization
- Limited by amount of training data
- Web query logs have generated new wave of research
  - L2R (LTR): “Learning to Rank”

What is Learning-to-Rank?

- Purpose
  - Learn a function automatically to rank results effectively
- Point-wise approach
  - Classify document to R / NR
- List-wise
  - The function is based on a ranked list of items
  - given two ranked list of the same items, which is better
- Pair-wise
  - The function is based on a pair of item
  - e.g., given two documents, predict partial ranking

Point-wise Approaches

- The function is based on features of a single object
  - e.g., regress the rel. score, classify docs into R and NR
- Very similar to classification
  - Examples of (D,Q) pairs with labels 1 or 0
- Classic retrieval models are also point-wise:
  - Calculate score(Q, D)
  - *If* score(Q,D) > θ → relevant
    else, irrelevant
- Referred to as *information filtering*
  - Standing query + new documents coming
  - Decide weather a new document is R on NR

List-based Approaches

- Given: ranked list A and ranked list B
  Task: decide which is better
- Need a loss function on a list of documents
- Challenge is scale
  - Huge number of potential lists
- Can develop tricks
  - Consider only possible re-rankings of top N retrieved by some fixed method
- Still expensive
  - No clear benefits over pairwise ones (so far)

Pair-wise Approaches

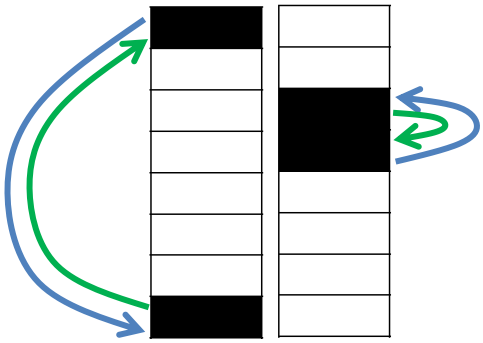
- Trying to classify
  - Which document of two should be ranked at a higher position?
- Optimize based on:
  - Margin between decision hyperplane and instances
  - Errors
  - Weighted based on some hyper-parameter C
  - Evaluation metric
- Example: SVM-rank
  - A generalization of SVM that supports ranking [Herbrichet al. 1999, 2000; Joachims et al. 2002]

Pair-wise Approaches

- The most popular approach
- Learning method: SVM-rank, RankBoost, GBRank, Ranknet, LambdaRank, LambdaMART
- Several issues of ranking SVM
  - Still, it does not directly optimize an evaluation metric
  - But pairwise ranking error often has better correlations with evaluation metrics than the loss/objective functions in point-wise approaches
    - Why: evaluation measures only care about rankings!
    - e.g., ground-truth: rel(D1) = 2, rel(D2) = 1
      - Regression model 1: pred.rel(D1) = 2, pred.rel(D2) = 3
      - Regression model 2: pred.rel(D1) = 1, pred.rel(D2) = 0
      - Model 1 is better than model 2 by criterion of evaluation regression (the prediction error), but model 2 yields a correct ranking of docs

Pair-wise Approaches

- LambdaMART:
  - Misordered pairs are not equally important
  - Depends on how much they contribute to the changes in the target evaluation measure



Pair-wise Approaches

- Optimizing for an evaluation metric
  - The general idea is to weight loss/objective function or gradient with pairwise changes in evaluation measure.
  - e.g., in LambdaMART: lambda gradient
- Can we optimize all measures?
  - Not necessarily
  - For some measures, pairwise change do not only relate to the two documents themselves, but also others …
    - Position-based measures do not have the issues (pairwise change only depends on the two documents)
    - Cascade measures may have issues

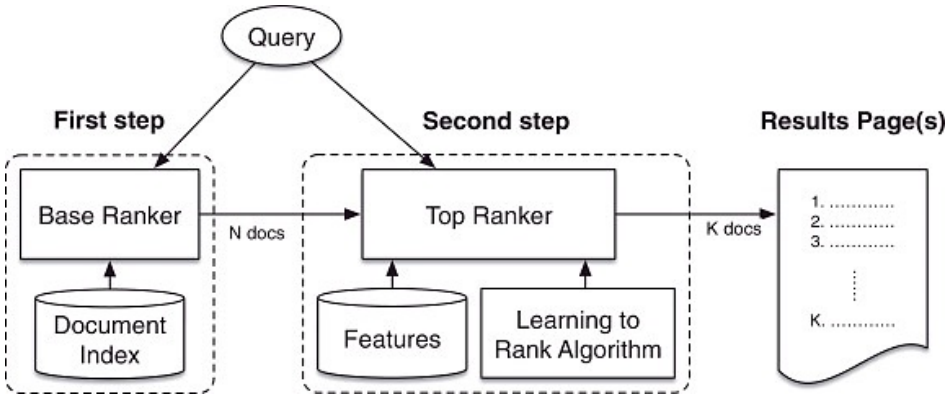
Pair-wise Approaches: Example

- Experiments
  - 1.2k queries, 45.5K documents with 1890 features
  - 800 queries for training, 400 queries for testing

	MAP	P@1	ERR	MRR	NDCG@5
ListNET	*0.2863*	*0.2074*	*0.1661*	*0.3714*	*0.2949*
LambdaMART	**0.4644**	**0.4630**	**0.2654**	**0.6105**	**0.5236**
RankNET	0.3005	0.2222	0.1873	0.3816	0.3386
RankBoost	0.4548	0.4370	0.2463	0.5829	0.4866
RankingSVM	0.3507	0.2370	0.1895	0.4154	0.3585
AdaRank	0.4321	0.4111	0.2307	0.5482	0.4421
pLogistic	0.4519	0.3926	0.2489	0.5535	0.4945
Logistic	0.4348	0.3778	0.2410	0.5526	0.4762

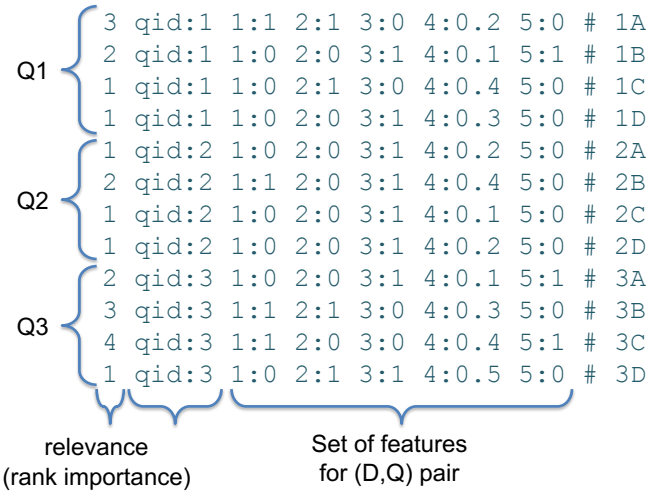
Honglin Wang Slides

L2R in Practice



Capannini, G., et al.
Quality versus efficiency in document scoring with learning-to-rank models.
IP&M 2016.

SVM-rank Example

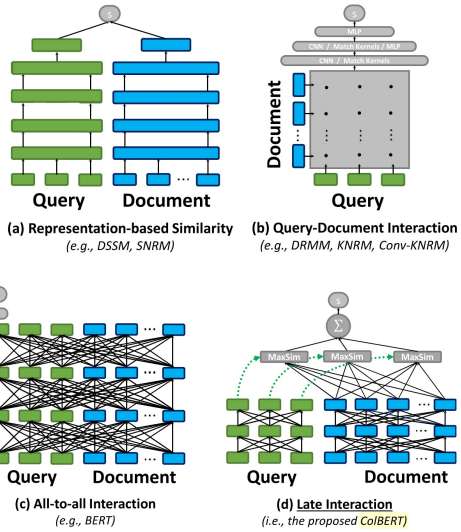


- Q3: 3C>3A, 3C>3B, 3C>3D, 3B>3A, 3B>3D, 3A>3D

Current work in L2R

- Deep learning models are mainly used
- No manual feature extraction is applied
- Try to use word-embeddings to represent queries and docs, then learn the features automatically
- Content-independent models: try to learn the pattern of relations between terms in Q and D
- Content dependent: dependent on the terms

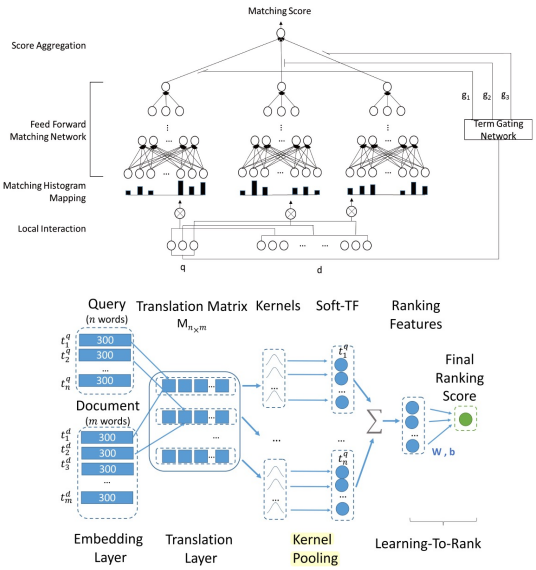
Types of Deep LTR Models



- Early Interaction-based: Learn on the signals from a query-document interaction.
- Late Interaction (Representation) based: Learn independent representations of queries and documents and then consider the interaction between them.
- Early interaction based approaches, e.g. DRMM, are relatively independent of the content (terms themselves) – tend to generalize well.
- Late interaction based approaches, e.g. ColBERT, are usually data hungry approaches – hence likely not to generalize well on standard ad-hoc IR collections.

Ganguly, guest lecture

DRMM & KNRM



- DRMM (up): *Deep Relevance Matching Model*. Uses histograms of word pair similarities (between doc and query) terms as inputs to a feed-forward network.
- The model seeks to utilize inherent patterns in these histograms to distinguish relevance from non-relevance.
- KNRM (down): *Kernel-based Neural Ranking Model*. It does not need to rely on histograms. Instead it applies 1D convolution.

Ganguly, guest lecture

Summary

- IR as a classification task
- Learning to rank (L2R) approaches
  - Point-wise
    - Information Filtering
  - List-wise
  - Pair-wise
    - Ranking SVM
    - LambdaMART
- Current work in L2R depends on deep learning models and word-embedding representations

Resources

- Nallapati, Ramesh.
  Discriminative models for information retrieval.
  SIGIR 2004.
- Burges, C. J. (2010).
  From ranknet to lambdarank to lambdamart: An overview.
  Learning, *11*(23-581), 81.
- Xiong, Chenyan, et al. End-to-end neural ad-hoc ranking with kernel pooling. *SIGIR 2017*
- Guest Lecture by Debasis Ganguly
- SVM^{Rank}: http://svmlight.joachims.org/
- L2R test sets:
  - Microsoft's LETOR project
    http://research.microsoft.com/en-us/um/beijing/projects/letor//default.aspx
  - Microsoft L2R datasets
    http://research.microsoft.com/en-us/projects/mslr/default.aspx