

Course Introduction

[DAT640] Information Retrieval and Text Mining

Ivica Kostric & Krisztian Balog

University of Stavanger

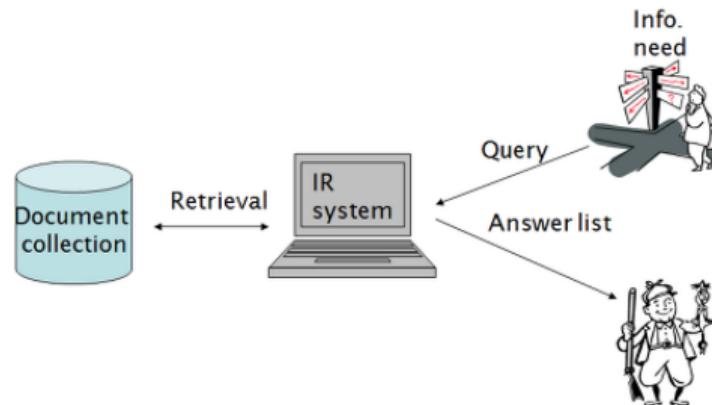
August 22, 2022



CC BY 4.0

About the course

- Techniques and methods for processing, mining, and searching in massive text collections
- **Information retrieval** (search engines): Analysis, organization, storage, and retrieval of information



- **Text mining** (text analytics): Deriving high-quality information from textual data by analyzing trends and patterns

Instructors



Ivica Kostric
Lecturer



Krisztian Balog
Course responsible



Nolwenn Bernard
Teaching assistant



Weronika Lajewska
Teaching assistant

Format

- **Lectures** (weeks 34-43)
 - Mondays as well as Wednesdays for the first three weeks
 - In-person, no recordings/streaming, slides are shared
- **Individual assignments** (weeks 34-43)
 - 10 assignments in total, graded
 - Tuesdays labs are dedicated to working on the assignments and getting help from TAs
- **Group project** (weeks 37-46)
 - Work in groups of 2-3 on a selected project, report will be graded
 - Wednesdays are dedicated to group project work, as well as Mondays and Tuesdays during the last three weeks
 - Lecturer and TAs are available to give feedback (time slots to be booked in advance)

w	Mon	Tue	Wed
34	M1	A0	M2
35	M3	A1.1	M4
36	M5	A1.2	M6
37	M7	A1.3	
38	M8	A2.1	
39	M9	A2.2	
40	M10	A2.3	G1
41	M11	A3.1	
42	M12	A3.2	
43	M13	A3.3	G2
44			
45			
46			G3

Resources

- **Everything is on Canvas**
 - Course schedule and curriculum
 - Lecture slides
 - Exercises and solutions
 - Assignments
 - etc.
- **Bring your own device (laptop)!**
 - Python 3.7+ (ideally, Anaconda distribution)

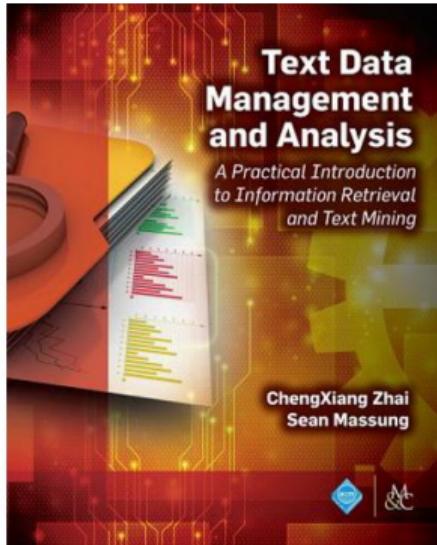
Course content

- Part I: Text representation
 - Text preprocessing and similarity
 - Text classification and clustering
- Part II: Classic IR
 - Search engine architecture and basic retrieval models
 - Advanced retrieval models
 - Retrieval evaluation
- Part III: Entity-oriented search
 - Knowledge bases and entity retrieval
 - Entity linking
 - Semantic search
- Part IV: Neural IR
 - Word embeddings and dense retrieval
 - Transformer-based models
- Part V: Advanced topics
 - Conversational information access

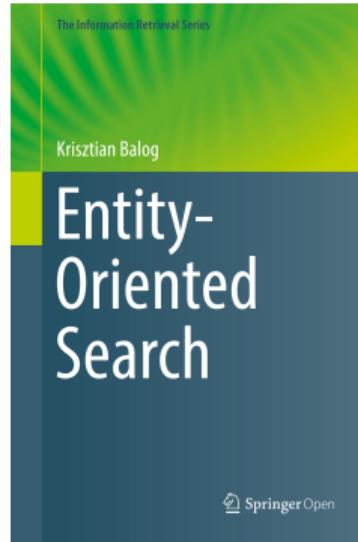
Prerequisites/requirements

- No formal prerequisites, **but** you are expected to know
 - Algorithms and data structures
 - Databases (basic concepts)
 - A bit of statistics
 - Python

Textbooks



**Text Data Management and Analysis:
A Practical Introduction to Information
Retrieval and Text Mining** (Zhai and
Massung), ACM and Morgan & Claypool
Publishers, 2016.



Entity-Oriented Search (Balog),
Springer, 2018.

Grading

- **Project work (40%)**
 - 50% from individual assignments
 - 50% from group project work
 - No re-sit option!
- **Written exam (60%)**
 - Digital exam (Inspera)
 - Open book
 - Mixture of exercises, multiple choice, and essay questions
- Both need to be a pass to get a (passing) grade for the course!

	weight	mark
Project work	40%	A-F
Written exam	60%	A-F

Lectures

- Mixes theory, discussion, and hands-on exercises
- Slides will be made available on Canvas beforehand

Question

When you see this box, it means you need to think about answering the question inside.

Exercise

When you see this box, you'll need to go to the referred exercise on Canvas

Individual assignments

- A new assignment is announced each week during the lecture period, with a deadline ≈ 10 days after
- Points vary based on difficulty
- Single delivery (no resubmissions, corrections, etc.)
- Deadlines are strict, no extensions, no exceptions!
- Assignments account for **50% of the project work final grade**
- Tuesday labs are dedicated to working on assignments—this is the time and place to get help

Assignments workflow

- For each assignment we provide
 - a task description
 - a skeleton code (Python file) that needs to be completed
 - a set of public tests you can use to verify your solution
- Assignments are graded automatically, using a combination of public and hidden tests
 - **public tests** are released with the assignment; if your solution passes these, it is likely correct
 - **hidden tests** will be used to test your solution after submission; these typically contain larger inputs/datasets, corner cases, or other inputs in order to test that you fully understood the methods and/or followed the instructions
- Upload completed Python file to Canvas

Group project

- Work in groups of 2-3
- There will be a pool of options to select a project from
- The task will be to tackle a problem, perform experiments, and write a report about the findings
- Groups can get weekly feedback (from week 38)
 - There is a booking system in place for you to reserve 15 minutes slot for updates and questions related to the project
- Evaluation is based on two intermediate and one final project submission
- Accounts for **50% of the project work final grade**

Contact

- For all course-related matters, the **primary contact email** is **ivica.kostric@uis.no**
- Tuesday labs are for working on the assignments. This is **the** time to get help!
 - Contacts for assignment-related issues: **nolwenn.m.bernard@uis.no** and **weronika.lajewska@uis.no**.
- If you need to talk to the lecturer, make an appointment via email. No drop-ins unannounced!

Some simple rules

- We're back to regular teaching, which means physical lectures and in-person communication. (We're slow on email.)
- Deadlines are strict and immutable, no extensions (don't even ask)!
 - Application for exceptions has to be submitted 72 hours prior the deadline **in writing** with necessary **proof attached** (medical certificate, etc.)
- Discussions while working on assignments is OK. Copying someone else's solution is considered cheating and will get both of you into **serious** trouble (cases will be reported to the Exam Office)

- **Is it obligatory to attend the lectures?**

No. However, it is highly recommended in order to get the full learning experience (especially that there is no recording/streaming). Also, note that everything that is covered in the lectures may be asked back at the exam.

- **Is it obligatory to attend the labs?**

No. However, this is the time and place to work and get help on the assignments.

Question

Do you have any questions?

Exercise

Exercise

E0-1 Python basics

Text Preprocessing and Similarity

[DAT640] Information Retrieval and Text Mining

Ivica Kostric & Krisztian Balog
University of Stavanger

August 22, 2022



CC BY 4.0

In this module

1. Representing text
2. Text similarity
3. Text preprocessing

Representing text

Question

What makes working with text challenging?

Representing text

- How to represent text for machine understanding?
 - To measure the similarity between two texts (later in this lecture)
 - To decide whether some document belongs to a category (text classification)
 - To decide which groups of documents belong together (text clustering)
 - To decide if a document is relevant to a search query (search)
 - ...
- We need a mathematical representation for words (referred to as *terms*) and for sequences of words (referred to as *documents*)

Representing words

- Discrete (sparse) representation
 - Each word is given a unique **term ID** (integer)
 - The **vocabulary** V contains a mapping between terms and their IDs
 - \Rightarrow This is the representation we use for now

ID	term
1	discrete
2	sparse
3	document
4	text
5	word
6	this
	...

Vocabulary

Representing words (2)

- Continuous (dense) representation
 - Each word is represented as a real-valued vector in a (relatively) small dimensional latent space, i.e., an **embedding vector**

$$\vec{w} = \begin{bmatrix} 0.15 & 0.07 & 0.83 & 0.46 & \dots & 0.02 \end{bmatrix}$$

- Similar/related words are close to each other in the embedding space
 - (To be covered in the second half of the course)

Representing documents

- Terms are represented by their respective IDs in the vocabulary
 - There might be out of vocabulary (OOV) words, i.e., terms not seen during the construction of the vocabulary
 - Those are substituted with a special OOV token (if term positions matter) or simply ignored
- A document might be represented as a **sequence of words**
- A document d can also be represented as a **term vector**
 - Each element of the vector corresponds to a term in the vocabulary
 - The value might represent
 - the presence/absence of a word (0/1)
 - the frequency of the word (int)
 - the word's importance (real)
 - This is a **sparse** representation, as most values will be zeros
 - Word position is ignored, therefore it is called the **bag-of-words** representation

ID	term
1	discrete
2	sparse
3	document
4	text
5	word
6	this
	...

Vocabulary

$d = \text{"this is sparse text"}$

$d = [t_6, OOV, t_2, t_4]$

$\vec{d} = \langle 0, 1, 0, 1, 0, 1, \dots \rangle$

Representing a collection of documents

- Document-term matrix, where
 - Rows correspond to documents (n)
 - Columns correspond to terms in the vocabulary (m)
- The matrix is huge, but most of the values are zeros; stored as a sparse matrix

	t_1	t_2	t_3	\dots	t_m
d_1	1	0	2		0
d_2	0	1	0		2
d_3	0	0	1		0
\dots					
d_n	0	1	0		0

Document-term matrix

Text similarity

Text similarity

- Core ingredient in many text mining and information retrieval problems
- Need to express the similarity between two pieces of text (referred to as *documents*, for simplicity)
- The choice of the similarity measure is closely tied with how documents are represented

Jaccard similarity

- **Jaccard similarity:** only the presence/absence of terms in documents is considered, with no regard to magnitude
- Defined as the ratio of shared terms and total terms in two documents:

$$sim_{\text{Jaccard}}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} ,$$

- where X and Y represent the terms that appear in documents d_1 and d_2 , respectively

Jaccard similarity

- Jaccard similarity for term vector-based representations:

$$sim_{\text{Jaccard}}(\mathbf{x}, \mathbf{y}) = \frac{\sum_i \mathbb{1}(x_i) \times \mathbb{1}(y_i)}{\sum_i \mathbb{1}(x_i + y_i)} ,$$

- here $\mathbb{1}(x)$ is an indicator function (1 if $x > 0$ and 0 otherwise).

Example

	term 1	term 2	term 3	term 4	term 5
doc x	1	0	1	0	3
doc y	0	2	4	0	1

Table: Document-term vectors with term frequencies.

$$\mathbf{x} = \langle 1, 0, 1, 0, 3 \rangle \quad \mathbf{y} = \langle 0, 2, 4, 0, 1 \rangle$$

$$sim_{\text{Jaccard}}(\mathbf{x}, \mathbf{y}) = \frac{0 + 0 + 1 + 0 + 1}{1 + 1 + 1 + 0 + 1} = \frac{2}{4}$$

Cosine similarity

- **Cosine similarity:** the cosine of the angle between the two document vectors plotted in their high-dimensional space; the larger the angle, the more dissimilar the documents are:

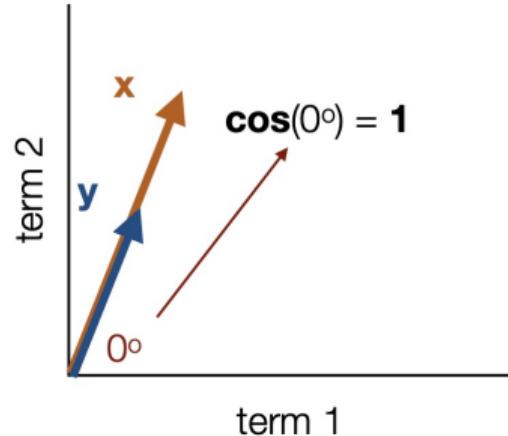
$$sim_{\cos}(x, y) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}} ,$$

- where \mathbf{x} and \mathbf{y} are the term vectors corresponding to documents d_1 and d_2 , respectively
- Term weights (x_i and y_i) may be raw term counts or TF-IDF-weighted frequencies

Cosine similarity - Geometric interpretation

	term 1	term 2
doc x	1	2
doc y	2	4

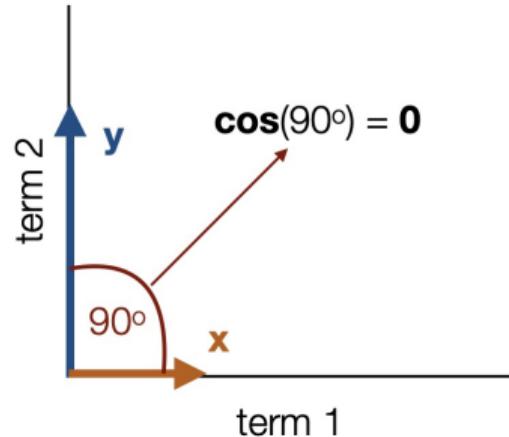
$$sim_{\cos}(x, y) = 1$$



Cosine similarity - Geometric interpretation

	term 1	term 2
doc <i>x</i>	1	0
doc <i>y</i>	0	2

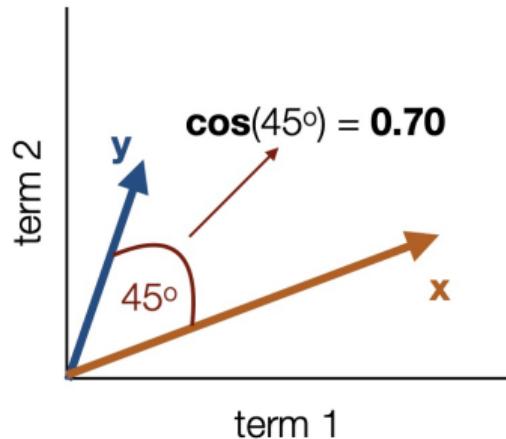
$$\text{sim}_{\cos}(x, y) = 0$$



Cosine similarity - Geometric interpretation

	term 1	term 2
doc x	4	2
doc y	1	3

$$sim_{\cos}(x, y) = 0.7$$



Cosine similarity

Example

	term 1	term 2	term 3	term 4	term 5
doc x	1	0	1	0	3
doc y	0	2	4	0	1

Table: Document-term vectors with term frequencies.

$$\mathbf{x} = \langle 1, 0, 1, 0, 3 \rangle \quad \mathbf{y} = \langle 0, 2, 4, 0, 1 \rangle$$

$$\begin{aligned} sim_{\cos}(x, y) &= \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}|| \cdot ||\mathbf{y}||} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}} \\ &= \frac{1 \times 0 + 0 \times 2 + 1 \times 4 + 0 \times 0 + 3 \times 1}{\sqrt{1^2 + 0^2 + 1^2 + 0^2 + 3^2} \sqrt{0^2 + 2^2 + 4^2 + 0^2 + 1^2}} = \frac{7}{\sqrt{11} \sqrt{21}} \end{aligned}$$

Exercise

E1-1 Term vector similarity

Text preprocessing

Text preprocessing



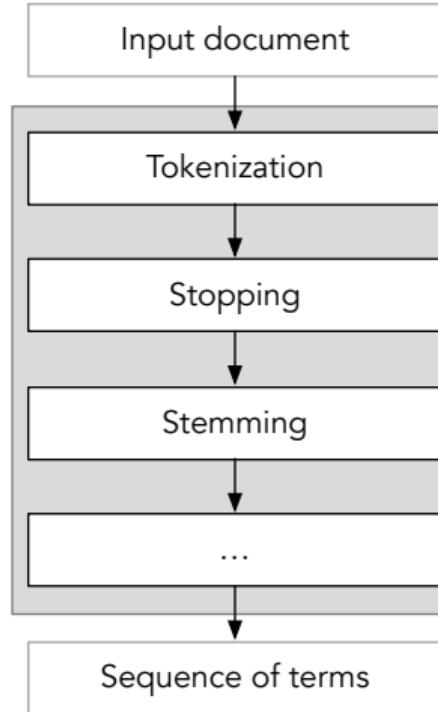
$$d_1 : [t_1, t_3, t_3]$$

$$d_2 : [t_2, t_4, t_5, t_6, t_2]$$

$$d_3 : [t_3, t_2, t_3]$$

...

Text preprocessing pipeline



Tokenization

- Parsing a string into individual words (tokens)
- Splitting is usually done along white spaces, punctuation marks, or other types of content delimiters (e.g., HTML markup)
- Sounds easy, but can be surprisingly complex, even for English
 - Even worse for many other languages

Question

What could be the issues with tokenization along whitespace and punctuation marks?

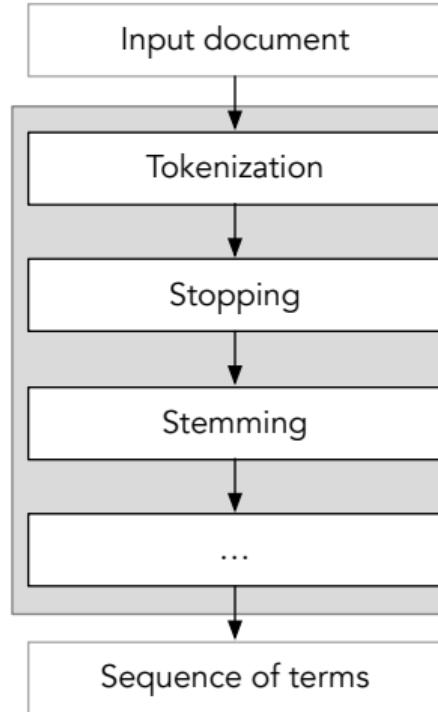
Tokenization issues

- Apostrophes can be a part of a word, a part of a possessive, or just a mistake
 - *rosie o'donnell, can't, 80's, 1890's, men's straw hats, master's degree, ...*
- Capitalized words can have different meaning from lower case words
 - *Bush, Apple, ...*
- Special characters are an important part of tags, URLs, email addresses, etc.
 - *C++, C#, ...*
- Numbers can be important, including decimals
 - *nokia 3250, top 10 courses, united 93, quicktime 6.5 pro, 92.3 the beat, 288358, ...*
- Periods can occur in numbers, abbreviations, URLs, ends of sentences, and other situations
 - *I.B.M., Ph.D., www.uis.no, F.E.A.R., ...*

Common practice

- Process documents in two stages
 - First pass is focused on identifying markup or tags
 - Second pass is done on the appropriate parts of the document structure
- Treat hyphens, apostrophes, periods, etc. like spaces
- Ignore capitalization
- Index even single characters
 - *o'connor* ⇒ *o connor*

Text preprocessing pipeline



Stopword removal

- Function words that have little meaning apart from other words: *the, a, an, that, those, ..*
- These are considered **stopwords** and are removed
- A stopwords list can be constructed by taking the top- k (e.g., 50) most common words in a collection
 - May be customized for certain domains or applications

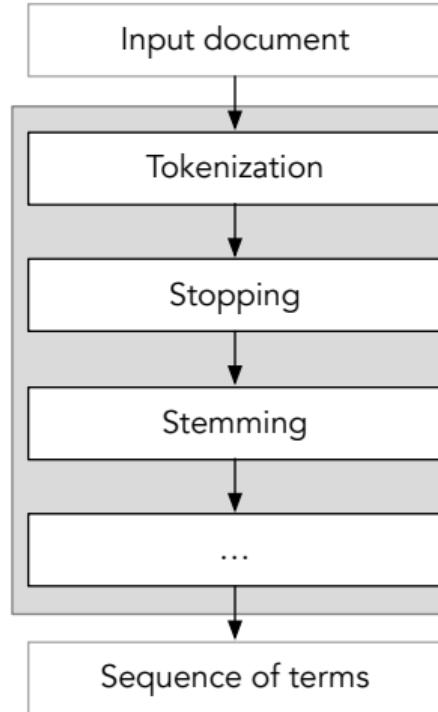
Example (minimal stopword list)

a	as	by	into	not	such	then	this	with
an	at	for	is	of	that	there	to	
and	be	it	it	on	the	these	was	
are	but	in	no	or	their	they	will	

Question

What about a text like “to be or not to be”?

Text preprocessing pipeline



Stemming

- Reduce the different forms of a word that occur to a common stem
 - Inflectional (plurals, tenses)
 - Derivational (making verbs nouns etc.)
- In most cases, these have the same or very similar meanings
- Basic types of stemmers
 - Algorithmic
 - Dictionary-based
 - Hybrid algorithmic-dictionary

Suffix-s stemmer

- Assumes that any word ending with an 's' is plural
 - *cakes* ⇒ *cake*, *dogs* ⇒ *dog*
- Cannot detect many plural relationships (false negative)
 - *centuries* ⇒ *century*
- In rare cases it detects a relationship where it does not exist (false positive)
 - *is* ⇒ *i*

Porter stemmer

- Most popular algorithmic stemmer
- Consists of 5 steps, each step containing a set of rules for removing suffixes
- Produces stems not words
- Makes a number of errors and difficult to modify

Example step (1 of 5)

Step 1a:

- Replace **sses** by **ss** (e.g., stresses → stress).
- Delete **s** if the preceding word part contains a vowel not immediately before the **s** (e.g., gaps → gap but gas → gas).
- Replace **ied** or **ies** by **i** if preceded by more than one letter, otherwise by **ie** (e.g., ties → tie, cries → cri).
- If suffix is **us** or **ss** do nothing (e.g., stress → stress).

Step 1b:

- Replace **eed**, **eedly** by **ee** if it is in the part of the word after the first non-vowel following a vowel (e.g., agreed → agree, feed → feed).
- Delete **ed**, **edly**, **ing**, **ingly** if the preceding word part contains a vowel, and then if the word ends in **at**, **bl**, or **iz** add **e** (e.g., fished → fish, pirating → pirate), or if the word ends with a double letter that is not **ll**, **ss** or **zz**, remove the last letter (e.g., falling → fall, dripping → drip), or if the word is short, add **e** (e.g., hoping → hope).
- Whew!

Porter stemmer examples

False positives

(should not have the same stem)

organization/organ

generalization/generic

numerical/numerous

policy/police

university/universe

addition/additive

negligible/negligent

execute/executive

past/paste

False negatives

(should have the same stem)

european/europe

cylinder/cylindrical

matrices/matrix

urgency/urgent

create/creation

analysis/analyses

useful/usefully

noise/noisy

decompose/decomposition

Krovetz stemmer

- Hybrid algorithmic-dictionary
- Word checked in dictionary
 - If present, either left alone or replaced with exception stems
 - If not present, word is checked for suffixes that could be removed
- After removal, dictionary is checked again
- Produces words not stems

Stemmer comparison

Original text

Document will describe **marketing** strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for **agrochemicals**, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales

Porter stemmer

market strategi carr compan agricultur chemic report predict market share chemic report
market statist **agrochem** pesticid herbicid fungicid insecticid fertil predict sale stimul
demand price cut volum sale

Krovetz stemmer

marketing strategy carry company agriculture chemical report prediction market share
chemical report market statistic **agrochemic** pesticide herbicide fungicide insecticide
fertilizer predict sale stimulate demand price cut volume sale

Effect of stemming

- Generally a small (but significant) effectiveness improvement for English
- Can be crucial for some languages (e.g., Arabic, Russian)

Exercise

E1-2 Text preprocessing

Summary

- Representing text (terms, documents, collection of documents)
- Measuring text similarity (Jaccard and cosine)
- Text preprocessing (tokenization, stopwords removal, stemming)

Reading

- Text Data Management and Analysis (Zhai&Massung)
 - Section 8.1

Assignment

- This is just a test
- No point for this; meant to help everyone familiarize themselves with the submission system

Text Classification and Clustering

[DAT640] Information Retrieval and Text Mining

Ivica Kostric & Krisztian Balog
University of Stavanger

August 24, 2022



CC BY 4.0

In this module

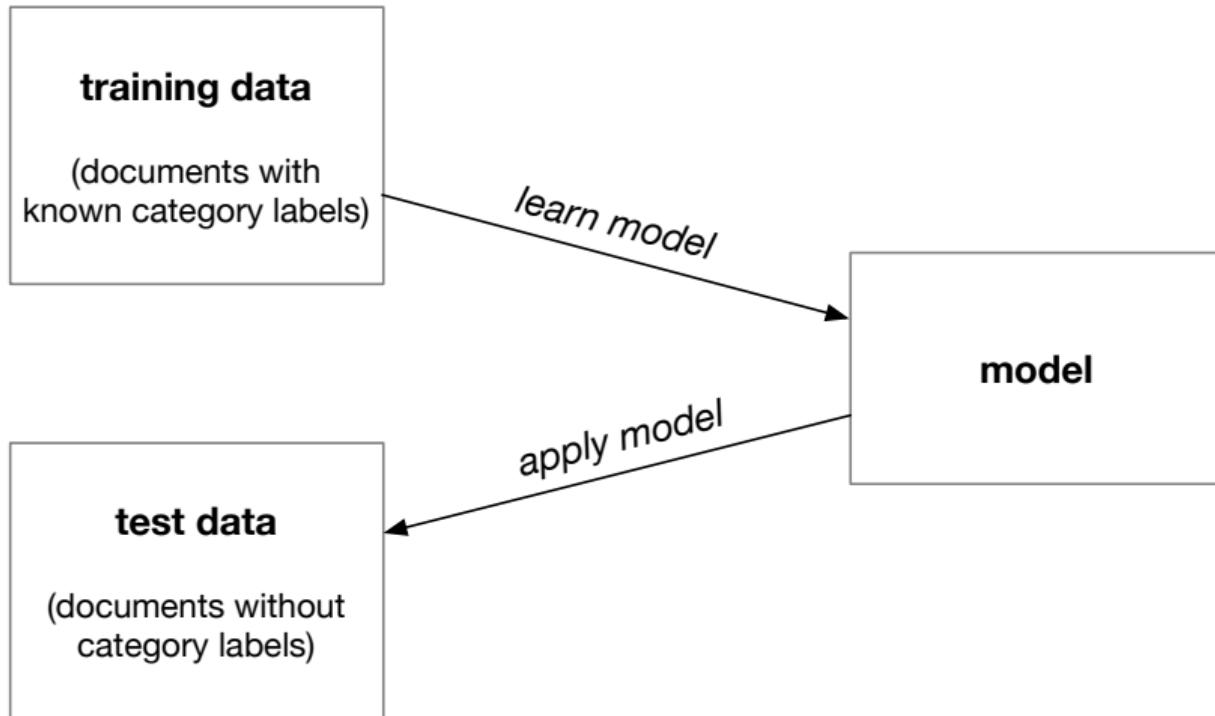
1. Text classification
2. Naive Bayes text classifier
3. Text classification evaluation
4. Text clustering

Text classification

Text classification

- **Classification** is the problem of assigning objects to one of several predefined categories
 - One of the fundamental problems in machine learning, where it is performed the basis of a training dataset (instances whose category membership is known)
- In **text classification** (or **text categorization**) the objects are text documents
- Binary classification (two classes, 0/1 or -/+)
 - E.g., deciding whether an email is spam or not
- Multiclass classification (n classes)
 - E.g., Categorizing news stories into topics (finance, weather, politics, sports, etc.)

General approach



Formally

- Given a training sample (\mathbf{X}, y) , where \mathbf{X} is a set of documents with corresponding labels y , from a set \mathbf{Y} of possible labels, the task is to learn a function $f(\cdot)$ that can predict the class $y' = f(x)$ for an unseen document x .

Families of approaches

- **Feature-based approaches** (“traditional” machine learning)
- Neural approaches (“deep learning”)

Question

What could be used as features in text classification?

Features for text classification

- Use words as features (**bag-of-words**)
 - Words will be referred to as **terms**
- Values can be, e.g., binary (term presence/absence) or integers (term counts)
- Documents are represented by their **term vector**
- **Document-term matrix** is huge, but most of the values are zeros; stored as a sparse matrix

	t_1	t_2	t_3	\dots	t_m
d_1	1	0	2		0
d_2	0	1	0		2
d_3	0	0	1		0
\dots					
d_n	0	1	0		0

Document-term matrix

Question

Is it possible to use other, non-term-frequency-based features?

Additional features for text classification

- Descriptive statistics (avg. sentence length, length of various document fields, like title, abstract, body,...)
- Document source
- Document quality indicators (e.g., readability level)
- Presence of images/attachments/JavaScript/...
- Publication date
- Language
- ...

Naive Bayes text classifier

Naive Bayes

- Example of a generative classifier
- Estimating the probability of document x belonging to class y

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

- $P(x|y)$ is the class-conditional probability
- $P(y)$ is the prior probability
- $P(x)$ is the evidence (note: it's the same for all classes)

Naive Bayes classifier

- Estimating the class-conditional probability $P(y|\mathbf{x})$
 - \mathbf{x} is a vector of term frequencies $\{x_1, \dots, x_n\}$

$$P(\mathbf{x}|y) = P(x_1, \dots, x_n|y)$$

- “Naive” assumption: features (terms) are independent:

$$P(\mathbf{x}|y) = \prod_{i=1}^n P(x_i|y)$$

- Putting our choices together, the probability that \mathbf{x} belongs to class y is estimated using:

$$P(y|\mathbf{x}) \propto P(y) \prod_{i=1}^n P(x_i|y)$$

Estimating prior class probabilities

- $P(y)$ is the probability of each class label
- It is essential when class labels are imbalanced

Estimating feature distribution

- How to estimate $P(x_i|y)$?
- Maximum likelihood estimation: count the number of times a term occurs in a class divided by its total number of occurrences

$$P(x_i|y) = \frac{c_{i,y}}{c_i}$$

- $c_{i,y}$ is the number of times term x_i appears in class y
- c_i is the total number of times term x_i appears in the collection
- But what happens if $c_{i,y}$ is zero?!

Smoothing

- Ensure that $P(x_i|y)$ is never zero
- Simplest solution:¹ Laplace (“add one”) smoothing

$$P(x_i|y) = \frac{c_{i,y} + 1}{c_i + m}$$

- m is the number of classes

¹More advanced smoothing methods will follow later for Language Modeling

Practical considerations

- In practice, probabilities are small, and multiplying them may result in numerical underflows
- Instead, we perform the computations in the log domain

$$\log P(y|x) \propto \log P(y) + \sum_{i=1}^n \log P(x_i|y)$$

Exercise

E2-1 Naive Bayes

Text classification evaluation

Evaluation

- Measuring the performance of a classifier
 - Comparing the predicted label y' against the true label y for each document in some set dataset
- Based on the number of records (documents) correctly and incorrectly predicted by the model
- Counts are tabulated in a table called the **confusion matrix**
- Compute various **performance measures** based on this matrix

Text classification evaluation

- Evaluating binary classification
- Evaluating multiclass classification
- Model development

Confusion matrix

		Predicted class	
		negative	positive
Actual class	negative	true negatives (TN)	false positives (FP)
	positive	false negatives (FN)	true positives (TP)

- False positives = Type I error (“raising a false alarm”)
- False negatives = Type II error (“failing to raise an alarm”)

Question

Which of Type I and Type II error is worse?

Type I vs. Type II errors²

Type I Error



Type II Error



²Source: <https://www.analyticsindiamag.com/understanding-type-i-and-type-ii-errors/>

Example

Id	Actual	Predicted
1	+	-
2	+	+
3	-	-
4	+	+
5	+	-
6	+	+
7	-	-
8	-	+
9	+	-
10	+	-

		predicted	
		-	+
actual	-	-	
	+		

Example

Id	Actual	Predicted
1	+	-
2	+	+
3	-	-
4	+	+
5	+	-
6	+	+
7	-	-
8	-	+
9	+	-
10	+	-

		predicted	
		-	+
actual	-	-	2
	+		

Example

Id	Actual	Predicted
1	+	-
2	+	+
3	-	-
4	+	+
5	+	-
6	+	+
7	-	-
8	-	+
9	+	-
10	+	-

		predicted	
		-	+
actual	-	2	1
	+		

Example

Id	Actual	Predicted
1	+	-
2	+	+
3	-	-
4	+	+
5	+	-
6	+	+
7	-	-
8	-	+
9	+	-
10	+	-

		predicted	
		-	+
actual	-	2	1
	+	4	3

Evaluation measures

- Summarizing performance in a single number

- **Accuracy**

Fraction of correctly classified items out of all items

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

- **Error rate**

Fraction of incorrectly classified items out of all items

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

		predicted	
		-	+
actual	-	TN	FP
	+	FN	TP

Evaluation measures (2)

- **Precision**

Fraction of items correctly identified as positive out of the total items identified as positive

$$P = \frac{TP}{TP + FP}$$

- **Recall** (also called Sensitivity or True Positive Rate)

Fraction of items correctly identified as positive out of the total actual positives

$$R = \frac{TP}{TP + FN}$$

		predicted	
		-	+
actual	-	TN	FP
	+	FN	TP

Evaluation measures (3)

- **F1-score**

The harmonic mean of precision and recall

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

		predicted	
		-	+
actual	-	TN	FP
	+	FN	TP

Evaluation measures (4)

- **False Positive Rate (Type I Error)**

Fraction of items wrongly identified as positive out of the total actual negatives

$$FPR = \frac{FP}{FP + TN}$$

- **False Negative Rate (Type II Error)**

Fraction of items wrongly identified as negative out of the total actual positives

$$FNR = \frac{FN}{FN + TP}$$

		predicted	
		-	+
actual	-	TN	FP
	+	FN	TP

Example

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} = \frac{5}{10} = 0.5$$

		predicted	
		-	+
actual	-	TN=2	FP=1
	+	FN=4	TP=3

$$P = \frac{TP}{TP + FP} = \frac{3}{4} = 0.75$$

$$R = \frac{TP}{TP + FN} = \frac{3}{7} = 0.429$$

$$F1 = \frac{2 \cdot P \cdot R}{P + R} = \frac{2 \cdot 3/4 \cdot 3/7}{3/4 + 3/7} = 0.545$$

Text classification evaluation

- Evaluating binary classification
- Evaluating multiclass classification
- Model development

Multiclass classification

- Imagine that you need to automatically sort news stories according to their topical categories

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Table: Categories in the 20-Newsgroups dataset

Multiclass classification

- Many classification algorithms are originally designed for binary classification
- Two main strategies for applying binary classification approaches to the multiclass case
 - One-against-rest
 - One-against-one
- Both apply a voting scheme to combine predictions
 - A tie-breaking procedure is needed (not detailed here)

One-against-rest

- Assume there are k possible target classes (y_1, \dots, y_k)
- Train a classifier for each target class y_i ($i \in [1..k]$)
 - Instances that belong to y_i are positive examples
 - All other instances y_j , $j \neq i$ are negative examples
- Combining predictions
 - If an instance is classified positive, the positive class gets a vote
 - If an instance is classified negative, all classes except for the positive class receive a vote

Example

- 4 classes (y_1, y_2, y_3, y_4)
- Classifying a given test instance (dots indicate the votes cast):

y_1	+	•	y_1	-	•	y_1	-	•	y_1	-	•
y_2	-		y_2	+		y_2	-		y_2	-	
y_3	-		y_3	-	•	y_3	+		y_3	-	
y_4	-		y_4	-	•	y_4	-	•	y_4	+	•
Pred.	+		Pred.	-		Pred.	-		Pred.	-	

- Sum votes received: $(y_1, \bullet\bullet\bullet)$, $(y_2, \bullet\bullet)$, $(y_3, \bullet\bullet)$, $(y_4, \bullet\bullet)$

One-against-one

- Assume there are k possible target classes (y_1, \dots, y_k)
- Construct a binary classifier for each pair of classes (y_i, y_j)
 - $\frac{k \cdot (k-1)}{2}$ binary classifiers in total
- Combining predictions
 - The predicted class receives a vote in each pairwise comparison

Example

- 4 classes (y_1, y_2, y_3, y_4)
- Classifying a given test instance (dots indicate the votes cast):

y_1	+
y_2	-
Pred.	+

•

y_1	+
y_3	-
Pred.	+

•

y_1	+
y_4	-
Pred.	-

•

y_2	+
y_3	-
Pred.	+

•

y_2	+
y_4	-
Pred.	-

•

y_3	+
y_4	-
Pred.	+

•

- Sum votes received: $(y_1, \bullet\bullet)$, (y_2, \bullet) , (y_3, \bullet) , $(y_4, \bullet\bullet)$

Question

How to evaluate multiclass classification?

Which of the evaluation measures from binary classification can be applied?

Evaluating multiclass classification

- Accuracy can still be computed as

$$ACC = \frac{\text{\#correctly classified instances}}{\text{\#total number of instances}}$$

- For other metrics
 - View it as a set of k binary classification problems (k is the number of classes)
 - Create confusion matrix for each class by evaluating “one against the rest”
 - Average over all classes

Confusion matrix

		Predicted				
		1	2	3	...	k
Actual	1	24	0	2		0
	2	0	10	1		1
	3	1	0	9		0
	...					
	k	2	0	1		30

Binary confusion matrices, one-against-rest

The diagram illustrates the decomposition of a k -class confusion matrix into k binary confusion matrices using one-against-rest logic.

Original Confusion Matrix:

		Predicted				
		1	2	3	...	k
Actual	1	24	0	2		0
	2	0	10	1		1
	3	1	0	9		0
	...					
	k	2	0	1		30

Decomposition into Binary Confusion Matrices:

For each class i , we create a binary confusion matrix where the rows represent the actual class and the columns represent the predicted class (1 if predicted i , $\neg 1$ if predicted $\neq i$).

		Predicted	
		1	$\neg 1$
Act.	1	TP=24	FN=3
	$\neg 1$	FP=2	TN=52

		Predicted	
		2	$\neg 2$
Act.	2	TP=10	FN=2
	$\neg 2$	FP=0	TN=69

... (implied for other classes)

For the sake of this illustration, we assume that the cells which are not shown are all zeros.

Averaging over classes

- Averaging can be performed on the instance level or on the class level
- **Micro-averaging** aggregates the results of individual instances across all classes
 - All instances are treated equal
- **Macro-averaging** computes the measure independently for each class and then take the average
 - All classes are treated equal

Micro-averaging

- Precision

$$P_\mu = \frac{\sum_{i=1}^k TP_i}{\sum_{i=1}^k (TP_i + FP_i)}$$

- Recall

$$R_\mu = \frac{\sum_{i=1}^k TP_i}{\sum_{i=1}^k (TP_i + FN_i)}$$

- F1-score

$$F1_\mu = \frac{2 \cdot P_\mu \cdot R_\mu}{P_\mu + R_\mu}$$

		predicted	
		i	$\neg i$
actual	i	TP_i	FN_i
	$\neg i$	FP_i	TN_i

Macro-averaging

- **Precision**

$$P_M = \frac{\sum_{i=1}^k \frac{TP_i}{TP_i+FP_i}}{k}$$

- **Recall**

$$R_M = \frac{\sum_{i=1}^k \frac{TP_i}{TP_i+FN_i}}{k}$$

- **F1-score**

$$F1_M = \frac{\sum_{i=1}^k \frac{2 \cdot P_i \cdot R_i}{P_i + R_i}}{k}$$

- where P_i and R_i are Precision and Recall, respectively, for class i

		predicted	
		i	$\neg i$
actual	i	TP_i	FN_i
	$\neg i$	FP_i	TN_i

Text classification evaluation

- Evaluating binary classification
- Evaluating multiclass classification
- Model development

Question

How can we evaluate the performance of the model during development?

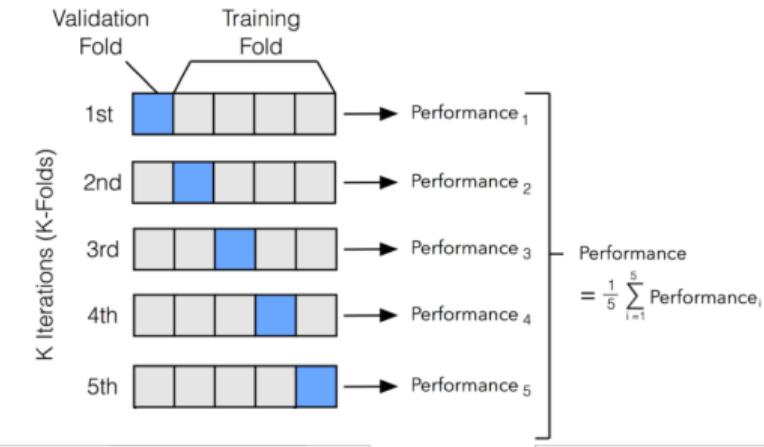
Using a validation set

- Idea: hold out part of the training data for testing into a **validation set**
- **Single train/validation split**
 - Split the training data into $X\%$ training split and $100 - X\%$ validation split (an 80/20 split is common)

Using a validation set³

- ***k*-fold cross-validation**

- Partition the training data randomly into k folds
- Use $k - 1$ folds for training and test on the k th fold; repeat k times (each fold is used for testing exactly once)
- k is typically 5 or 10
- Extreme: k is the number of data points, to maximize the number of training material available (called “leave-one-out” evaluation)



³Image source:

http://ethen8181.github.io/machine-learning/model_selection/model_selection.html

Exercise

E2-2 Cross-validation

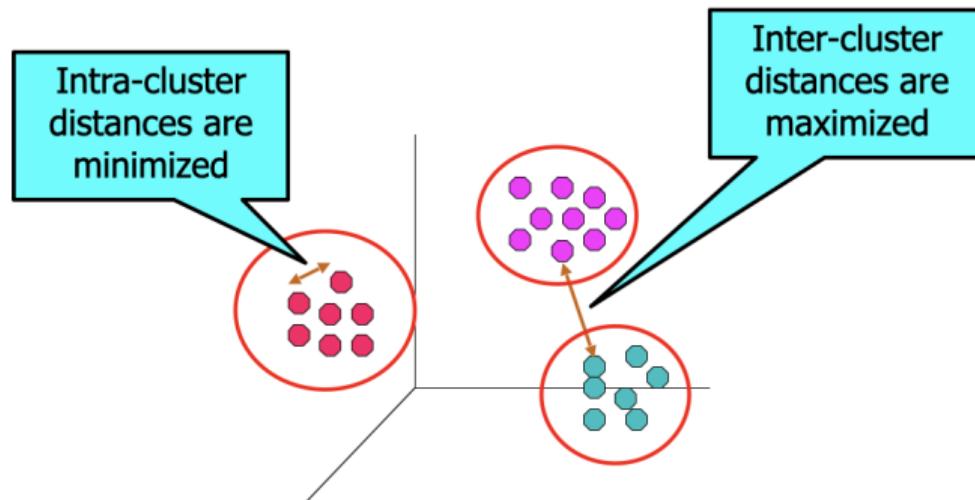
Text clustering

Clustering

- **Clustering** is concerned with the task of grouping similar objects together
 - Objects can be documents, sentences, words, users, etc.
- It is a general data mining technique for exploring large datasets
 - Clustering can reveal natural semantic structures
 - Can also help to navigate data, discover redundant content, etc.
- Clustering is regarded as an unsupervised learning problem

Clustering

Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



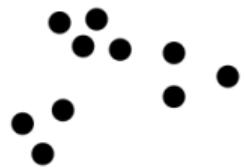
Question

How many clusters should be formed?

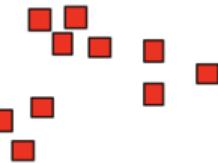


Original data

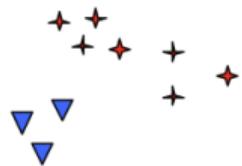
The notion of a cluster can be ambiguous



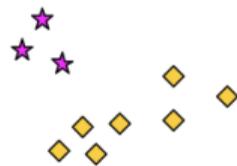
Original data



Two Clusters



Four Clusters



Six Clusters

Types of clustering

- Partitional vs. hierarchical
 - Partitional: non-overlapping clusters such that each data object is in exactly one cluster
 - Hierarchical: a set of nested clusters organized as a hierarchical tree
- Exclusive vs. non-exclusive
 - Whether objects may belong to a single or multiple clusters
- Partial versus complete
 - In some cases, we only want to cluster some of the data
- Hard vs. soft
 - In hard clustering each object can only belong to a single cluster
 - In soft (or “fuzzy”) clustering, an object belongs to every cluster with some probability

Text clustering

- Clustering algorithms
- Agglomerative Hierarchical Clustering
- K-means clustering

Clustering techniques

- **Similarity-based clustering:** require a *similarity function* to work. Each object can only belong to one cluster (*hard clustering*).
 - **Agglomerative clustering** (also called **hierarchical clustering**): gradually merge similar objects to generate clusters (“bottom-up”)
 - **Divisive clustering**: gradually divide the whole set into smaller clusters (“top-down”)
- **Model-based techniques:** rely on a probabilistic model to capture the latent structure of data
 - Typically, this is an example of *soft clustering*, since one object may be in multiple clusters (with some probability)

Similarity-based clustering

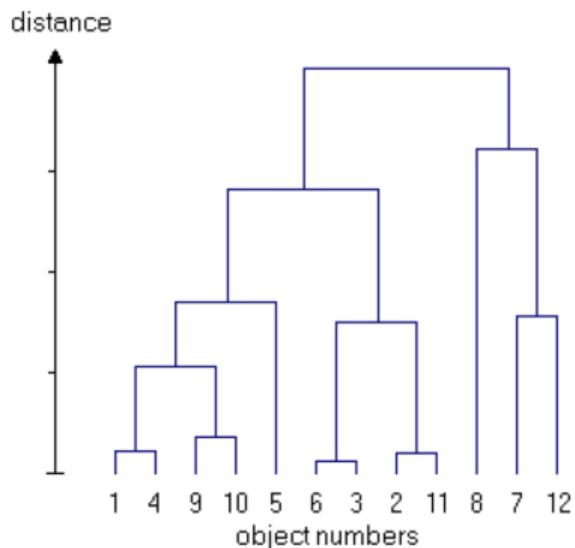
- Both agglomerative and divisive clustering methods require a document-document **similarity measure**, $\text{sim}(d_1, d_2)$
- In particular, the similarity measure needs to be
 - **symmetric**: $\text{sim}(d_1, d_2) = \text{sim}(d_2, d_1)$
 - **normalized**: $\text{sim}(d_1, d_2) \in [0, 1]$
- The choice of similarity measure is closely tied with how documents are represented

Text clustering

- Clustering algorithms
- Agglomerative Hierarchical Clustering
- K-means clustering

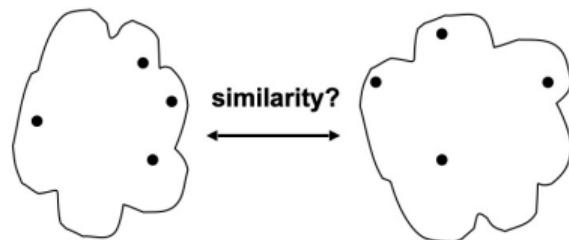
Agglomerative Hierarchical Clustering

- Progressively construct clusters to generate a hierarchy of merged groups (“bottom-up”)
- Start with each document being a cluster on its own, and gradually merge clusters into larger and larger groups until there is only one cluster left
- This series of merges forms a **dendrogram**
- The tree may then be segmented based on how many clusters are needed
 - Alternatively, the merging may be stopped when the desired number of clusters is found



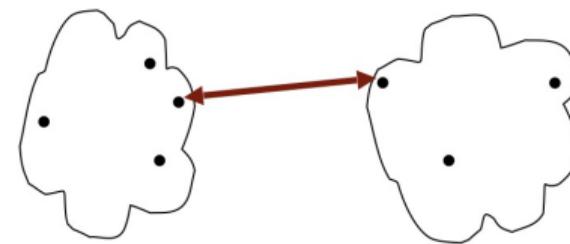
Measuring inter-cluster similarity

- Single-link
- Complete-link
- Average-link
- Prototype-based (centroid)



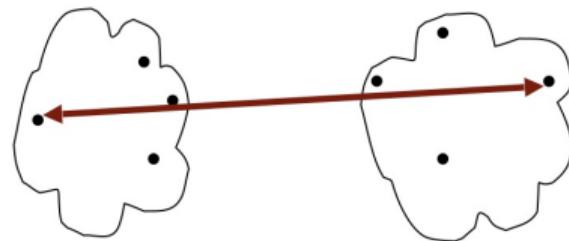
Single-link (“min”)

- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
 - Results in “looser” clusters



Complete-link (“max”)

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
 - Results in “tight” and “compact” clusters (tends to break large clusters)

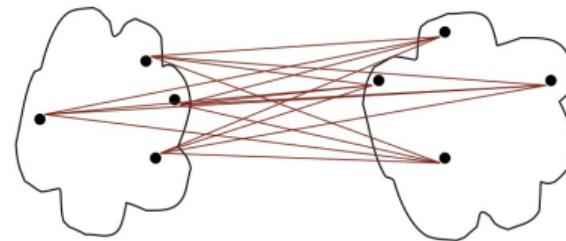


Average-link (“avg”)

- Similarity of two clusters is the average of pairwise similarity between points in the two clusters

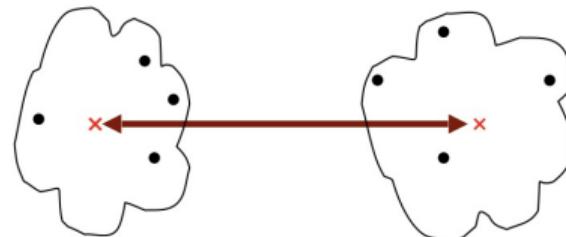
$$\text{sim}(C_i, C_j) = \frac{\sum_{x \in C_i, y \in C_j} \text{sim}(x, y)}{|C_i| \times |C_j|}$$

- Less susceptible to noise and outliers than single- and complete-link



Prototype-based (centroid)

- Represent clusters by their centroids and base their similarity on the similarity of the centroids
 - To find the centroid, one computes the (arithmetic) mean of the points' positions separately for each dimension



Text clustering

- Clustering algorithms
- Agglomerative Hierarchical Clustering
- K-means clustering

K-means clustering

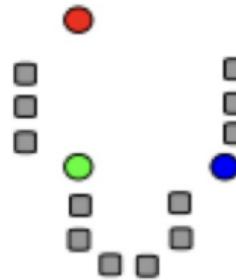
- Divisive clustering
- Start with an initial tentative clustering and iteratively improve it until we reach some stopping criterion
- It's a particular manifestation of the Expectation-Maximization algorithmic paradigm
- A cluster is represented with a **centroid**: representing all other objects in the cluster, usually as an average of all its members' values
- Finds a user-specified number of clusters (K)

Basic K-means algorithm

1. Select K points as initial centroids
2. **Repeat**
 - 2.1 Form K clusters by assigning each point to its closest centroid
 - 2.2 Recompute the centroid of each cluster
3. **Until** centroids do not change

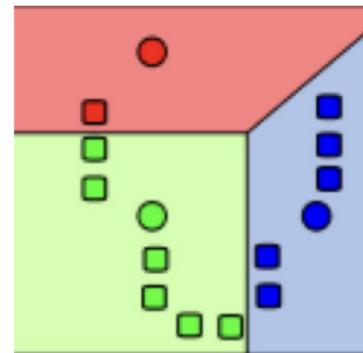
Basic K-means algorithm

1. Select K points as initial centroids \Leftarrow
2. Repeat
 - 2.1 Form K clusters by assigning each point to its closest centroid
 - 2.2 Recompute the centroid of each cluster
3. Until centroids do not change



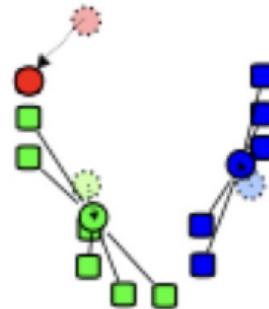
Basic K-means algorithm

1. Select K points as initial centroids
2. **Repeat**
 - 2.1 Form K clusters by assigning each point to its closest centroid \Leftarrow
 - 2.2 Recompute the centroid of each cluster
3. **Until** centroids do not change



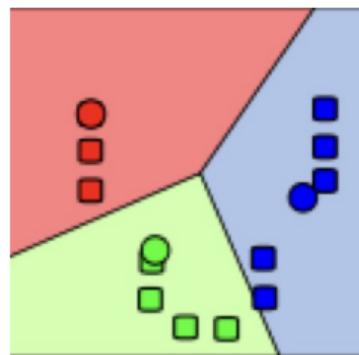
Basic K-means algorithm

1. Select K points as initial centroids
2. **Repeat**
 - 2.1 Form K clusters by assigning each point to its closest centroid
 - 2.2 Recompute the centroid of each cluster \Leftarrow
3. **Until** centroids do not change



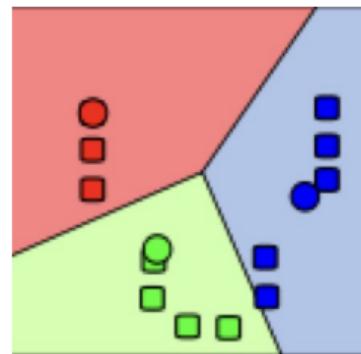
Basic K-means algorithm

1. Select K points as initial centroids
2. **Repeat** ⇐
 - 2.1 Form K clusters by assigning each point to its closest centroid
 - 2.2 Recompute the centroid of each cluster
3. **Until** centroids do not change



Basic K-means algorithm

1. Select K points as initial centroids
2. **Repeat**
 - 2.1 Form K clusters by assigning each point to its closest centroid
 - 2.2 Recompute the centroid of each cluster
3. **Until centroids do not change** \Leftarrow



Exercise

E2-3 Text classification sklearn

Summary

- Problem of text classification (binary and multiclass variants)
- Feature-bases text classifiers (bag-of-words representation, document-term matrix)
- Naive Bayes classifier
- Evaluation (confusion matrix, binary/multiclass)
- Evaluation measures (accuracy, precision, recall, F1, micro- and macro-averaging)
- Training/test splits, cross-validation
- Problem of (text) clustering
- Similarity-based clustering algorithms (Agglomerative Hierarchical Clustering and K-means)

Reading

- Text Data Management and Analysis (Zhai&Massung)
 - Chapter 14 (Sections 14.1, 14.2)
 - Chapter 15 (Sections 15.1–15.4, 15.5.2, 15.6)

Search Engine Architecture and Basic Retrieval Models

[DAT640] Information Retrieval and Text Mining

Ivica Kostric & Krisztian Balog

University of Stavanger

August 29, 2022



CC BY 4.0

In this module

1. Introduction to Information Retrieval
2. Search engine architecture
3. Indexing and query processing
4. Retrieval models

Introduction to Information Retrieval

Information Retrieval (IR)

“Information retrieval is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information.”

(Salton, 1968)

Modern definition

“Making the **right information** available to the **right person** at the **right time in the right form**.”



Searching in databases

Query: *records with balance > \$50,000 in branches located in Amherst, MA.*

Name	Branch	Balance
Sam I. Am	Amherst, MA	\$95,342.11
Patty MacPatty	Amherst, MA	\$23,023.23
Bobby de West	Amherst, NY	\$78,000.00
Xing O'Boston	Boston, MA	\$50,000.01

Searching in text

Query: *deadly disease due to diet*

The screenshot shows a Google search results page with the query "deadly disease due to diet" entered into the search bar. The results are filtered under the "All" tab. The first result is a link to "Illnesses Caused by Bad Diet | LIVESTRONG.COM". The second result is "The 5 Deadliest Food-Borne Illnesses - and How to Prevent Them" from Forbes. The third result is "Dangerous diseases that are caused by fast foods Body - Medical Daily". The fourth result is "Britain's poor diet more deadly than its smoking habit as alcohol ...". The fifth result is "Lifestyle disease - Wikipedia, the free encyclopedia". Each result includes a snippet of text and a timestamp or date.

deadly disease due to diet - Google

https://www.google.no/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=deadly%20disease%20due%20to%20...

Google

deadly disease due to diet

All Images Videos News Shopping More Search tools

About 816,000 results (0.57 seconds)

Illnesses Caused by Bad Diet | LIVESTRONG.COM
www.livestrong.com › Food and Drink ▾
Jan 26, 2014 - The body relies on the nutrients it gets from food to keep it running smoothly and in good health. ... List of Diseases Caused by Poor Nutrition.

The 5 Deadliest Food-Borne Illnesses - and How to Prevent Them
www.forbes.com/...the-5-deadliest-food-borne-illnesses-and-how-to-prevent-them/ ▾
Sep 28, 2011 - Here, the 5 deadliest types of food-borne bacteria and how to keep ... While listeriosis, the disease caused by the bacteria Listeria, is less ...

Dangerous diseases that are caused by fast foods Body - Medical Daily
www.medicaldaily.com/dangerous-diseases-are-caused-fast-foods-body-234278 ▾
Nov 8, 2010 - If you have seen the movie *"Super Size Me"*, which raised concerns about the nutrition levels of fast food in the United States so long ago, ...

Britain's poor diet more deadly than its smoking habit as alcohol ...
www.telegraph.co.uk › News › Health › Health News ▾
Sep 15, 2015 - Britain's poor diet more deadly than its smoking habit as alcohol related ... The number of deaths from alcohol-related disease has soared.

Lifestyle disease - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/Lifestyle_disease ▾
Lifestyle diseases are defined as diseases linked with the way people live their life. This is commonly caused by alcohol, drug and smoking abuse as well as lack of physical activity and unhealthy eating.

Which of the results are relevant?

Core problem in IR

How to match information needs (“queries”) and information objects (“documents”)

Core issues in IR

- **Relevance**

- Simple (and simplistic) definition: A relevant document contains the information that a person was looking for when they submitted a query to the search engine
 - Many factors influence a person's decision about what is relevant (task, context, novelty, ...)
 - Distinction between *topical relevance* vs. *user relevance* (all other factors)
- *Retrieval models* define a view of relevance
- *Ranking algorithms* used in search engines are based on retrieval models
- Most models are based on statistical properties of text rather than linguistic
- Exact matching of words is not enough!

Core issues in IR

- **Evaluation**

- Experimental procedures and measures for comparing system output with user expectations
- Typically use test collection of documents, queries, and relevance judgments
- *Recall* and *precision* are two examples of effectiveness measures

Core issues in IR

- **Information needs**

- Keyword queries are often poor descriptions of actual information needs
- Interaction and context are important for understanding user intent
- Query modeling techniques such as query expansion, aim to refine the information need and thus improve ranking

Dimensions of IR

- IR is more than just text, and more than just web search
 - Although these are central
- Content
 - Text, images, video, audio, scanned documents, ...
- Applications
 - Web search, vertical search, enterprise search, desktop search, social search, legal search, chatbots and virtual assistants, ...
- Tasks
 - Ad hoc search, filtering, question answering, response ranking, ...

Search engines in operational environments

- Performance
 - Response time, indexing speed, etc.
- Incorporating new data
 - Coverage and freshness
- Scalability
 - Growing with data and users
- Adaptability
 - Tuning for specific applications

Search engine architecture

Search engine architecture

- A software architecture consists of software components, the interfaces provided by those components, and the relationships between them
 - Describes a system at a particular level of abstraction
- Architecture of a search engine determined by 2 requirements
 - Effectiveness (quality of results)
 - Efficiency (response time and throughput)
- Two main processes:
 - Indexing (offline)
 - Querying (online)

Search engine architecture

- Indexing process
- Query process

Indexing

Indexing is the process that makes a document collection searchable

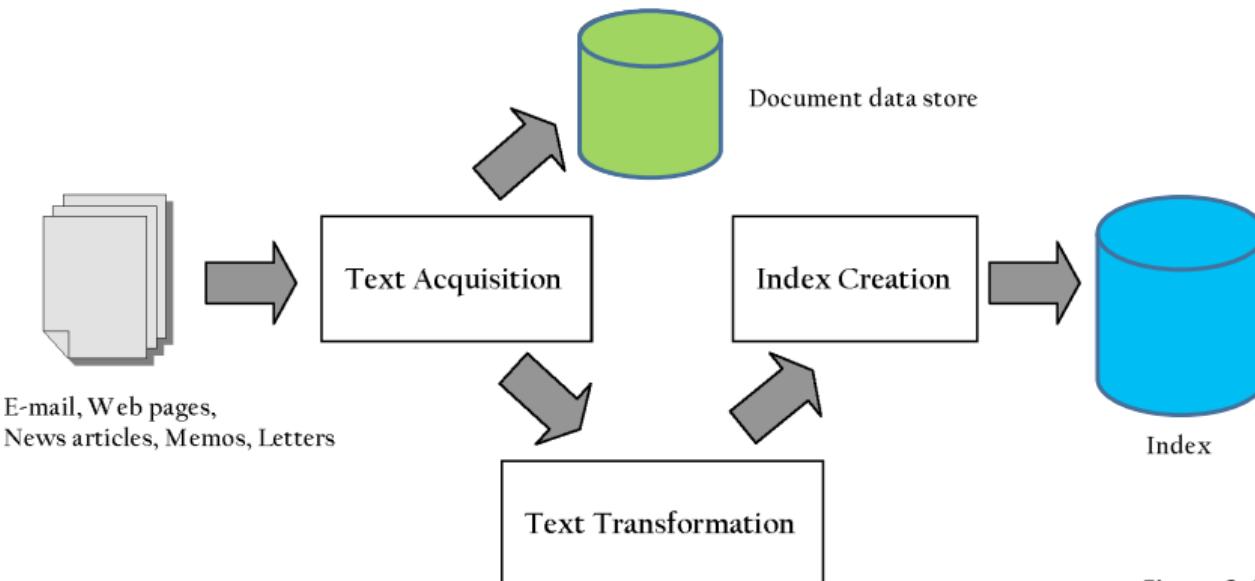
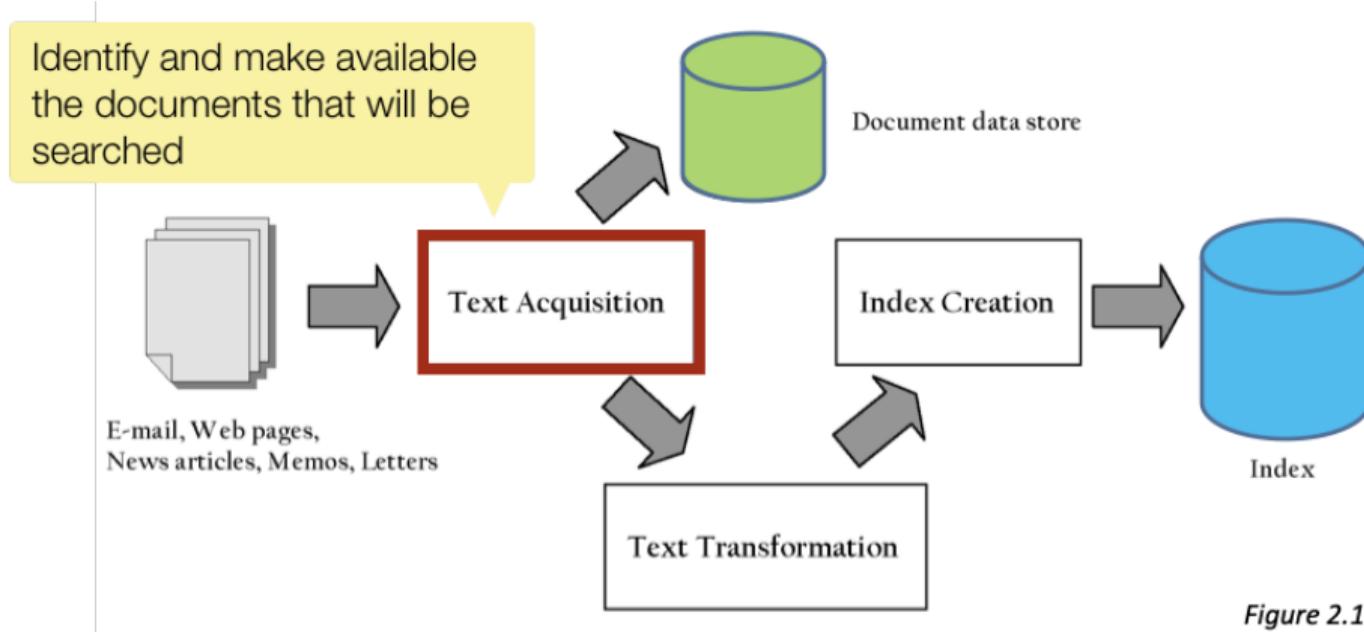


Figure 2.1

Text acquisition



Text acquisition

- **Crawler:** identifies and acquires documents for search engine
 - Many types: web, enterprise, desktop, etc.
 - Web crawlers follow links to find documents
 - Must efficiently find huge numbers of web pages (coverage) and keep them up-to-date (freshness)
 - Single site crawlers for site search
 - Topical or focused crawlers for vertical search
 - Document crawlers for enterprise and desktop search
 - Follow links and scan directories
- **Feeds:** real-time streams of documents
 - E.g., web feeds for news, blogs, video, radio, TV
 - RSS is common standard

Document data store

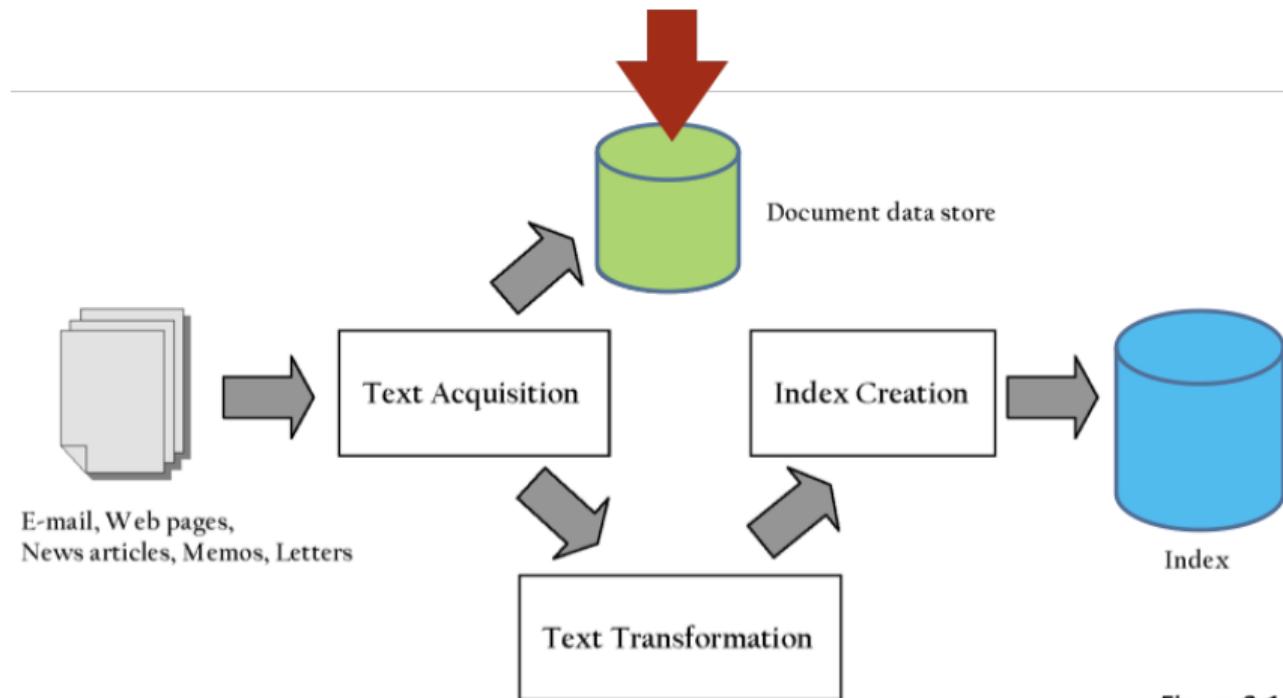
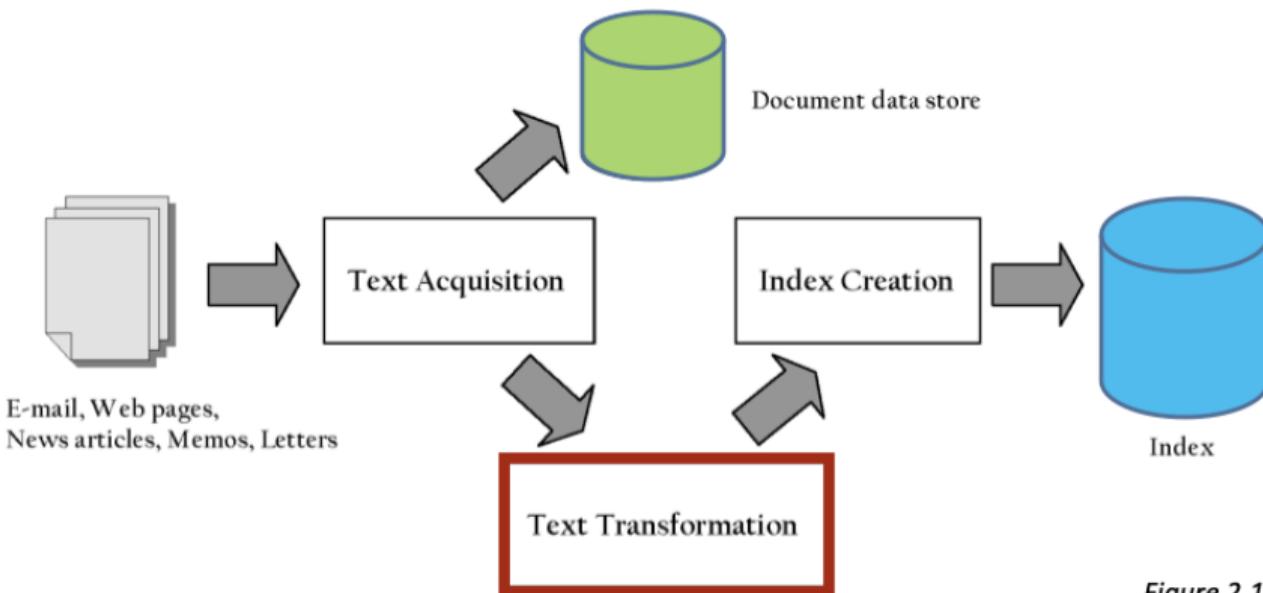


Figure 2.1

Document data store

- Stores text, metadata, and other related content for documents
 - Metadata is information about document such as type and creation date
 - Other content includes links, anchor text
- Provides fast access to document contents for search engine components
 - E.g. result list generation
- Could use relational database system
 - More typically, a simpler, more efficient storage system is used due to huge numbers of documents

Text transformation



Transform documents into index terms or features

Figure 2.1

Text transformation

- Tokenization, stopword removal, stemming
- Semantic annotation
 - Named entity recognition
 - Text categorization
 - ...
- Link analysis
 - Anchor text extraction
 - ...

Index creation

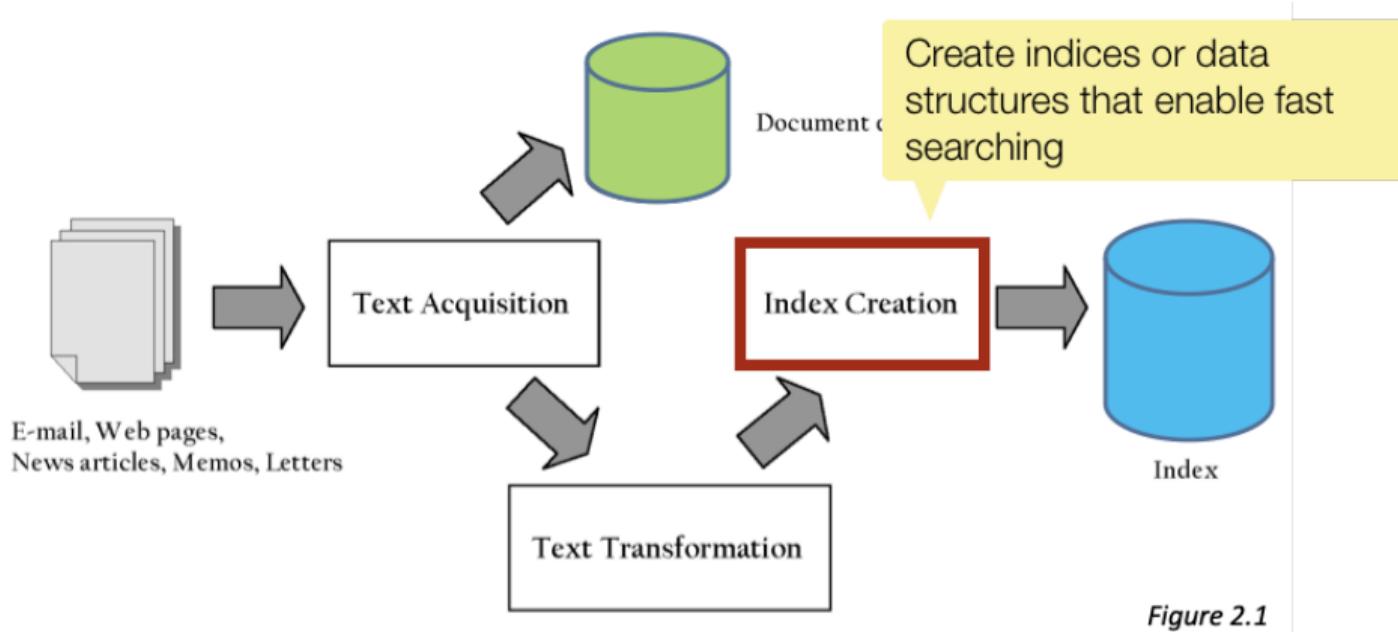


Figure 2.1

Index creation

- Gathers counts and positions of words and other features used in ranking algorithm
- Format is designed for fast query processing
- Index may be distributed across multiple computers and/or multiple sites
- (More in a bit)

Search engine architecture

- Indexing process
- Query process

Query process

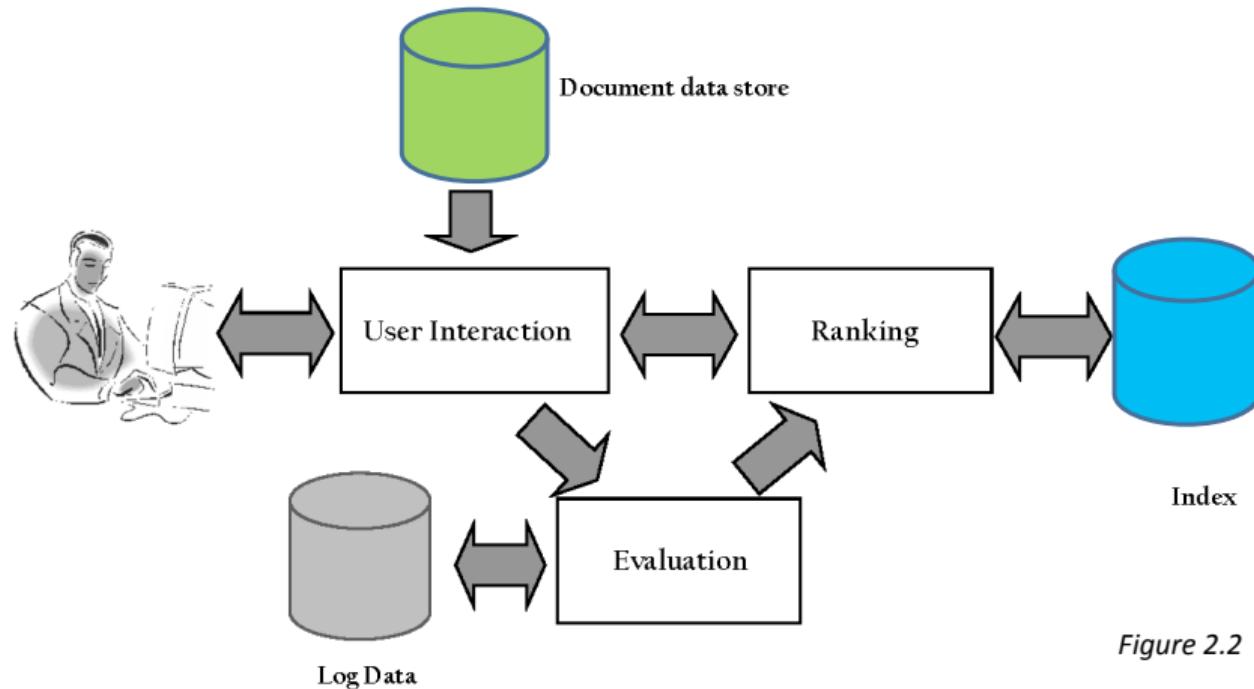


Figure 2.2

User interaction

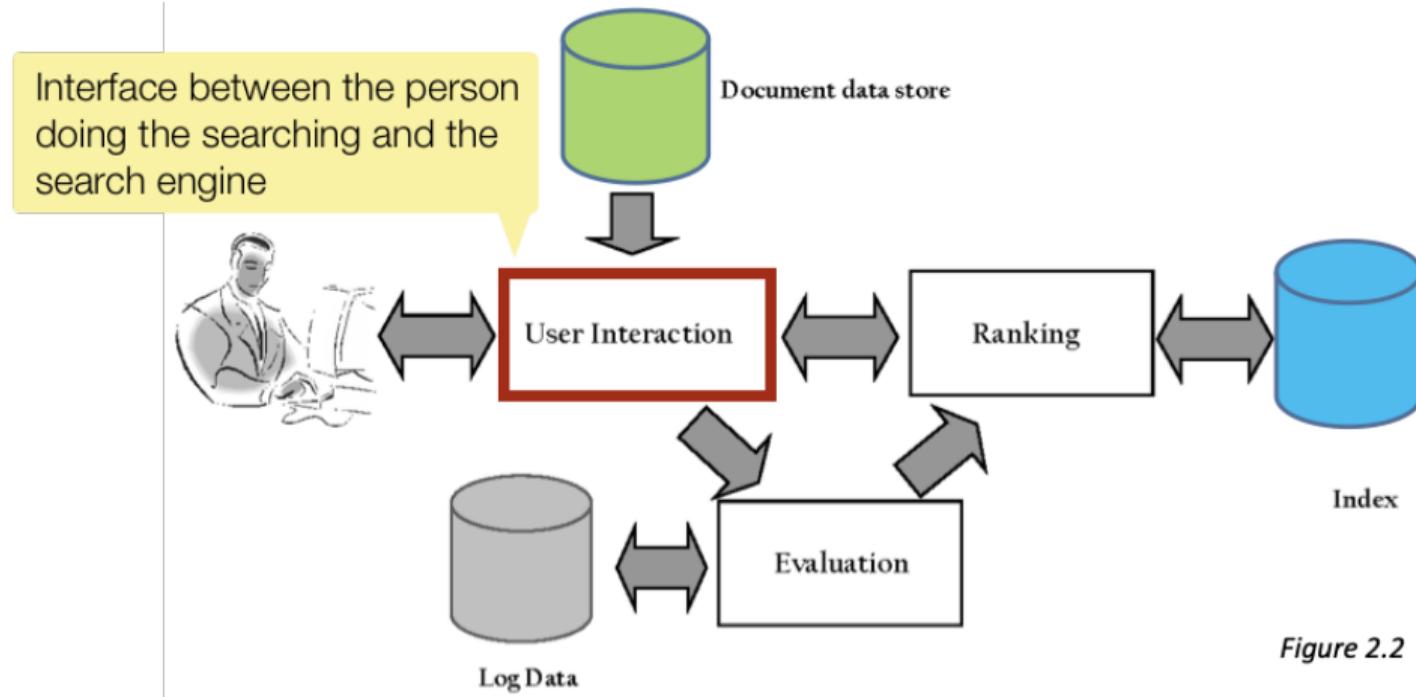
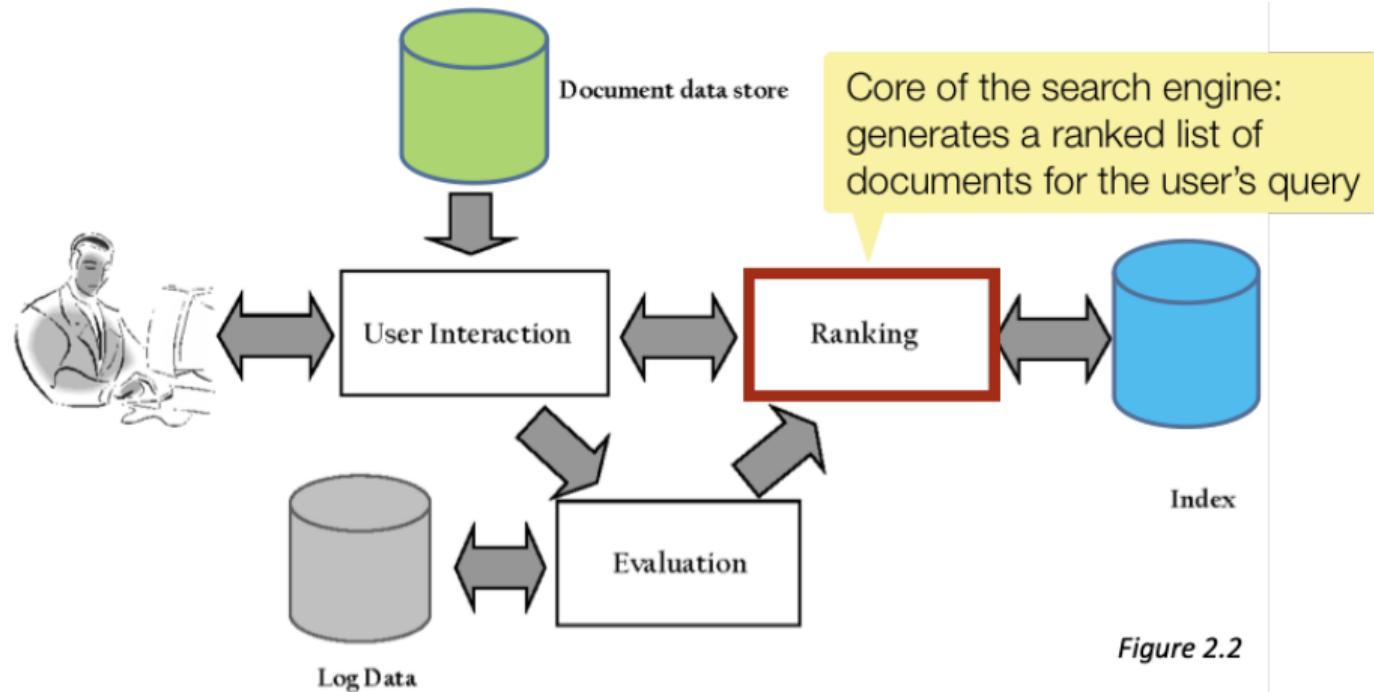


Figure 2.2

User interaction

- **Query input:** accepting the user's query and transforming it into index terms
 - Most web search query languages are very simple (i.e., small number of operators)
 - There are more complicated query languages (proximity operators, structure specification, etc.)
- **Results output:** taking the ranked list of documents from the search engine and organizing it into the results shown to the user
 - Generating *snippets* to show how queries match documents
 - *Highlighting* matching words and passages
 - May provide *clustering* of search results and other visualization tools

Ranking



Ranking

- Calculates scores for documents using a *ranking algorithm*, which is based on a *retrieval model*
- Core component of search engine
- Many variations of ranking algorithms and retrieval models exist
- **Performance optimization:** designing ranking algorithms for efficient processing
 - *Term-at-a-time* vs. *document-at-a-time* processing
 - *Safe* vs. *unsafe* optimizations
- **Distribution:** processing queries in a distributed environment
 - *Query broker* distributes queries and assembles results

Evaluation

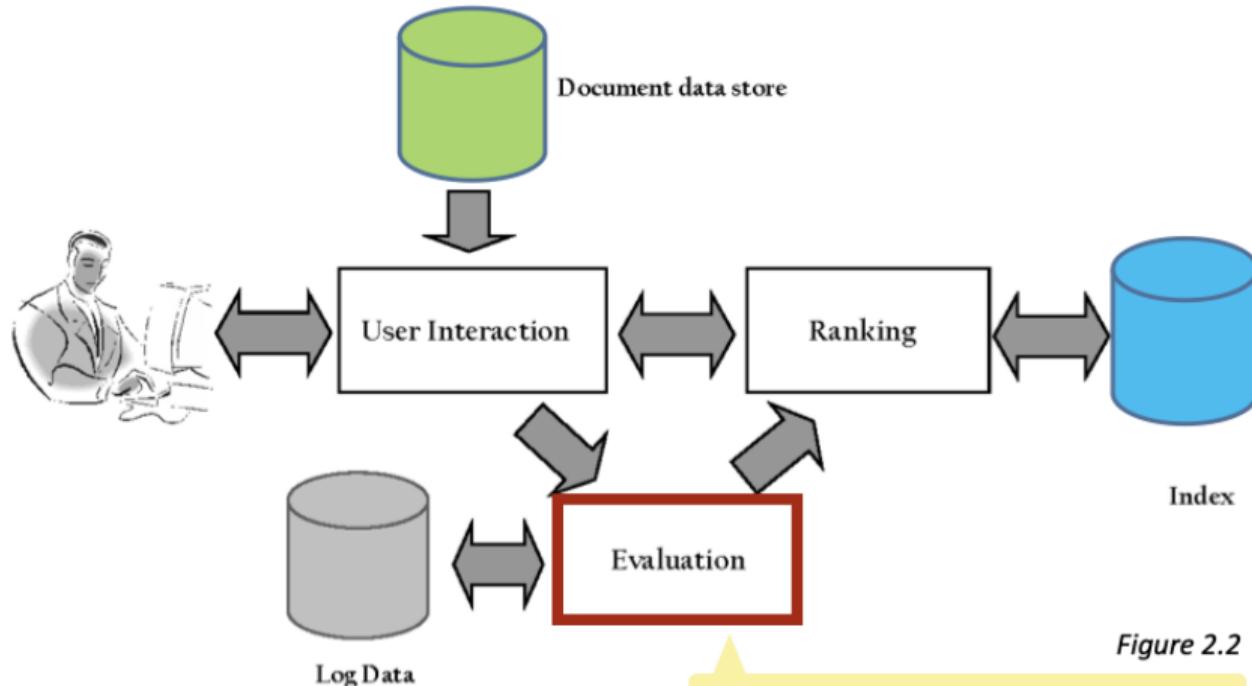


Figure 2.2

Measure and monitor
effectiveness and efficiency.
Record and analyze usage data

Evaluation

- **Logging** user queries and interaction is crucial for improving search effectiveness and efficiency
 - *Query logs* and *clickthrough data* used for query suggestion, spell checking, query caching, ranking, advertising search, and other components
- **Ranking analysis:** measuring and tuning ranking effectiveness
- **Performance analysis:** measuring and tuning system efficiency

Indexing and query processing

Indexing and query processing

- Indexing
- Query processing

Indices

- Text search has unique requirements, which leads to unique data structures
- Indices are data structures designed to make search faster
- Most common data structure is the *inverted index*
 - General name for a class of structures
 - “Inverted” because documents are associated with words, rather than words with documents
 - Similar to a concordance

Index

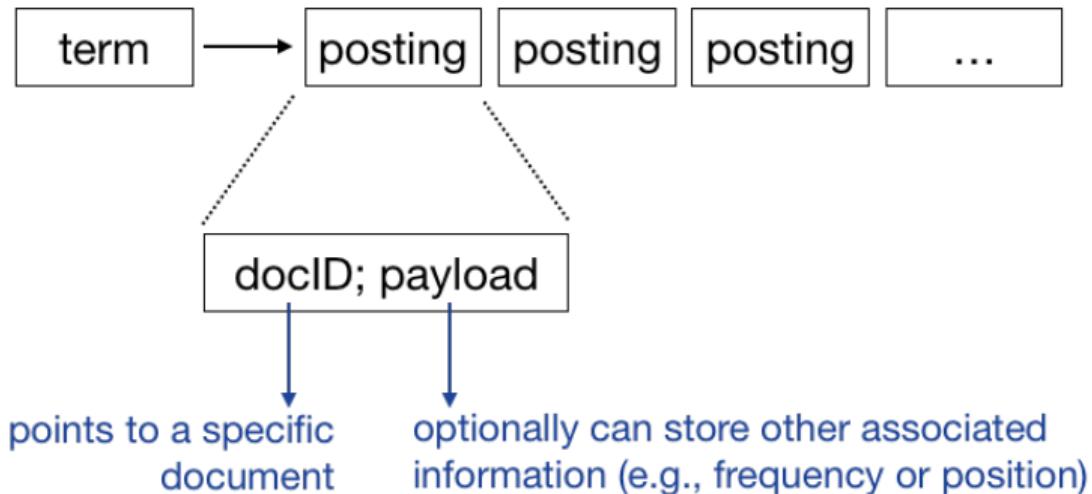
Note: *italic* page numbers indicate specific methods, whilst **bold** page numbers indicate major sections on the subject.

Abraham Maslow	10	argument.....	110
acceptance	138, 228, 229	Aristotle	250
accepting	27	arousal	25, 27, 114, 131, 210
action	132, 261	as if.....	233
active care.....	188	Asch, Solomon.....	73
active listening.....	176	Ashby, Ross.....	38
advocate	242	asking.....	235
affirmation	246	aspirations	192
agreeableness.....	57	assertion.....	217
aha.....	81	association.....	72
aim inhibition.....	209	assonance.....	118
alignment	18, 61, 92, 187	assumption	217
alliteration	118	assumptions.....	233

Inverted Index

- Each index term is associated with a *postings list* (or *inverted list*)
 - Contains lists of documents, or lists of word occurrences in documents, and other information
 - Each entry is called a *posting*
 - The part of the posting that refers to a specific document or location is called a *pointer*
 - Each document in the collection is given a unique number (docID)
 - The posting can store additional information, called the *payload*
 - Lists are usually *document-ordered* (sorted by docID)

Postings list



Example

- S_1 Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.
- S_2 Fishkeepers often use the term tropical fish to refer only those requiring fresh water, with saltwater tropical fish referred to as marine fish.
- S_3 Tropical fish are popular aquarium fish, due to their often bright coloration.
- S_4 In freshwater fish, this coloration typically derives from iridescence, while salt water fish are generally pigmented.

Four sentences from the Wikipedia entry for *tropical fish*

Simple inverted index

Each document that contains the term is a posting. No additional payload.

docID

and	1	only	2
aquarium	3	pigmented	4
are	3	popular	3
around	4	refer	2
as	2	referred	2
both	1	requiring	2
bright	3	salt	1 4
coloration	3	saltwater	2
derives	4	species	1
due	3	term	2
environments	1	the	1 2
fish	1	their	3
fishkeepers	2	this	4
found	1	those	2
fresh	2	to	2 3
freshwater	1	tropical	1 2 3
from	4	typically	4
generally	4	use	2
in	1	water	1 2 4
include	1	while	4
including	1	with	2
iridescence	4	world	1
marine	2		
often	2		
	3		

Inverted index with counts

The payload is the frequency of the term in the document.

Supports better ranking algorithms.

docID: freq

and	1:1	only	2:1
aquarium	3:1	pigmented	4:1
are	3:1	popular	3:1
around	1:1	refer	2:1
as	2:1	referred	2:1
both	1:1	requiring	2:1
bright	3:1	salt	1:1
coloration	3:1	saltwater	2:1
derives	4:1	species	1:1
due	3:1	term	2:1
environments	1:1	the	1:1
fish	1:2	their	3:1
fishkeepers	2:1	this	4:1
found	1:1	those	2:1
fresh	2:1	to	2:2
freshwater	1:1	tropical	1:2
from	4:1	typically	4:1
generally	4:1	use	2:1
in	1:1	water	1:1
include	1:1	while	4:1
including	1:1	with	2:1
iridescence	4:1	world	1:1
marine	2:1		
often	2:1		
	3:1		

Inverted index with term positions

There is a separate posting for each term occurrence in the document. The payload is the term position.

Supports proximity matches. E.g., find “tropical” within 5 words of “fish”

docID. position

and	1,15	marine	2,22
aquarium	3,5	often	2,2
are	3,3	only	2,10
around	1,9	pigmented	4,16
as	2,21	popular	3,4
both	1,13	refer	2,9
bright	3,11	referred	2,19
coloration	3,12	requiring	2,12
derives	4,7	salt	1,16
due	3,7	saltwater	2,16
environments	1,8	species	1,18
fish	1,2 1,4 2,7 2,18 2,23 3,2 3,6 4,3 4,13	term	2,5
fishkeepers	2,1	the	1,10
found	1,5	their	3,9
fresh	2,13	this	4,4
freshwater	1,14 4,2	those	2,11
from	4,8	to	2,8
generally	4,15	tropical	1,1
in	1,6 4,1	typically	4,6
include	1,3	use	2,3
including	1,12	water	1,17
iridescence	4,9	while	4,10
		with	2,15
		world	1,11

Issues

- Compression
 - Inverted lists are very large
 - Compression of indexes saves disk and/or memory space
- Optimization techniques to speed up search
 - Read less data from inverted lists
 - “Skipping” ahead
 - Calculate scores for fewer documents
 - Store highest-scoring documents at the beginning of each inverted list
- Distributed indexing

Example

Create a simple inverted index for the following document collection

Doc 1	new home sales top forecasts
Doc 2	home sales rise in july
Doc 3	increase in home sales in july
Doc 4	july new home sales rise

Solution

new	1	4		
home	1	2	3	4
sales	1	2	3	4
top	1			
forecasts	1			
rise	2	4		
in	2	3		
july	2	3	4	
increase	3			

Exercise

E3-1 Inverted index

Indexing and query processing

- Indexing
- Query processing

Scoring documents

- Objective: estimate the relevance of documents in the collection w.r.t. the input query q (so that the highest-scoring ones can be returned as retrieval results)
- In principle, this would mean scoring all documents in the collection
- In practice, we're only interested in the top- k results for each query
- Common form of a retrieval function

$$score(d, q) = \sum_{t \in q} w_{t,d} \times w_{t,q}$$

- where $w_{t,d}$ is the weight of term t in document d and $w_{t,q}$ is the weight of that term in the query q

Question

How to compute these retrieval functions for all document in the collection?

Query processing

- Strategies for processing the data in the index for producing query results
 - We benefit from the inverted index by scoring only documents that contain at least one query term
- Term-at-a-time
 - Accumulates scores for documents by processing term lists one at a time
- Document-at-a-time
 - Calculates complete scores for documents by processing all term lists, one document at a time
- Both approaches have optimization techniques that significantly reduce time required to generate scores

Term-at-a-time query processing

```
scores = {}      // score accumulator maps doc IDs to scores
for w ∈ q do
    for d, count ∈ Idx.fetch_docs(w) do
        scores[d] = scores[d] + score_term(count)
    end for
end for
return top k documents from scores
```

Term-at-a-time query processing

salt	1:1	4:1		
partial scores	1:1	4:1		
<hr/>				
old partial scores	1:1	4:1		
water	1:1	2:1	4:1	
<hr/>				
new partial scores	1:2	2:1	4:2	
<hr/>				
old partial scores	1:2	2:1	4:2	
tropical	1:2	2:2	3:1	
final scores	1:4	2:3	3:1	4:2

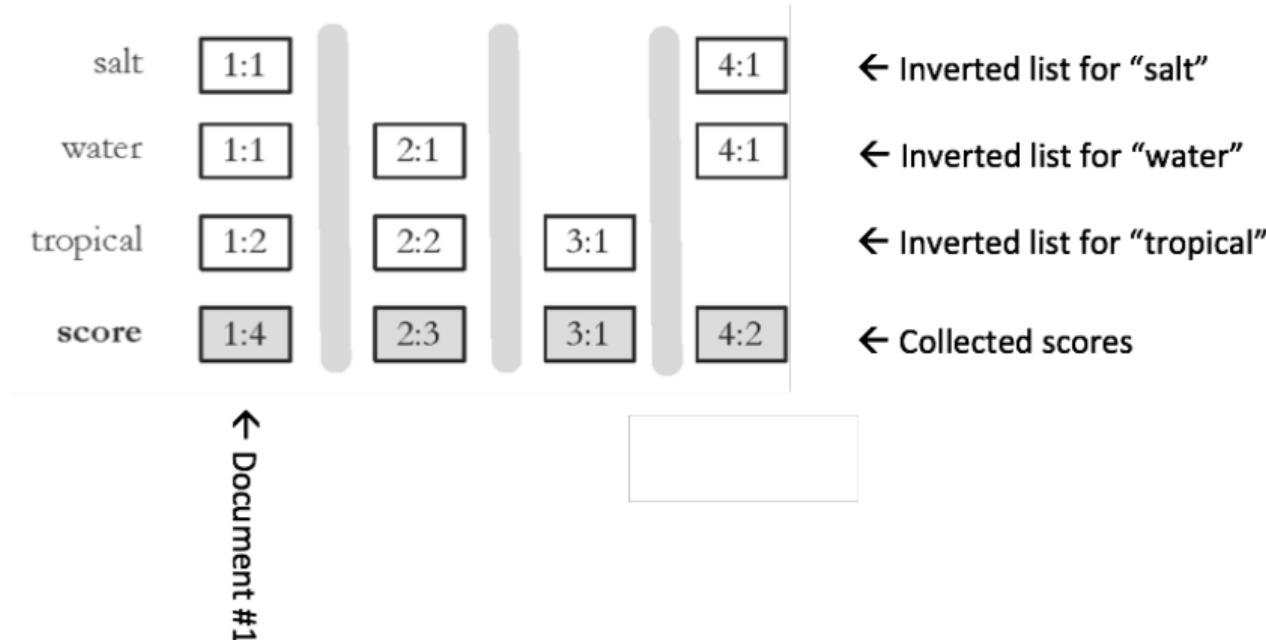
From term-at-a-time to document-at-a-time query processing

- Term-at-a-time query processing
 - Advantage: simple, easy to implement
 - Disadvantage: the score accumulator will be the size of document matching at least one query term
- Document-at-a-time query processing
 - Make the score accumulator data structure smaller by scoring entire documents at once. We are typically interested only in top- k results
 - Idea #1: hold the top- k best completely scored documents in a priority queue
 - Idea #2: Documents are sorted by document ID in the posting list. If documents are scored ordered by their IDs, then it is enough to iterate through each query term's posting list only once
 - Keep a pointer for each query term. If the posting equals the document currently being scored, then get the term count and move the pointer; otherwise the current document does not contain the query term

Document-at-a-time query processing

```
context = {}      // maps a document to a list of matching terms
for w ∈ q do
    for d, count ∈ Idx.fetch_docs(w) do
        context[d].append(count)
    end for
end for
priority_queue = {}      // low score is treated as high priority
for d, term_counts ∈ context do
    score = 0
    for count ∈ term_counts do
        score = score + score_term(count)
    end for
    priority_queue.push(d, score)
    if priority_queue.size() > k then
        priority_queue.pop()      // removes lowest score so far
    end if
end for
Return sorted documents from priority_queue
```

Document-at-a-time query processing



Exercise

E3-2 Query processing DaaT

Retrieval models

Retrieval models

- Bag-of-words representation
 - Simplified representation of text as a bag (multiset) of words
 - Disregards word ordering, but keeps multiplicity
- Common form of a retrieval function

$$score(d, q) = \sum_{t \in q} w_{t,d} \times w_{t,q}$$

- Note: we only consider terms in the query, $t \in q$
- $w_{t,d}$ is the term's weight in the document
- $w_{t,q}$ is the term's weight in the query
- $score(d, q)$ is (in principle) to be computed for every document in the collection

Example retrieval functions

- General scoring function

$$score(d, q) = \sum_{t \in q} w_{t,d} \times w_{t,q}$$

- **Example 1:** Count the number of matching query terms in the document

$$w_{t,d} = \begin{cases} 1, & c_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- where $c_{t,d}$ is the number of occurrences of term t in document d

$$w_{t,q} = c_{t,q}$$

- where $c_{t,q}$ is the number of occurrences of term t in query q

Example retrieval functions

- General scoring function

$$score(d, q) = \sum_{t \in q} w_{t,d} \times w_{t,q}$$

- **Example 2:** Instead of using raw term frequencies, assign a weight that reflects the term's importance

$$w_{t,d} = \begin{cases} 1 + \log c_{t,d}, & c_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- where $c_{t,d}$ is the number of occurrences of term t in document d

$$w_{t,q} = c_{t,q}$$

- where $c_{t,q}$ is the number of occurrences of term t in query q

Retrieval models

- Vector space model
- Term weighting
- Vector space models (cont'd)
- Language models
- Summary

Vector space model

- Basis of most IR research in the 1960s and 70s
- Still used
- Provides a simple and intuitively appealing framework for implementing
 - Term weighting
 - Ranking
 - Relevance feedback

Vector space model

- Main underlying assumption: if document d_1 is more similar to the query than another document d_2 , then d_1 is *more relevant* than d_2
- Documents and queries are viewed as vectors in a high dimensional space, where each dimension corresponds to a term

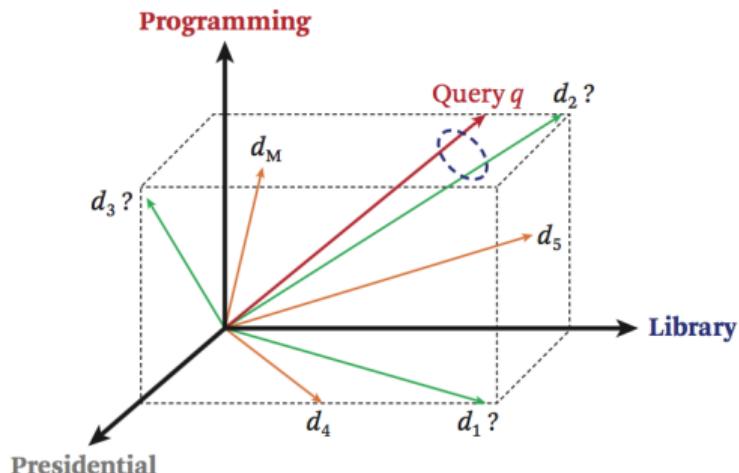


Figure: Illustration is taken from (Zhai&Massung, 2016)[Fig. 6.2]

Instantiation

- The vector space model provides a *framework* that needs to be instantiated by deciding
 - How to select terms? (i.e., vocabulary construction)
 - How to place documents and queries in the vector space (i.e., term weighting)
 - How to measure the similarity between two vectors (i.e., similarity measure)

Simple instantiation (bit vector representation)

- Each word in the vocabulary V defines a dimension
- Bit vector representation of queries and documents (i.e., only term presence/absence)
- Similarity measure is the dot product

$$sim(q, d) = \vec{q} \cdot \vec{d} = \sum_{t \in V} w_{t,q} \times w_{t,d}$$

- where $w_{t,q}$ and $w_{t,d}$ are either 0 or 1

Question

What are potential shortcomings of this simple instantiation?

Retrieval models

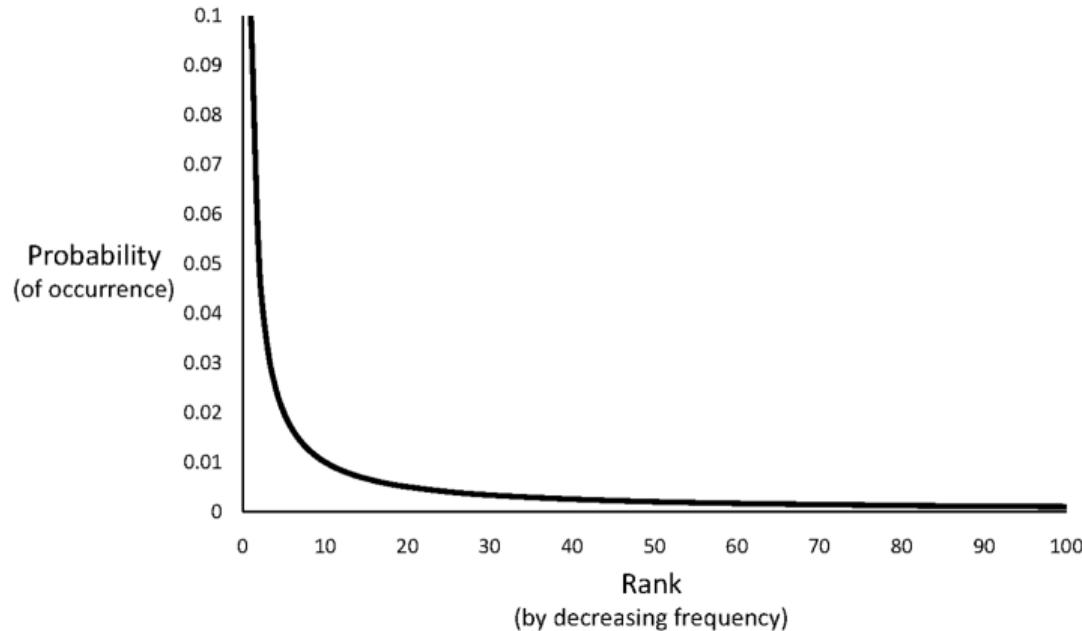
- Vector space model
- Term weighting
- Vector space models (cont'd)
- Language models
- Summary

English language

- Most frequent words
 - the (7%)
 - of (3.5%)
 - and (2.8%)
- Top 135 most frequent words account for half of the words used

Zip's law

- Given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table
 - Word number n has a frequency proportional to $1/n$



Term weighting

- Intuition #1: terms that appear often in a document should get high weights
 - E.g., The more often a document contains the term “dog,” the more likely that the document is “about” dogs
- Intuition #2: terms that appear in many documents should get low weights
 - E.g., stopwords, like “a,” “the,” “this,” etc.
- How do we capture this mathematically?
 - Term frequency
 - Inverse document frequency

Term frequency (TF)

- We write $c_{t,d}$ for the raw count of a term in a document
- **Term frequency** $tf_{t,d}$ reflects the importance of a term (t) in a document (d)
- Variants
 - Binary: $tf_{t,d} \in \{0, 1\}$
 - Raw count: $tf_{t,d} = c_{t,d}$
 - **L1-normalized:** $tf_{t,d} = \frac{c_{t,d}}{|d|}$
 - where $|d|$ is the length of the document, i.e., the sum of all term counts in d :
 $|d| = \sum_{t \in d} c_{t,d}$
 - L2-normalized: $tf_{t,d} = \frac{c_{t,d}}{\|d\|}$
 - where $\|d\| = \sqrt{\sum_{t \in d} (c_{t,d})^2}$
 - Log-normalized: $tf_{t,d} = 1 + \log c_{t,d}$
 - ...
- By default, when we refer to TF we will mean the L1-normalized version

Inverse document frequency (IDF)

- **Inverse document frequency** idf_t reflects the importance of a term (t) in a collection of documents
 - The more documents that a term occurs in, the less discriminating the term is between documents, consequently, the less “useful”

$$idf_t = \log \frac{N}{n_t}$$

- where N is the total number of documents in the collection and n_t is the number of documents that contain t
- Log is used to “dampen” the effect of IDF

IDF illustration¹

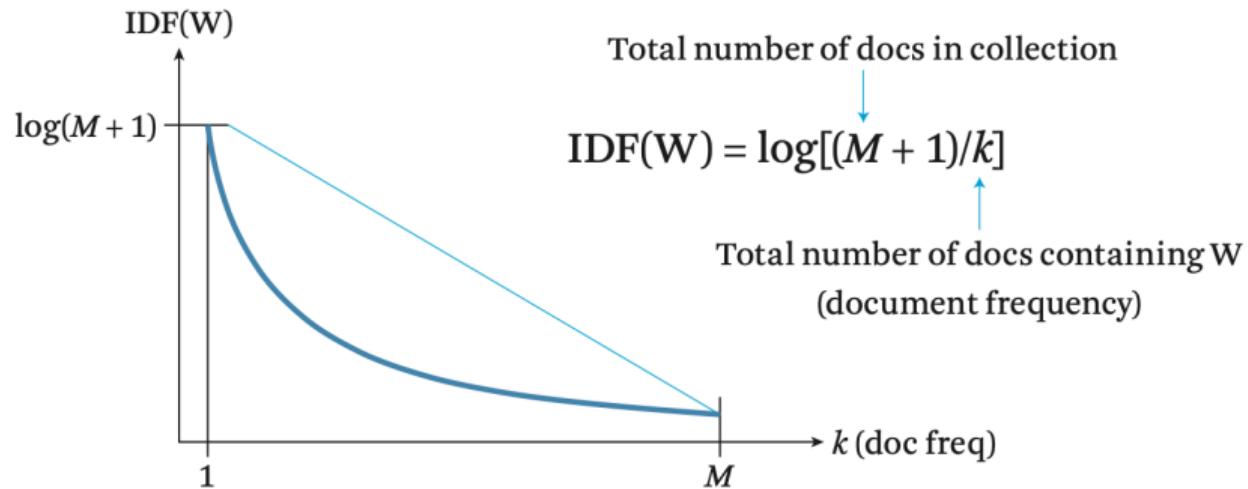


Figure: Illustration of the IDF function as the document frequency varies. Figure is taken from (Zhai&Massung, 2016)[Fig. 6.10]

¹Note that the textbook uses a slightly different IDF formula with $+1$ in the numerator.

Term weighting (TF-IDF)

- Combine TF and IDF weights by multiplying them:

$$tfidf_{t,d} = tf_{t,d} \cdot idf_t$$

- Term frequency weight measures importance in document
- Inverse document frequency measures importance in collection

Exercise

E3-3 Document-term matrix

Exercise

E3-4 TFIDF weighting

Retrieval models

- Vector space model
- Term weighting
- Vector space models (cont'd)
- Language models
- Summary

Improved instantiation (TF-IDF weighting)

- Idea: incorporate term importance by considering term frequency (TF) and inverse document frequency (IDF)
 - TF rewards terms that occur frequently in the document
 - IDF rewards terms that do not occur in many documents
- A possible ranking function using the TF-IDF weighting scheme:

$$score(d, q) = \sum_{t \in q \cap d} tf_{t,q} \times tf_{t,d} \times idft$$

- Note: the above formula uses raw term frequencies and applies IDF only on one of the (document/query) vectors

Many different variants out there!

- Different variants of TF and IDF
- Different TF-IDF weighting for the query and for the document
- Different similarity measure (e.g., cosine)

- BM25 was created as the result of a series of experiments (“Best Match”)
- Popular and effective ranking algorithm
- The reasoning behind BM25 is that good term weighting is based on three principles
 - Term frequency
 - Inverse document frequency
 - Document length normalization

BM25 scoring

$$score(d, q) = \sum_{t \in q} \frac{c_{t,d} \times (1 + k_1)}{c_{t,d} + k_1(1 - b + b \frac{|d|}{avgdl})} \times idf_t$$

- Parameters
 - k_1 : calibrating term frequency scaling
 - b : document length normalization
- Note: several slight variations of BM25 exist!

Recall: TF transformation

- Many different ways to transform raw term frequency counts

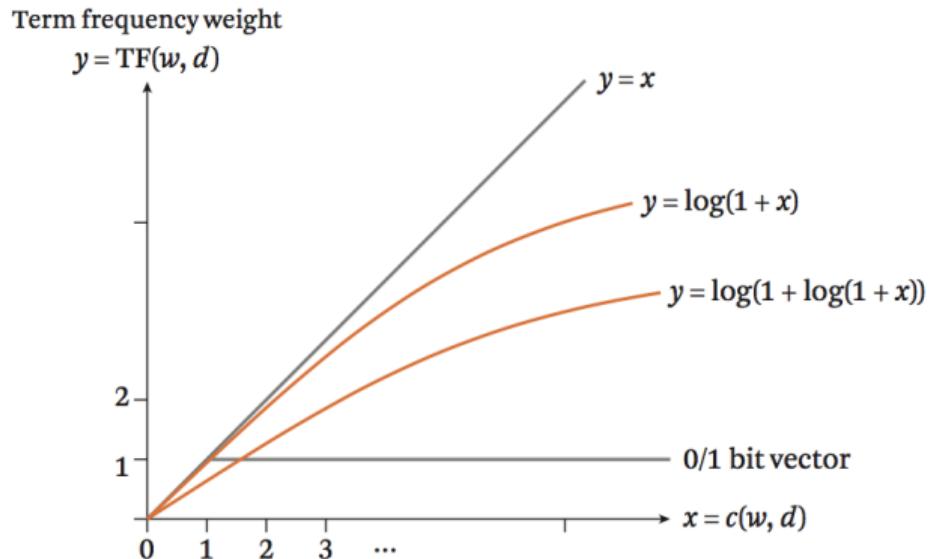


Figure: Illustration is taken from (Zhai&Massung, 2016)[Fig. 6.14]

BM25 TF transformation

- Idea: term saturation, i.e., repetition is less important after a while

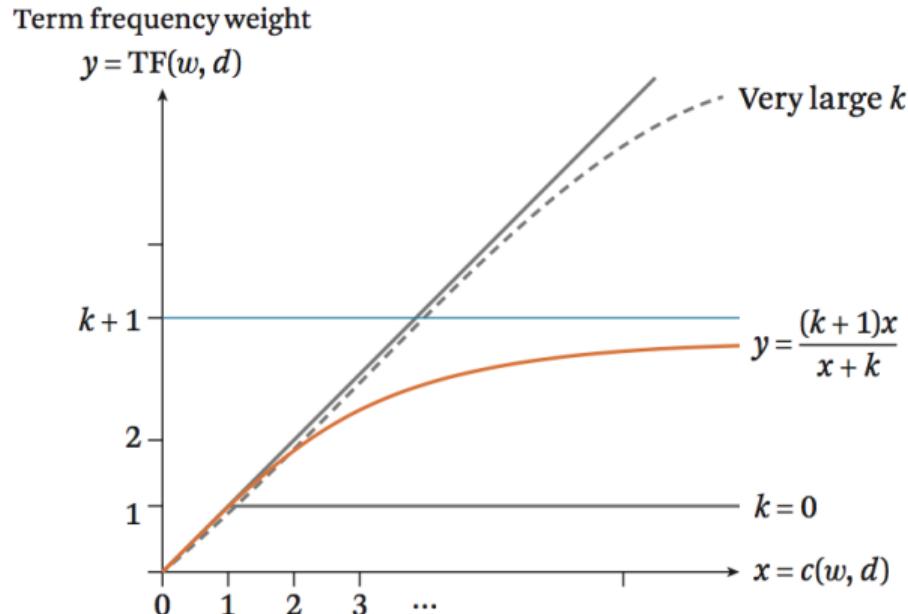


Figure: Illustration is taken from (Zhai&Massung, 2016)[Fig. 6.15]

BM25 document length normalization

- Idea: penalize long documents w.r.t. average document length (which serves as pivot)

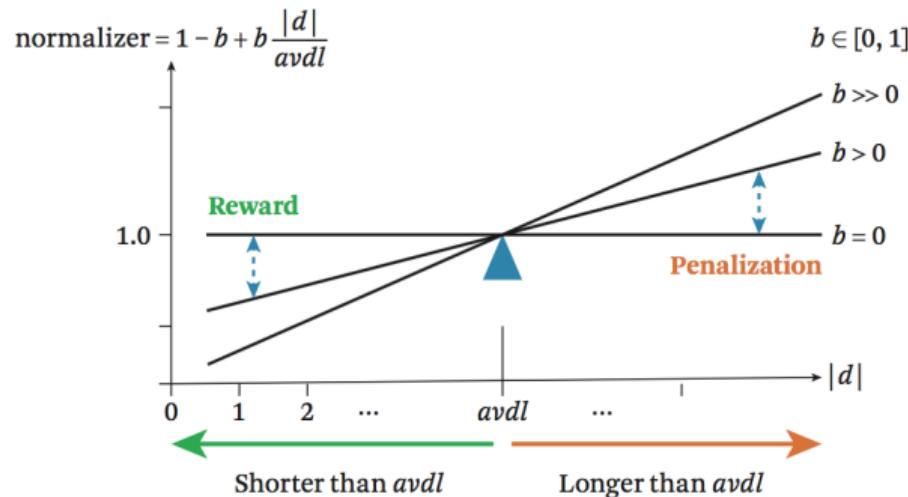


Figure: Illustration is taken from (Zhai&Massung, 2016)[Fig. 6.17]

BM25 parameter setting

- k_1 : calibrating term frequency scaling
 - 0 corresponds to a binary model
 - large values correspond to using raw term frequencies
 - typical values are between 1.2 and 2.0; a common default value is 1.2
- b : document length normalization
 - 0: no normalization at all
 - 1: full length normalization
 - typical value: 0.75

Retrieval models

- Vector space model
- Term weighting
- Vector space models (cont'd)
- Language models
- Summary

Language models

- Based on the notion of probabilities and processes for generating text
- Wide range of usage across different applications
 - Speech recognition
 - “I ate a cherry” is a more likely sentence than “Eye eight uh Jerry”
 - OCR and handwriting recognition
 - More probable sentences are more likely correct readings
 - Machine translation
 - More likely sentences are probably better translations

Language models for ranking documents

- Represent each document as a multinomial probability distribution over terms
- Estimate the probability that the query was “generated” by the given document
 - How likely is the search query given the language model of the document?

Query likelihood retrieval model

- Rank documents d according to their likelihood of being relevant given a query q :

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

- Query likelihood:** Probability that query q was “produced” by document d

$$P(q|d) = \prod_{t \in q} P(t|\theta_d)^{c_{t,q}}$$

- Document prior, $P(d)$:** Probability of the document being relevant to *any* query

Query likelihood

$$P(q|d) = \prod_{t \in q} P(t|\theta_d)^{c_{t,q}}$$

- θ_d is the document language model
 - Multinomial probability distribution over the vocabulary of terms
- $c_{t,q}$ is the raw frequency of term t in the query
- **Smoothing:** ensuring that $P(t|\theta_d)$ is > 0 for all terms

Jelinek-Mercer smoothing

- Linear interpolation between the empirical document model and a collection (background) language model

$$P(t|\theta_d) = (1 - \lambda)P(t|d) + \lambda P(t|C)$$

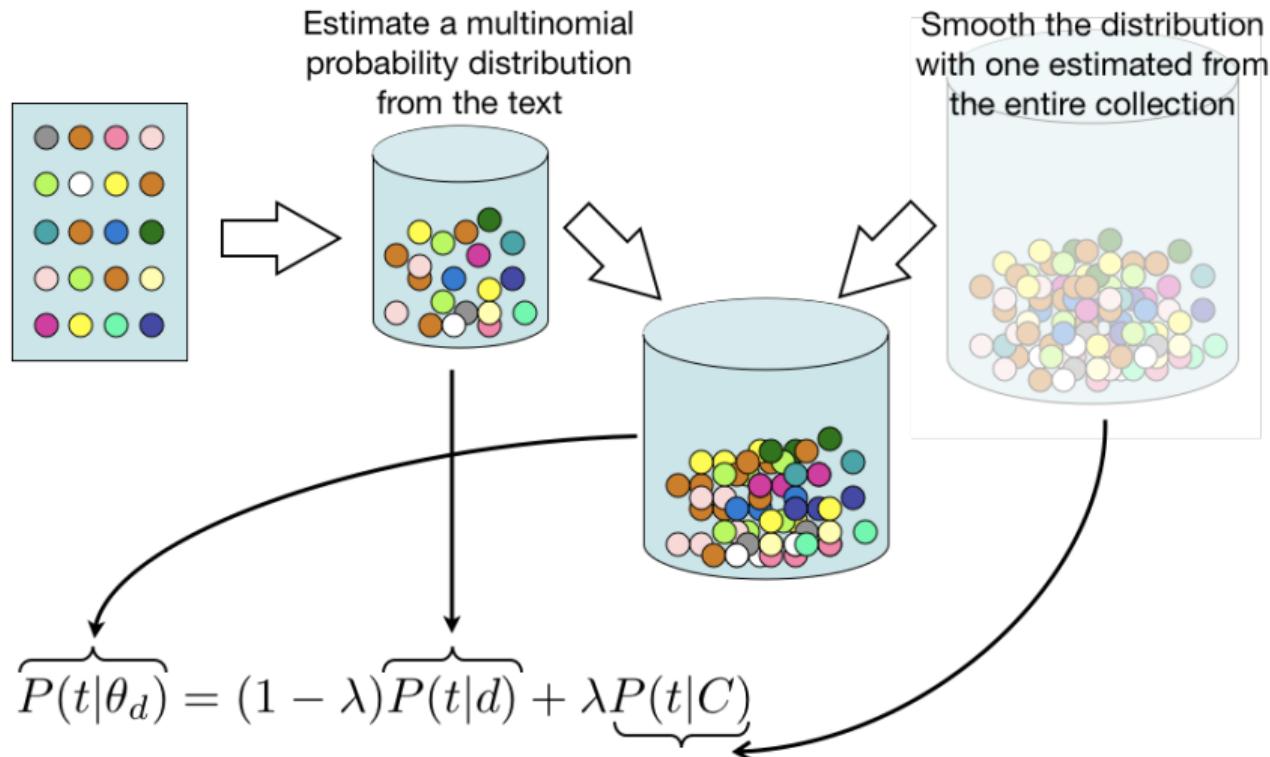
- $\lambda \in [0, 1]$ is the smoothing parameter
- Empirical document model (maximum likelihood estimate):

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

- Collection (background) language model (maximum likelihood estimate):

$$P(t|C) = \frac{\sum_{d'} c_{t,d'}}{\sum_{d'} |d'|}$$

Jelinek-Mercer smoothing



Dirichlet smoothing

- Smoothing is inversely proportional to the document length

$$P(t|\theta_d) = \frac{c_{t,d} + \mu P(t|C)}{|d| + \mu}$$

- μ is the smoothing parameter (typically ranges from 10 to 10000)
- Notice that Dirichlet smoothing may also be viewed as a linear interpolation in the style of Jelinek-Mercer smoothing, by setting

$$\lambda = \frac{\mu}{|d| + \mu} \qquad (1 - \lambda) = \frac{|d|}{|d| + \mu}$$

Query likelihood scoring (Example)

- query: “sea submarine”

$$\begin{aligned} P(q|d) &= P(\text{sea}|\theta_d) \times P(\text{submarine}|\theta_d) \\ &= ((1 - \lambda)P(\text{sea}|d) + \lambda P(\text{sea}|C)) \\ &\quad \times ((1 - \lambda)P(\text{submarine}|d) + \lambda P(\text{submarine}|C)) \end{aligned}$$

- where
 - $P(\text{sea}|d)$ is the relative frequency of term “sea” in document d
 - $P(\text{sea}|C)$ is the relative frequency of term “sea” in the entire collection
 - ...

Practical considerations

- Since we are multiplying small probabilities, it is better to perform computations in the log space

$$\begin{aligned} P(q|d) &= \prod_{t \in q} P(t|\theta_d)^{c_{t,q}} \\ &\Downarrow \\ \log P(q|d) &= \sum_{t \in q} c_{t,q} \times \log P(t|\theta_d) \end{aligned}$$

- Notice that it is a particular instantiation of our general scoring function $score(d, q) = \sum_{t \in q} w_{t,d} \times w_{t,q}$ by setting
 - $w_{t,d} = \log P(t|\theta_d)$
 - $w_{t,q} = c_{t,q}$

Retrieval models

- Vector space model
- Term weighting
- Vector space models (cont'd)
- Language models
- Summary

- Retrieval model is based on the idea of query-document similarity. Three main components:
 - Term frequency
 - Inverse document frequency
 - Document length normalization
- Retrieval function

$$score(d, q) = \sum_{t \in q} \frac{c_{t,d} \times (1 + k_1)}{c_{t,d} + k_1(1 - b + b \frac{|d|}{avgdl})} \times idf_t$$

- Parameters
 - k_1 : calibrating term frequency scaling ($k_1 \in [1.2..2]$)
 - b : document length normalization ($b \in [0, 1]$)

Language models

- Retrieval model is based on the probability of observing the query given that document
- Log query likelihood scoring

$$score(d, q) = \log P(q|d) = \sum_{t \in q} \log P(t|\theta_d) \times c_{t,q}$$

- Jelinek-Mercer smoothing

$$score(d, q) = \sum_{t \in q} \log \left((1 - \lambda) \frac{c_{t,d}}{|d|} + \lambda P(t|C) \right) \times c_{t,q}$$

- Dirichlet smoothing

$$score(d, q) = \sum_{t \in q} \log \frac{c_{t,d} + \mu P(t|C)}{|d| + \mu} \times c_{t,q}$$

Question

What other statistics are needed to compute these retrieval functions, in addition to term frequencies ($c_{t,d}$)?

- Total number of documents in the collection (for IDF computation) (int)
- Document length for each document (dictionary)
- Average document length in the collection (int)
- (optionally pre-computed) IDF score for each term (dictionary)

Language models

- Document length for each document (dictionary)
- Sum TF for each term (dictionary)
- Sum of all document lengths in the collection (int)
- (optionally pre-computed) Collection term probability $P(t|C)$ for each term (dictionary)

Exercise

E3-5 Vector space retrieval

Summary

- The problem of information retrieval and core issues (information needs, relevance, evaluation)
- Main components of search engines; indexing and querying processes
- Inverted index (posting list, posting)
- Index creation (without payload, with counts, with position information)
- Query processing (term-at-a-time and document-at-a-time scoring)
- General scoring function
- Vector space model, TF-IDF and BM25 retrieval models
- Language models (query likelihood scoring, different smoothing methods)

Reading

- Text Data Management and Analysis (Zhai&Massung)
 - Chapter 5: Sections 5.3, 5.4
 - Chapter 6
 - Chapter 8: Sections 8.2, 8.3 (optionally, 8.5, 8.6)
 - Chapter 10: Section 10.1 (optionally, Section 10.2)

Advanced Retrieval Models

[DAT640] Information Retrieval and Text Mining

Ivica Kostric & Krisztian Balog

University of Stavanger

August 31, 2022



CC BY 4.0

In this module

1. Fielded retrieval models
2. Query modeling
3. Web search
4. Learning to rank

Fielded retrieval models

Fielded retrieval models

- Quick recap of BM25 and LM
- Fielded extensions of BM25 and LM

BM25 scoring

BM25:

$$score(d, q) = \sum_{t \in q} \frac{c_{t,d} \times (1 + k_1)}{c_{t,d} + k_1(1 - b + b \frac{|d|}{avgdl})} \times idf_t$$

- Parameters
 - k_1 : calibrating term frequency scaling
 - b : document length normalization
- Note: several slight variations of BM25 exist!

Language Models

Query likelihood scoring:

$$P(q|d) = \prod_{t \in q} P(t|\theta_d)^{c_{t,q}}$$

- θ_d is the document language model
 - Multinomial probability distribution over the vocabulary of terms
- $c_{t,q}$ is the raw frequency of term t in the query
- **Smoothing:** ensuring that $P(t|\theta_d)$ is > 0 for all terms

Smoothing

- Jelinek-Mercer smoothing:

$$P(t|\theta_d) = (1 - \lambda)P(t|d) + \lambda P(t|C)$$

- $\lambda \in [0, 1]$ is the smoothing parameter
- Empirical document model (maximum likelihood estimate):

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

- Collection (background) language model (maximum likelihood estimate):

$$P(t|C) = \frac{\sum_{d'} c_{t,d'}}{\sum_{d'} |d'|}$$

- Dirichlet smoothing:

$$P(t|\theta_d) = \frac{c_{t,d} + \mu P(t|C)}{|d| + \mu}$$

- μ is the smoothing parameter (typically ranges from 10 to 10000)

Fielded retrieval models

- Quick recap of BM25 and LM
- Fielded extensions of BM25 and LM

Motivation

- Documents are composed of multiple fields
 - E.g., title, body, anchors, etc.
- Modeling internal document structure may be beneficial for retrieval

Example

The screenshot shows the PROMISE website with a purple header. The header features the PROMISE logo (a red square with yellow bars), the text "PROMISE Participative Research Observatory for Multimedia and Multilingual Information Systems Evaluation", the European Union flag, and a "Log-in" button. A navigation bar below the header includes links for Overview, Achievements, Use cases, Publications, Events (which is highlighted in black), CLEF, Media Center, and Contacts. A search bar is also present.

The main content area displays the title "PROMISE Winter School 2013" and the location "Bressanone, Italy" over a background image of snow-covered mountains. Below this, there are two columns: one for "Winter School 2013" listing Programme, Lecturers, Venue, Registration and Accommodation, Sponsor and Patronage, and Flyer; and another for "PROMISE Winter School 2013" detailing the theme "Bridging between Information Retrieval and Databases", the date "Bressanone, Italy 4 - 8 February 2013", and a descriptive paragraph about the school's aim to provide a grounding in core topics for information access and retrieval.

PROMISE Winter School 2013
Bressanone, Italy 4 - 8 February 2013

The aim of the PROMISE Winter School 2013 on "Bridging between Information Retrieval and Databases" is to give participants a grounding in the core topics that constitute the multidisciplinary area of information access and retrieval to unstructured, semistructured, and structured information. The school is a week-long event consisting of guest lectures from invited speakers who are recognized experts in the field. The school is intended for PhD students, Masters students or senior researchers such as post-doctoral researchers from the fields of databases, information retrieval, and related fields.

Unstructured representation

PROMISE Winter School 2013

Bridging between Information Retrieval and Databases

Bressanone, Italy 4 - 8 February 2013

The aim of the PROMISE Winter School 2013 on "Bridging between Information Retrieval and Databases" is to give participants a grounding in the core topics that constitute the multidisciplinary area of information access and retrieval to unstructured, semistructured, and structured information. The school is a week-long event consisting of guest lectures from invited speakers who are recognized experts in the field. The school is intended for PhD students, Masters students or senior researchers such as post-doctoral researchers from the fields of databases, information retrieval, and related fields. [...]

Example

```
<html>
<head>
    <title>Winter School 2013</title>
    <meta name="keywords" content="PROMISE, school, PhD, IR, DB, [...]" />
    <meta name="description" content="PROMISE Winter School 2013, [...]" />
</head>
<body>
    <h1>PROMISE Winter School 2013</h1>
    <h2>Bridging between Information Retrieval and Databases</h2>
    <h3>Bressanone, Italy 4 - 8 February 2013</h3>
    <p>The aim of the PROMISE Winter School 2013 on "Bridging between Information Retrieval and Databases" is to give participants a grounding in the core topics that constitute the multidisciplinary area of information access and retrieval to unstructured, semistructured, and structured information. The school is a week-long event consisting of guest lectures from invited speakers who are recognized experts in the field. The school is intended for PhD students, Masters students or senior researchers such as post-doctoral researchers from the fields of databases, information retrieval, and related fields. </p>
    [...]
</body>
</html>
```

The screenshot shows a website layout for the PROMISE Winter School 2013. The header features a blue banner with the text 'PROMISE Winter School 2013' and 'Bridging between Information Retrieval and Databases'. Below the banner, the text 'Bressanone, Italy 4 - 8 February 2013' is displayed. The main content area is divided into two columns. The left column, titled 'Winter School 2013', contains a navigation menu with links to 'Programme', 'Lecturers', 'Venue', 'Registration and Accommodation', 'Sponsor and Patronage', and 'Flyer'. The right column, titled 'PROMISE Winter School 2013', contains the same descriptive text as the code above. At the bottom of the page, there is a section titled 'Important Dates' with the note 'Registration Deadline (extended): 20th'.

Fielded representation (based on HTML markup)

<i>d</i> ₁ : title	Winter School 2013
<i>d</i> ₂ : meta	PROMISE, school, PhD, IR, DB, [...] PROMISE Winter School 2013, [...]
<i>d</i> ₃ : headings	PROMISE Winter School 2013 Bridging between Information Retrieval and Databases Bressanone, Italy 4-8 February 2013
<i>d</i> ₄ : body	The aim of the PROMISE Winter School 2013 on "Bridging between Information Retrieval and Databases" is to give participants a grounding in the core topics that constitute the multidisciplinary area of information access and retrieval to unstructured, semistructured, and structured information. The school is a week-long event consisting of guest lectures from invited speakers who are recognized experts in the field. The school is intended for PhD students, Masters students or senior researchers such as postdoctoral researchers from the fields of databases, information retrieval, and related fields. [...]

Fielded extension of retrieval models

- BM25 \Rightarrow BM25F
- Language Models (LM) \Rightarrow Mixture of Language Models (MLM)

- Extension of BM25 incorporating multiple fields
- The soft normalization and term frequencies need to be adjusted
- Original BM25 retrieval function:

$$score(d, q) = \sum_{t \in q} \frac{c_{t,d} \times (1 + k_1)}{c_{t,d} + k_1 \times B} \times idf_t$$

- where B is the soft normalization:

$$B = (1 - b + b \frac{|d|}{avgdl})$$

- Replace term frequencies $c_{t,d}$ with *pseudo term frequencies* $\tilde{c}_{t,d}$
- BM25F retrieval function:

$$\text{score}(d, q) = \sum_{t \in q} \frac{\tilde{c}_{t,d}}{k_1 + \tilde{c}_{t,d}} \times idf_t$$

- Pseudo term frequency calculation

$$\tilde{c}_{t,d} = \sum_i w_i \times \frac{c_{t,d_i}}{B_i}$$

- where
 - i corresponds to the field index
 - w_i is the field weight (such that $\sum_i w_i = 1$)
 - B_i is soft normalization for field i , where b_i becomes a field-specific parameter

$$B_i = (1 - b_i + b_i \frac{|d_i|}{avgdl_i})$$

Mixture of Language Models (MLM)

- Idea: Build a separate language model for each field, then take a linear combination of them

$$P(t|\theta_d) = \sum_i w_i P(t|\theta_{d_i})$$

- where
 - i corresponds to the field index
 - w_i is the field weight (such that $\sum_i w_i = 1$)
 - $P(t|\theta_{d_i})$ is the field language model

Field language model

- Smoothing goes analogously to document language models, but term statistics are restricted to the given field i
- Using Jelinek-Mercer smoothing:

$$P(t|\theta_{d_i}) = (1 - \lambda_i)P(t|d_i) + \lambda_i P(t|C_i)$$

- where both the empirical field model ($P(t|d_i)$) and the collection field model ($P(t|C_i)$) are maximum likelihood estimates:

$$P(t|d_i) = \frac{c_{t,d_i}}{|d_i|}$$

$$P(t|d_i) = \frac{\sum_{d'} c_{t,d'_i}}{\sum_{d'} |d'_i|}$$

Setting parameter values

- Retrieval models often contain parameters that must be tuned to get the best performance for specific types of data and queries
- For experiments
 - Use training and test data sets
 - If less data available, use cross-validation by partitioning the data into k subsets
- Many techniques exist to find optimal parameter values given training data
 - Standard problem in machine learning
- For standard retrieval models, involving few parameters, *grid search* is feasible
 - Perform a sweep over the possible values of each parameter, e.g., from 0 to 1 in steps of 0.1

Query modeling

Query modeling based on feedback

- Take the results of a user's actions or previous search results to improve retrieval
- Often implemented as updates to a query, which then alters the list of documents
- Overall process is called **relevance feedback**, because we get feedback information about the relevance of documents
 - **Explicit feedback**: user provides relevance judgments on some documents
 - **Pseudo relevance feedback** (or *blind feedback*): we don't involve users but "blindly" assume that the top- k documents are relevant
 - **Implicit feedback**: infer relevance feedback from users' interactions with the search results (clickthroughs)

Feedback in an IR system

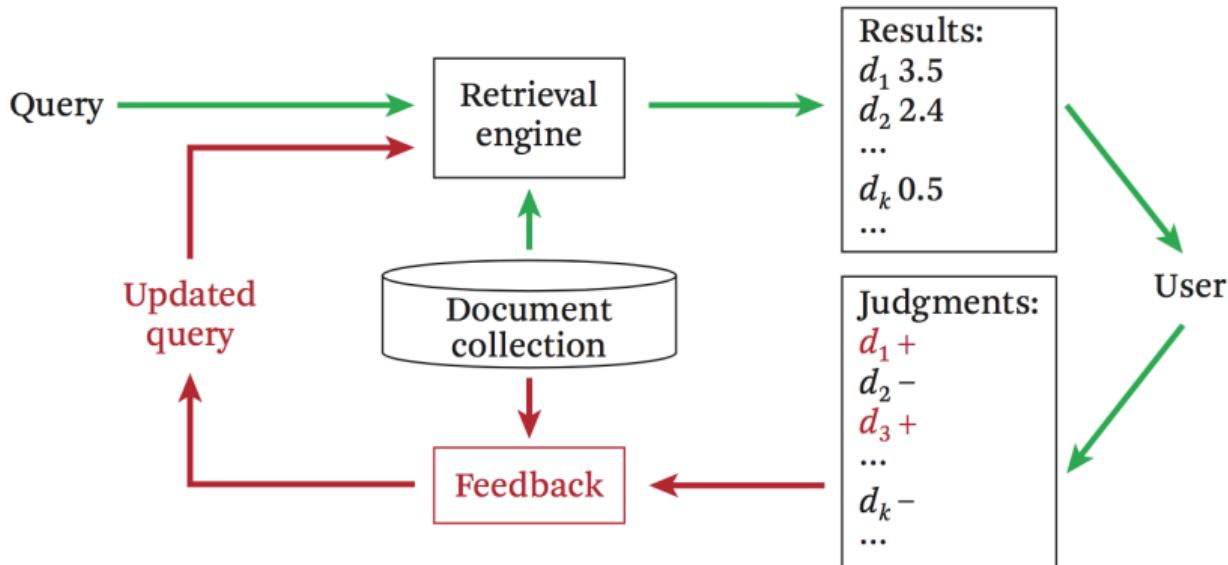


Figure: Illustration is taken from (Zhai&Massung, 2016)[Fig. 7.1]

Feedback in the Vector Space Model

- It is assumed that we have examples of relevant (D^+) and non-relevant (D^-) documents for a given query
- General idea: modify the query vector (adjust weight of existing terms and/or assign weight to new terms)
 - As a result, the query will usually have more terms, which is why this method is often called **query expansion**

Rocchio feedback

- Idea: adjust the weights in the query vector to move it closer to the cluster of relevant documents

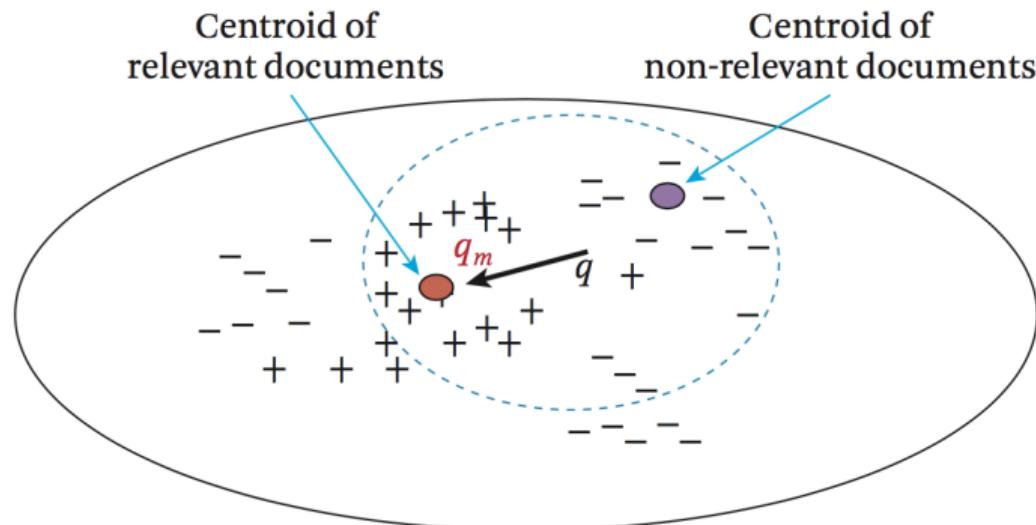


Figure: Illustration is taken from (Zhai&Massung, 2016)[Fig. 7.2]

Rocchio feedback

- Modified query vector:

$$\vec{q}_m = \alpha \vec{q} + \frac{\beta}{|D^+|} \sum_{d \in D^+} \vec{d} - \frac{\gamma}{|D^-|} \sum_{d \in D^-} \vec{d}$$

- \vec{q} : original query vector
- D^+, D^- : set of relevant and non-relevant feedback documents
- α, β, γ : parameters that control the movement of the original vector
- The second and third terms of the equation correspond to the centroid of relevant and non-relevant documents, respectively

Practical considerations

- Modifying all the weights in the query (and then using them all for scoring documents) is computationally heavy
 - Often, only terms with the highest weights are retained
- Non-relevant examples tend not to be very useful
 - Sometimes negative examples are not used at all, or γ is set to a small value

Exercise

E4-1 Rocchio feedback

Feedback in Language Models

- We generalize the query likelihood function to allow us to include feedback information more easily
- (Log) query likelihood

$$\log P(q|d) \propto \sum_{t \in q} c_{t,q} \times \log P(t|\theta_d)$$

- Generalize $c_{t,q}$ to a query model $P(t|\theta_q)$

$$\log P(q|d) \propto \sum_{t \in q} P(t|\theta_q) \times \log P(t|\theta_d)$$

- Often referred to as **KL-divergence** retrieval, because it provides the same ranking as minimizing the Kullback-Leibler divergence between the query model θ_q and the document model θ_d
- Using a maximum likelihood query model this is rank-equivalent to query likelihood scoring

Query models

- Maximum likelihood estimate (original query)

$$P_{ML}(t|\theta_q) = \frac{c_{t,q}}{|q|}$$

- I.e., the relative frequency of the term in the query
- Linear interpolation with a feedback query model $\hat{\theta}_q$

$$P(t|\theta_q) = \alpha P_{ML}(t|\theta_q) + (1 - \alpha)P(t|\hat{\theta}_q)$$

- α has the same interpretation as in the Rocchio feedback model, i.e., how much we rely on the original query

Relevance models

- **Relevance models** are a theoretically sound and effective way of estimating feedback query models
- Main idea: consider other terms that co-occur with the original query terms in the set of feedback documents \hat{D}
 - Commonly taken to be the set of top- k documents ($k=10$ or 20) retrieved using the original query with query likelihood scoring
- Two variants with different independence assumptions
- Relevance model 1
 - Assume full independence between the original query terms and the expansion terms:

$$P_{RM1}(t|\hat{\theta}_q) \approx \sum_{d \in \hat{D}} P(d) P(t|\theta_d) \prod_{t' \in q} P(t'|\theta_d)$$

- Often referred to as $RM3$ when linearly combined with the original query

Relevance models

- Relevance model 2
 - The original query terms $t' \in q$ are still assumed to be independent of each other, but they are dependent on the expansion term t :

$$P_{RM2}(t|\hat{\theta}_q) \approx P(t) \prod_{t' \in q} \sum_{d \in \hat{D}} P(t'|\theta_d) P(d|t)$$

- where $P(d|t)$ is computed as

$$P(d|t) = \frac{P(t|\theta_d)P(d)}{P(t)} = \frac{P(t|\theta_d)P(d)}{\sum_{d' \in \hat{D}} P(t|\theta_{d'})P(d')}$$

Illustration

t	$P_{ML}(t \theta_q)$	t	$P(t \theta_q)$
machine	0.5000	vision	0.2796
vision	0.5000	machine	0.2762
		image	0.0248
		vehicles	0.0224
		safe	0.0220
		cam	0.0214
		traffic	0.0178
		technology	0.0176
		camera	0.0173
		object	0.0147

Table: Baseline (left) and expanded (right) query models for the query *machine vision*; only the top 10 terms are shown.

Feedback summary

- Overall goal is to get a richer representation of the user's underlying information need by enriching/refining the initial query
- Interpolation with the original query is important
- Relevance feedback is computationally expensive! Number of feedback terms and expansion terms are typically limited (10..50) for efficiency considerations
- Queries may be hurt by relevance feedback ("query drift")

Web search

Web search

- Before the Web: search was small scale, usually focused on libraries
- Web search is a major application that everyone cares about
- Challenges
 - Scalability (users as well as content)
 - Ensure high-quality results (fighting SPAM)
 - Dynamic nature (constantly changing content)

Some specific techniques

- Crawling
 - Freshness
 - Focused crawling
 - Deep Web crawling
- Indexing
 - Distributed indexing
- **Retrieval** ⇐
 - Link analysis

Deep (or hidden) Web

- Much larger than the “conventional” Web
- Three broad categories:
 - Private sites
 - No incoming links, or may require log in with a valid account
 - Form results
 - Sites that can be reached only after entering some data into a form
 - Scripted pages
 - Pages that use JavaScript, Flash, or another client-side language to generate links

Question

How to make content on the Deep Web searchable (indexable)?

Surfacing the Deep Web

- Pre-compute all interesting form submissions for each HTML form
- Each form submission corresponds to a distinct URL
- Add URLs for each form submission into search engine index

Link analysis

- Links are a key component of the Web
- Important for navigation, but also for search

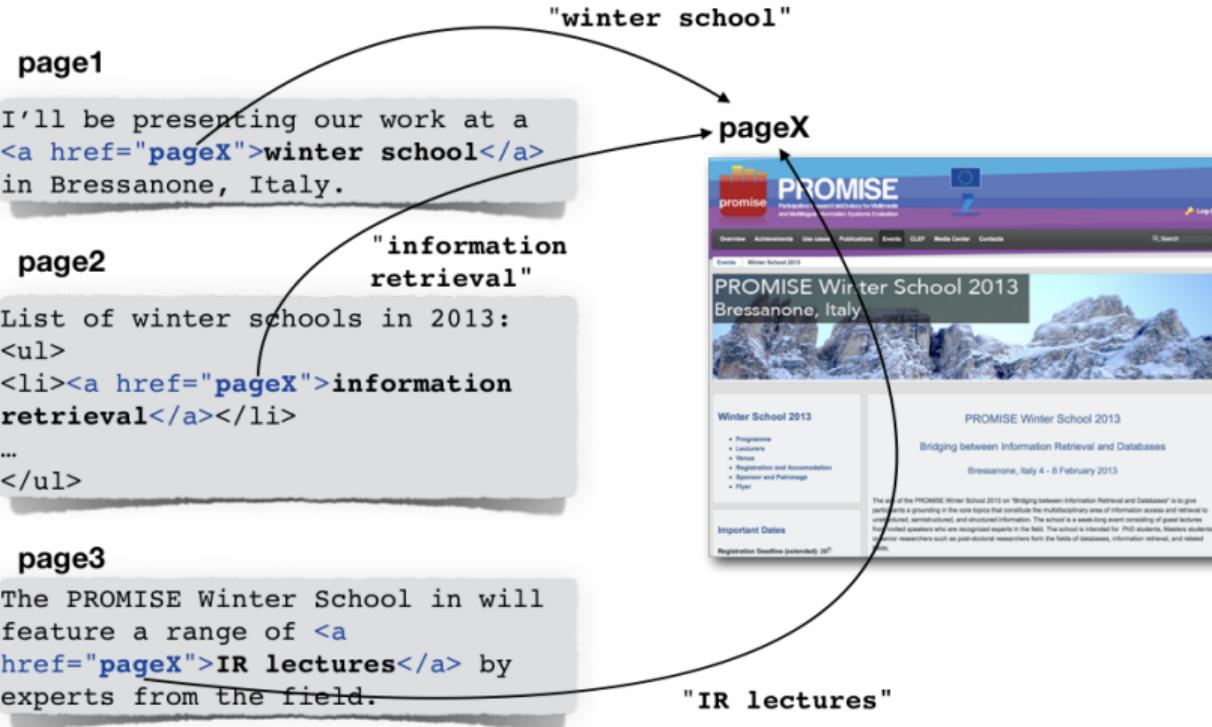


- Both anchor text and links are used by search engines

Anchor text

- Aggregated from all incoming links and added as a separate document field
- Tends to be short, descriptive, and similar to query text
 - Can be thought of a description of the page “written by others”
- Has a significant impact on effectiveness for *some types of queries*

Example



Fielded document representation

title	Winter School 2013
meta	PROMISE, school, PhD, IR, DB, [...] PROMISE Winter School 2013, [...]
headings	PROMISE Winter School 2013 Bridging between Information Retrieval and Databases Bressanone, Italy 4-8 February 2013
body	The aim of the PROMISE Winter School 2013 on "Bridging between Information Retrieval and Databases" is to give participants a grounding in the core topics that constitute the multidisciplinary area of information access and retrieval to unstructured, semistructured, and structured information. The school is a week-long event consisting of guest lectures from invited speakers who are recognized experts in the field. The school is intended for PhD students, Masters students or senior researchers such as postdoctoral researchers from the fields of databases, information retrieval, and related fields. [...]
anchors	winter school information retrieval IR lectures

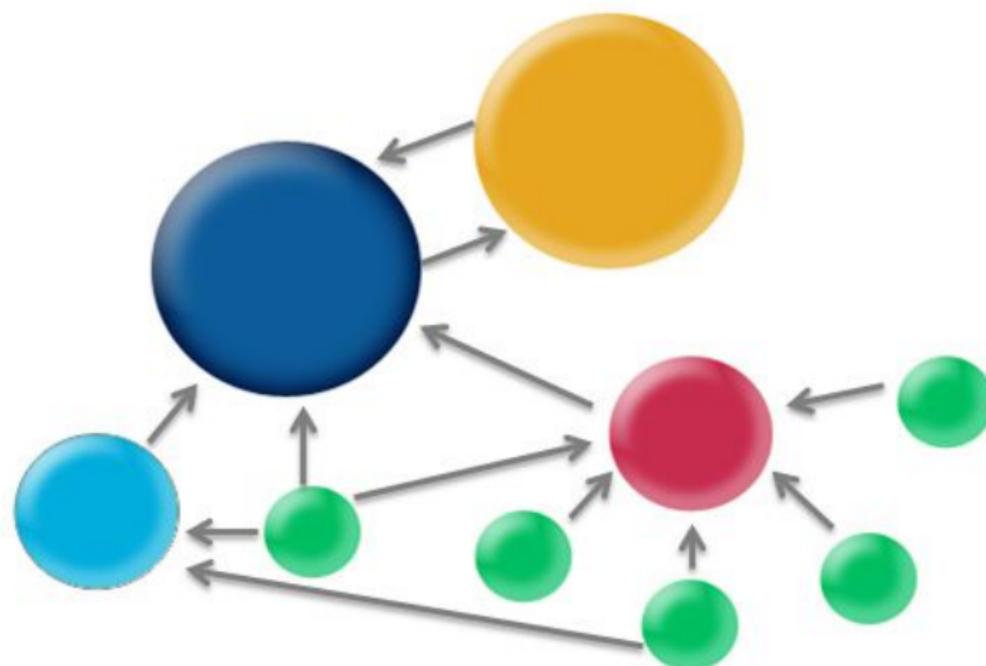
Document importance on the Web

- What are web pages that are popular and useful to *many* people?
- Use the links between web pages as a way to measure popularity
- The most obvious measure is to count the number of *inlinks*
 - Quite effective, but very susceptible to SPAM

PageRank

- Algorithm to rank web pages by popularity
- Proposed by Google founders Sergey Brin and Larry Page in 1998
- Main idea: **A web page is important if it is pointed to by other important web pages**
- PageRank is a numeric value that represents the importance of a web page
 - When one page links to another page, it is effectively casting a vote for the other page
 - More votes implies more importance
 - Importance of each vote is taken into account when a page's PageRank is calculated

Illustration



Source: <https://www.shoutmeloud.com/how-to-calculate-pagerank-google-seo.html>

Random Surfer Model

- PageRank simulates a user navigating on the Web randomly as follows
- The user is currently at page a
 - She moves to one of the pages linked from a with probability $1 - q$
 - She jumps to a random web page with probability q
 - This is to ensure that the user doesn't "get stuck" on any given page (i.e., on a page with no outlinks)
- Repeat the process for the page she moved to
- The PageRank score of a page is the average probability of the random surfer visiting that page

PageRank formula

$$PR(a) = \frac{q}{T} + (1 - q) \sum_{i=1}^n \frac{PR(p_i)}{L(p_i)}$$

Jump to a random page with this probability (q is typically set to 0.15)

Follow one of the hyperlinks in the current page with this probability

PageRank value of page p_i

Number of outgoing links of page p_i

PageRank of page a

Total number of pages in the Web graph

page a is pointed by pages $p_1 \dots p_n$

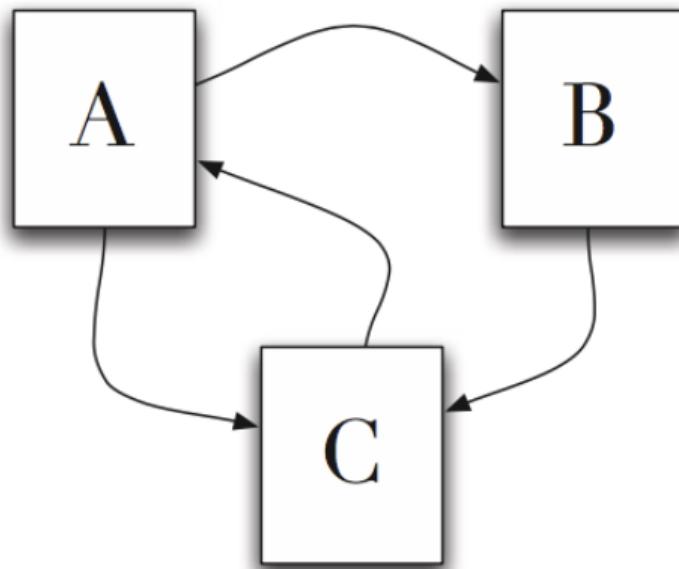
The diagram illustrates the PageRank formula $PR(a) = \frac{q}{T} + (1 - q) \sum_{i=1}^n \frac{PR(p_i)}{L(p_i)}$. It shows the formula at the top with annotations pointing to its parts. An annotation for q points to the term $\frac{q}{T}$. Another annotation for T points to the term $\sum_{i=1}^n$. Annotations for $1 - q$ and n point to the term $(1 - q) \sum_{i=1}^n$. Annotations for $PR(p_i)$ and $L(p_i)$ point to the terms $PR(p_i)$ and $L(p_i)$ respectively. To the left of the formula, an annotation for "PageRank of page a" points to the variable a . Below the formula, an annotation for "Total number of pages in the Web graph" points to the variable T . To the right of the formula, an annotation for "page a is pointed by pages $p_1 \dots p_n$ " points to the variable i . Above the formula, two annotations are present: one for "Jump to a random page with this probability (q is typically set to 0.15)" pointing to the term $\frac{q}{T}$, and another for "Follow one of the hyperlinks in the current page with this probability" pointing to the term $(1 - q) \sum_{i=1}^n$. Annotations for "PageRank value of page p_i " and "Number of outgoing links of page p_i " point to the variables p_i and $L(p_i)$ respectively.

Technical issues

- This is a recursive formula. PageRank values need to be computed iteratively
 - We don't know the PageRank values at start. We can assume equal values ($1/T$)
- Number of iterations?
 - Good approximation already after a small number of iterations; stop when change in absolute values is below a given threshold

Example #1

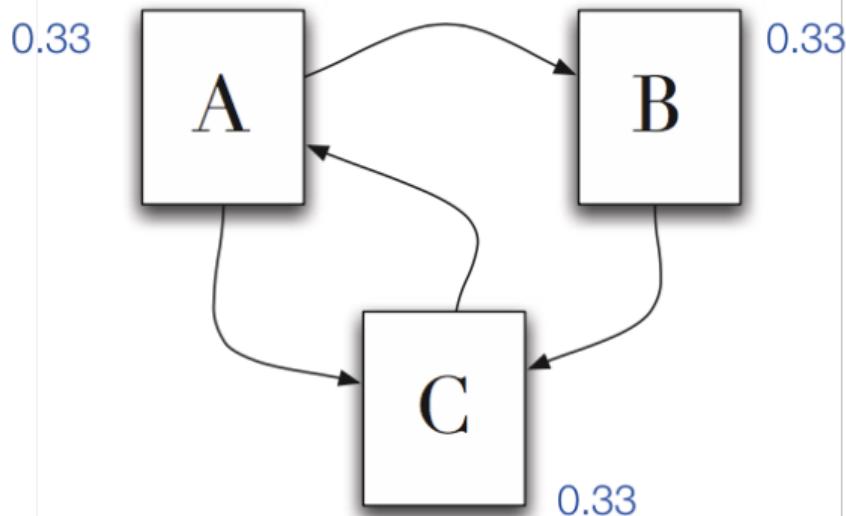
q=0
(no random jumps)



Example #1

Iteration 0: assume that the PageRank values are the same for all pages

q=0
(no random jumps)

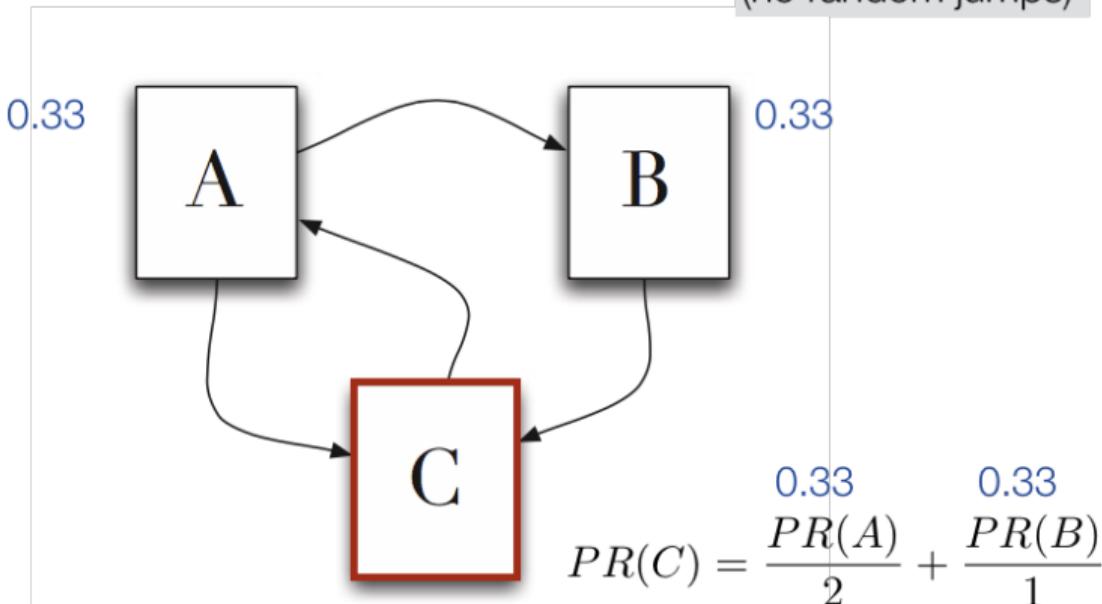


Example #1

Iteration 1

q=0

(no random jumps)



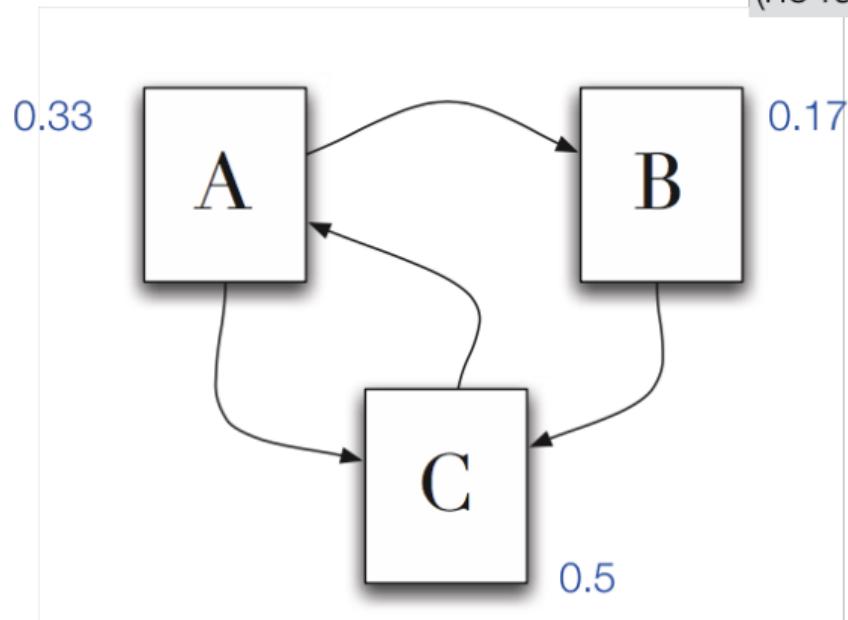
PageRank of C depends on the
PageRank values of A and B

=0.5

Example #1

at the end of **Iteration 1**

q=0
(no random jumps)

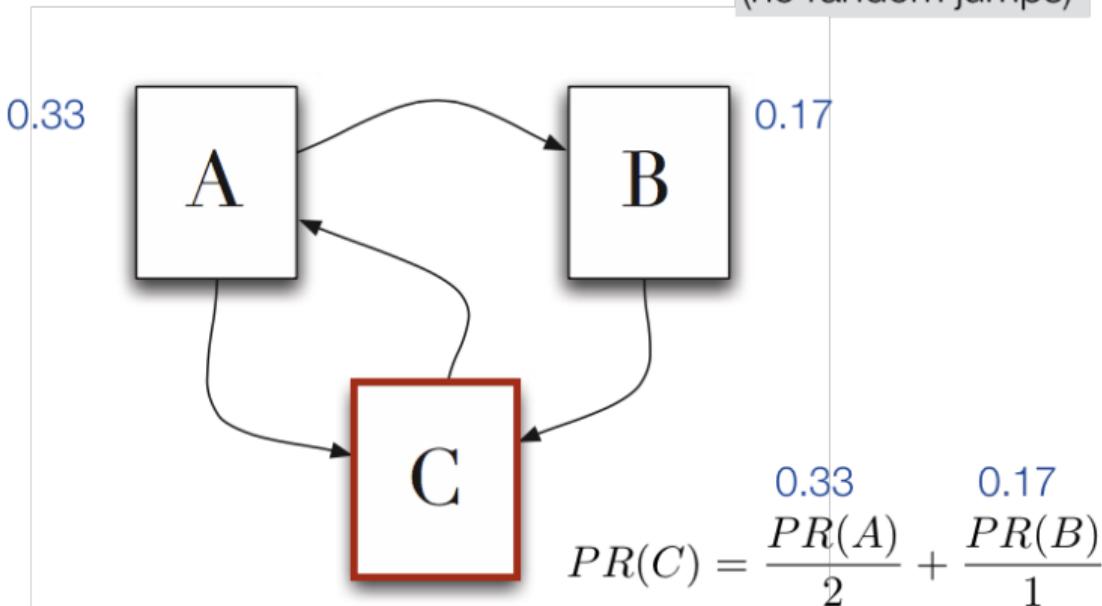


Example #1

Iteration 2

q=0

(no random jumps)



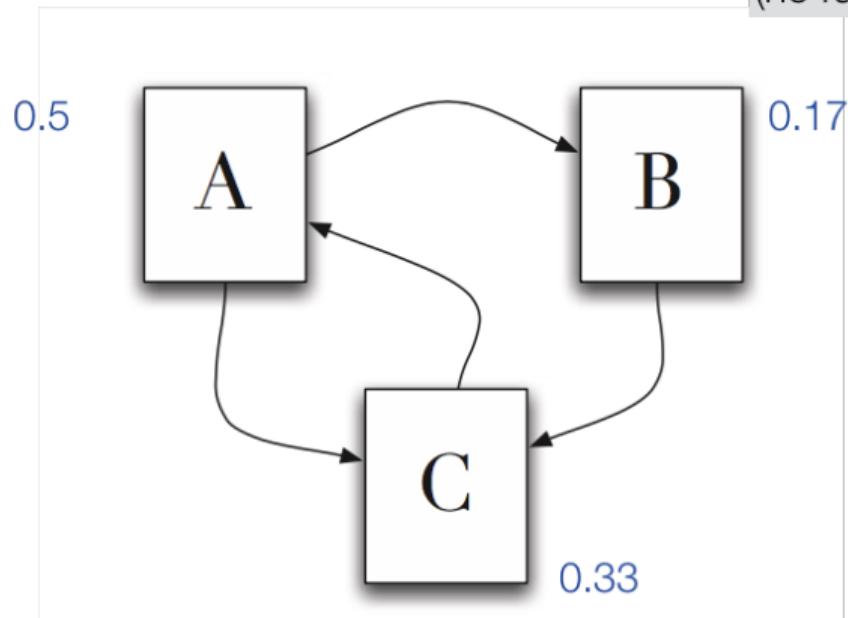
PageRank of C depends on the
PageRank values of A and B

=0.33

Example #1

at the end of **Iteration 2**

q=0
(no random jumps)

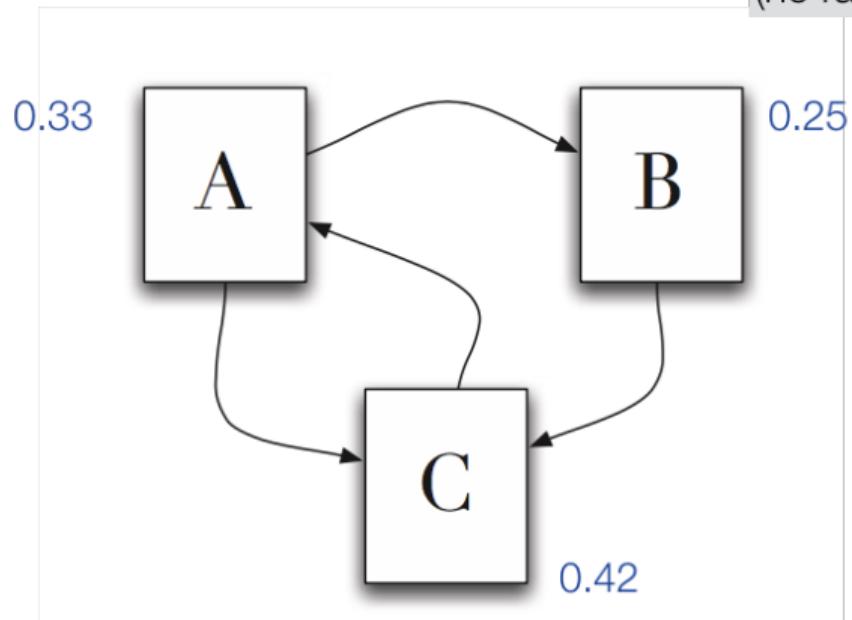


Example #1

at the end of **Iteration 3**

q=0

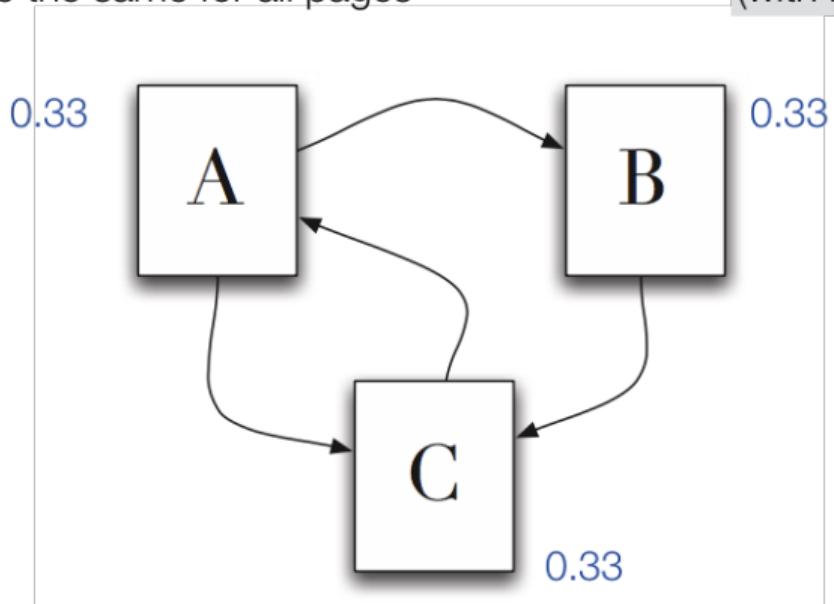
(no random jumps)



Example #2

Iteration 0: assume that the PageRank values are the same for all pages

q=0.2
(with random jumps)

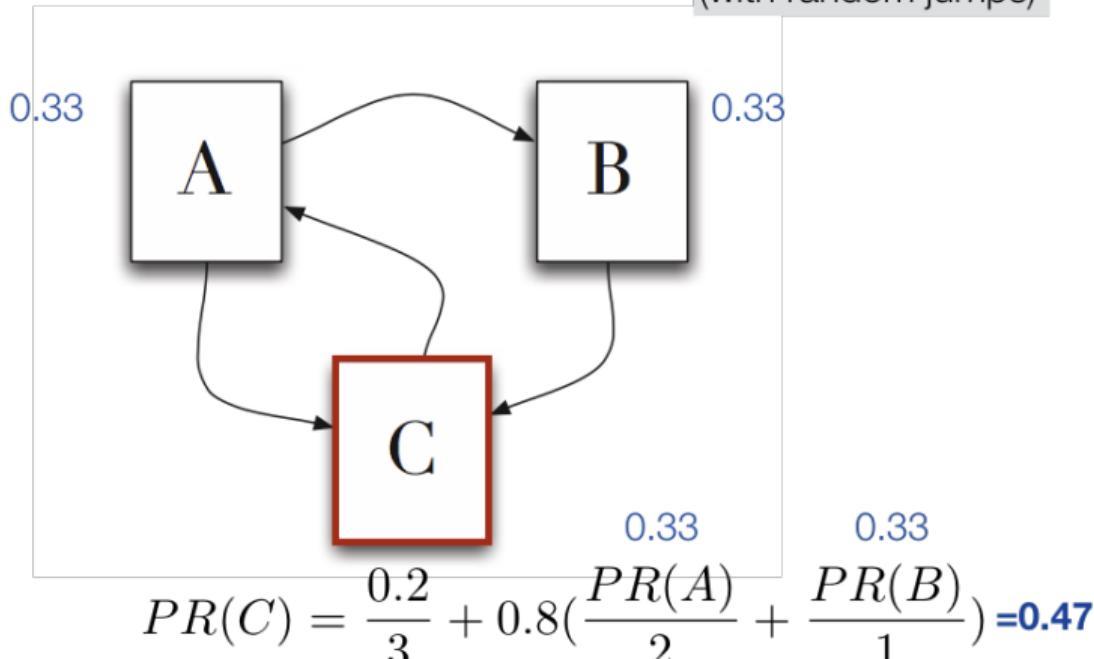


Example #2

Iteration 1

q=0.2

(with random jumps)

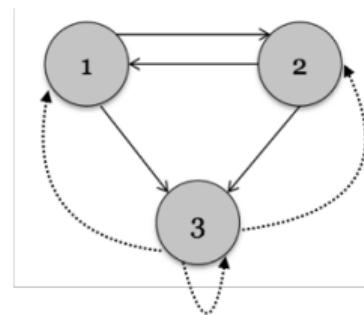
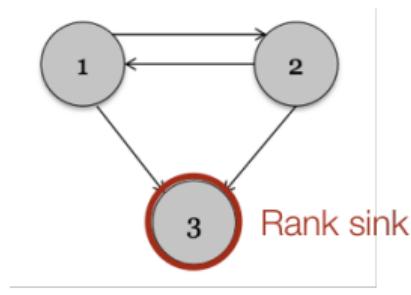


Question

How are PageRank scores affected by pages that do not have any outgoing links?

Dealing with “rank sinks”

- How to handle *rank sinks* (“dead ends”), i.e., pages that have no outlinks?
- Assume that it links to all other pages in the collection (including itself) when computing PageRank scores



Online PageRank checkers

Google pagerank checker

All Images News Maps Videos More Settings Tools

About 3,450,000 results (0.47 seconds)

 <https://checkpagerank.net> ▾

Check Page Rank - Check Your PageRank Free!

Page Rank Checker and Domain Analysis Tool. Free tool reports PageRank and other important SEO statistics about your website. Fake Page Rank Detection!

[What is PageRank?](#) · [PageRank is BACK!!!](#) · [PageRank Blog](#) · [Business Directory](#)

 <https://www.prchecker.info> › check_page_rank

Google PageRank Checker - Check Google page rank instantly

Page Rank Checker is a completely free service to check Google pagerank instantly using our online page rank check tool or a small pagerank button.

 <https://www.prchecker.info> ▾

Google PageRank Checker - Check Google page rank of any ...

Page Rank Checker is a completely Free tool to check Google PR, page rank of your web site easily and possibly display your Google PageRank on your web ...

 <https://dnschecker.org> › pagerank

Page Rank Checker - Check Your Website Pagerank

With Pagerank Checker, you can check the page rank of your website. Just enter domain and check what is the current pagerank of your website.

PageRank summary

- Important example of query-independent document ranking
 - Web pages with high PageRank are preferred
- It is, however, not as important as conventional wisdom holds
 - Just one of the many features a modern web search engine uses
 - It tends to have the most impact on popular queries

Exercise

E4-2 PageRank

Incorporating document importance (e.g., PageRank)

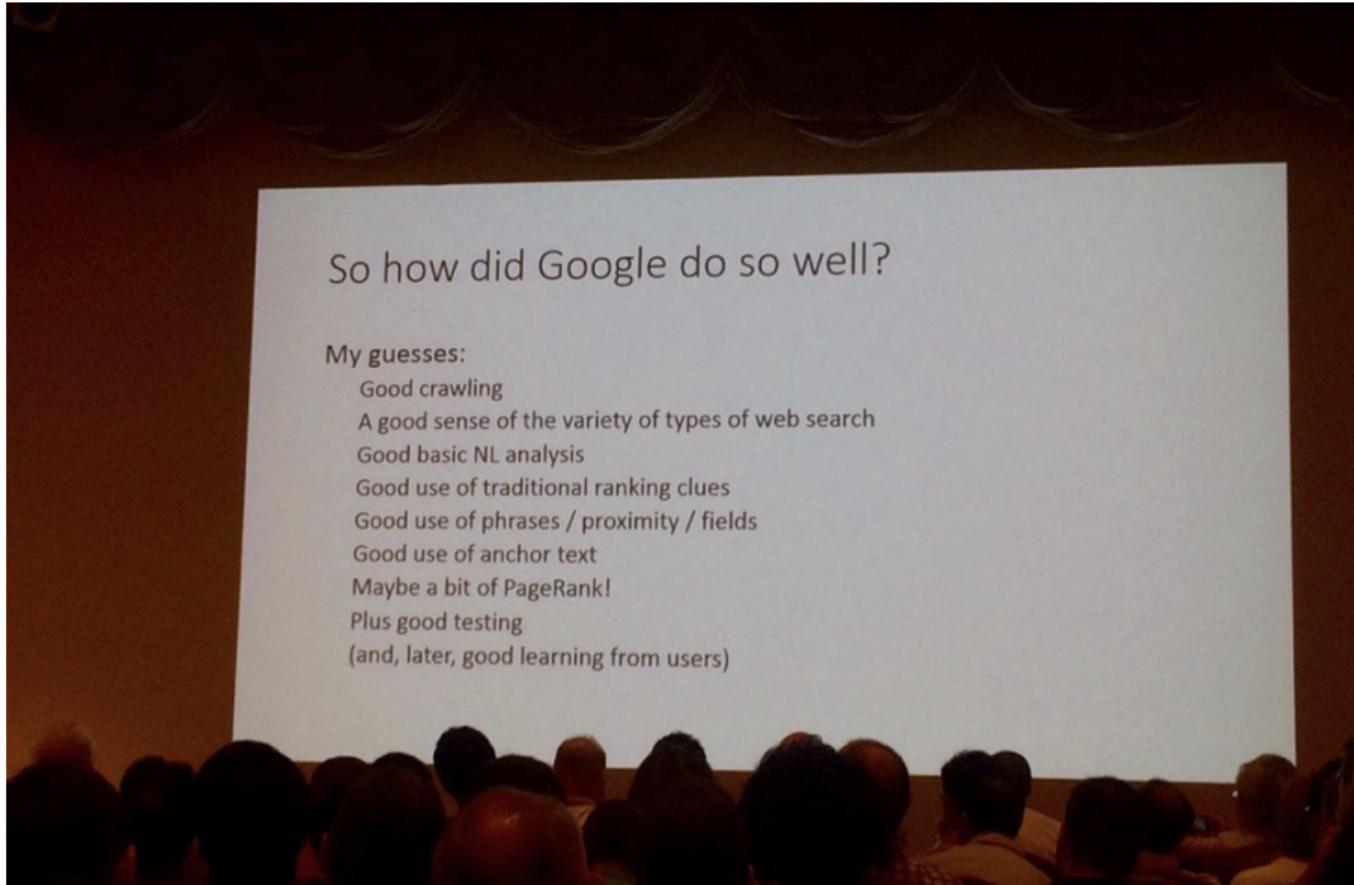
- How to incorporate document importance into the ranking?
- As a query-independent (“static”) score component

$$score'(d, q) = score(d, q) \times score(d)$$

- In case of Language Models, document importance is encoded as the document prior $P(d)$

$$P(d|q) \propto P(q|d)P(d)$$

Stephen Robertson, SIGIR'17 keynote



Question

What is search engine optimization (SEO)?

Learning to rank

Learning to rank

- Motivation
- Learning to rank methods
- Practical considerations

Recap

- Classical retrieval models
 - Vector space model, BM25, LM
- Three main components
 - Term frequency
 - How many times query terms appear in the document
 - Document length
 - Any term is expected to occur more frequently in long document; account for differences in document length
 - Document frequency
 - How often the term appears in the entire collection

Additional factors

- So far: content-based matching
- Many additional signals, e.g.,
 - Document quality
 - PageRank
 - SPAM score
 - ...
 - Implicit (click-based) feedback
 - How many times users clicked on a document given a query?
 - How many times this particular user clicked on a document given the query?
 - ...
 - ...

Question

How to combine all these clues for ranking?

Learning to rank

- Motivation
- Learning to rank methods
- Practical considerations

Machine learning for IR

- We hypothesize that the probability of relevance is related to some combination of *features*
 - Each feature is a clue or signal that can help determine relevance
- We employ machine learning to learn an “optimal” combination of features, based on training data
 - There may be several hundred features; impossible to tune by hand
 - Training data is (item, query, relevance) triples
- Modern systems (especially on the Web) use a great number of features
 - In 2008, Google was using over 200 features¹

¹The New York Times (2008-06-03)

Some example features

- Log frequency of query word in anchor text
- Query word in color on page?
- #images on page
- #outlinks on page
- PageRank
- URL length
- URL contains “~”?
- Page length
- ...

Simple example

- We assume that the relevance of a document is related to a linear combination of all the features:

$$\log \frac{P(R = 1|q, d)}{1 - P(R = 1|q, d)} = \beta_0 + \sum_{i=1}^n \beta_i x_i$$

- x_i is the value of the i^{th} feature
 - β_i is the weight of the i^{th} feature
- This leads to the following probability of relevance:

$$P(R = 1|q, d) = \frac{1}{1 + \exp\{-\beta_0 - \sum_{i=1}^n \beta_i x_i\}}$$

- This *logistic regression* method gives us an estimate in $[0, 1]$

Learning to Rank (LTR)

- Learn a function automatically to rank items (documents) effectively
 - Training data: (item, query, relevance) triples
 - Output: ranking function $h(q, d)$
- Three main groups of approaches
 - Pointwise
 - Pairwise
 - Listwise

Pointwise LTR

- Specifying whether a document is relevant (binary) or specifying a degree of relevance
 - Classification: Predict a categorical (unordered) output value (relevant or not)
 - Regression: Predict an ordered or continuous output value (degree of relevance) ↵
- All the standard classification/regression algorithms can be directly used
- Note: classical retrieval models are also point-wise: $score(q, d)$

Pairwise LTR

- The learning function is based on a pair of items
 - Given two documents, classify which of the two should be ranked at a higher position
 - I.e., learning relative preference
- E.g., Ranking SVM, LambdaMART, RankNet

Listwise LTR

- The ranking function is based on a ranked list of items
 - Given two ranked list of the same items, which is better?
- Directly optimizes a retrieval metric
 - Need a loss function on a list of documents
 - Can get fairly complex compared to pointwise or pairwise approaches
- Challenge is scale: huge number of potential lists
- E.g., AdaRank, ListNet

How to?

- Develop a feature set
 - The most important step!
 - Usually problem dependent
- Choose a good ranking algorithm
 - E.g., Random Forests work well for pairwise LTR
- Training, validation, and testing
 - Similar to standard machine learning applications

Features for document retrieval

- Query features
 - Depend only on the query
- Document features
 - Depend only on the document
- Query-document features
 - Express the degree of matching between the query and the document

Query features

- Query length (number of terms)
- Sum of IDF scores of query terms in a given field (title, content, anchors, etc.)
- Total number of matching documents
- Number of named entities in the query
- ...

Document features

- Length of each document field (title, content, anchors, etc.)
- PageRank score
- Number of inlinks
- Number of outlinks
- Number of slash in URL
- Length of URL
- ...

Query-document features

- Retrieval score of a given document field (e.g., BM25, LM, TF-IDF)
- Sum of TF scores of query terms in a given document field (title, content, anchors, URL, etc)
- Retrieval score of the entire document (e.g., BM25F, MLM)
- ...

Learning to rank

- Motivation
- Learning to rank methods
- Practical considerations

Feature normalization

- Feature values are often normalized to be in the $[0, 1]$ range *for a given query*
 - Esp. matching features that may be on different scales across queries because of query length
- Min-max normalization:

$$\tilde{x}_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

- x_1, \dots, x_n : original values for a given feature
- \tilde{x}_i : normalized value for the i^{th} instance

Class imbalance and computation cost

- Many more non-relevant than relevant instances
- Classifiers usually do not handle huge imbalance well
- Also, it is not feasible to extract features for all documents in the corpus
- Sampling is needed!

Two-stage ranking pipeline

- Implemented as a re-ranking mechanism (two-step retrieval)
 - Step 1 (initial ranking): Retrieve top-N ($N=100$ or 1000) candidate documents using a baseline approach (e.g., BM25). (This, essentially, is *document sampling*.)
 - Step 2 (re-ranking): Create feature vectors and re-rank these top-N candidates to arrive at the final ranking
- Often, candidate documents from first-pass retrieval and labeled (judged) documents are combined together for learning a model
 - Retrieved but not judged documents are assumed to be non-relevant
- Feature computation
 - Document features may be computed offline
 - Query and query-document features are computed online (at query time)
 - Avoid using too many expensive features!

Summary

- Fielded extensions of retrieval models (BM25F, MLM)
- Types of relevance feedback (explicit, implicit, pseudo/blind)
- Query expansion in the vector space model (Rocchio feedback)
- Query expansion in a language modeling setting (Relevance Models)
- Utilizing links on the Web: anchor text and link structure
- PageRank algorithm
- Learning ranking functions (learning to rank)
- Main classes of learning-to-rank approaches (pointwise, pairwise, listwise)
- Types of features (query, document, query-document)
- Practical considerations in learning-to-rank

Reading

- Text Data Management and Analysis (Zhai&Massung)
 - Chapter 7
 - Chapter 10: Sections 10.3, 10.4
- Relevance-Based Language Models (Lavrenko&Croft, 2001)
 - <https://sigir.org/wp-content/uploads/2017/06/p260.pdf>

Retrieval Evaluation

[DAT640] Information Retrieval and Text Mining

Ivica Kostric & Krisztian Balog

University of Stavanger

September 5, 2022



CC BY 4.0

In this module

1. Retrieval evaluation
2. Statistical significance testing

Retrieval evaluation

Evaluation



“To measure is to know.
If you can not measure it,
you can not improve it.”

—Lord Kelvin

What to measure?

- **Effectiveness** ⇐ our focus
 - How accurate are the search results?
 - I.e., the system's capability of ranking relevant documents ahead of non-relevant ones
- Efficiency
 - How quickly can a user get the results?
 - I.e., the response time of the system
- Usability
 - How useful is the system for real user tasks?

Evaluation in IR

- Search engine evaluation must rely on users!
- Core question: How can we get users involved?

Types of evaluation

- **Offline** (test collection based) ↛ our focus
- **Online** (live evaluation) ↛ our focus
- User studies
- Simulation of users
- ...

Retrieval evaluation

- Offline evaluation
- Online evaluation

Test collection based evaluation

- *Cranfield evaluation methodology*
- Basic idea: Build reusable test collections
- Ingredients of an IR test collection
 - Dataset (corpus of documents or *information objects*)
 - Test queries (set of *information needs*)
 - Relevance assessments
 - Evaluation measures

Relevance assessments

- Ground truth labels for query-item pairs
- **Binary**
 - 0: non-relevant
 - 1: relevant
- **Graded**, for example,
 - -1: spam / junk
 - 0: non-relevant
 - 1: somewhat relevant
 - 2: relevant
 - 3: highly relevant / perfect match

query 1	item 11	0
	item 12	1
	item 13	1
	item 14	0
	item 15	0
	...	
query 2	item 21	1
	item 22	1
	item 23	0
	...	

*ground truth with
binary assessments*

Obtaining relevance assessments

- Obtaining relevance judgments is an expensive, time-consuming process
 - Who does it?
 - What are the instructions?
 - What is the level of agreement?
- Two approaches
 - Expert judges
 - Crowdsourcing

Text Retrieval Conference (TREC)

- Organized by the US National Institute of Standards and Technology (NIST)
- Yearly benchmarking cycle
- Developing test collections for various information retrieval tasks
- Relevance judgments created by expert judges, i.e., retired information analysts (CIA)



Examples of TREC document collections

Name	#Documents	Size
CACM	3k	2.2 MB
AP	242k	0.7 GB
GOV2	25M	426 GB
ClueWeb09	1B	25 TB

TREC topic example

<top>

<num> Number: 794

<title> pet therapy

<desc> Description:

How are pets or animals used in therapy for humans and what are the benefits?

<narr> Narrative:

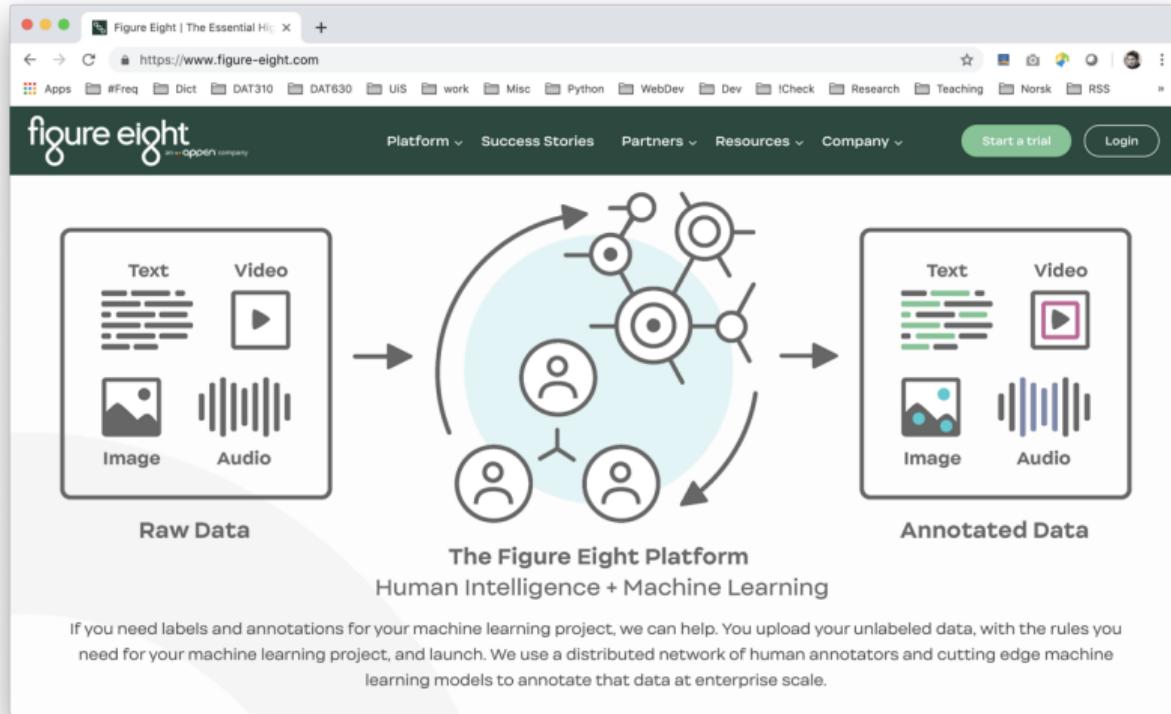
Relevant documents must include details of how pet- or animal-assisted therapy is or has been used. Relevant details include information about pet therapy programs, descriptions of the circumstances in which pet therapy is used, the benefits of this type of therapy, the degree of success of this therapy, and any laws or regulations governing it.

</top>

Crowdsourcing

- Obtain relevance judgments on a crowdsourcing platform
 - Often branded as “human intelligence platforms”
- “Microtasks” are performed in parallel by large, paid crowds





Example microtask

Query: button down shirt

[Click here to search on Google](#)

Result Title: Men's Essential Poplin Button-down Shirt

Result Image:



A photograph of a man from the waist up, facing slightly to his left. He is wearing a black long-sleeved button-down shirt and light-colored khaki trousers. He is standing against a plain white background.

[more colors](#)

[Click here to look at the result page](#)

Rate how well '**Men's Essential Poplin Button-down Shirt**' matches the query (required)

Irrelevant	Somewhat relevant	Relevant	Perfect Match
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Other search related annotation tasks

Read the text below. Pay close attention to detail.

I am trying to upgrade my old phone to an iPhone 6s and I noticed the deal on pre-owned iPhone 6s for \$72 off.

Which attribute is the text about? (n)

- Account
- Billing
- Equipment
- Payment
- Network
- Plan
- Purchase
- Refund and Escalation
- Web Assistance
- Other

Please pick the category that relates to the best or the sentence:

Step 1: Review the Assorted Garments



Is this a jumpsuit? (required)
 Yes, it's a jumpsuit.
 No.

What is the PATTERN of the jumpsuit? (required)
 Checkered - consistent repeating pattern (e.g. plaid)
 Stripes - consistent colored patterns, not texture or material of garment
 Geometric/Solid Other
 Dots - repeated round spots, not texture or material of garment

What is the SLEEVE LENGTH? (required)
 Short
 Medium
 Long
 Sleeveless
 Bare
 Long

Be sure to keep the intended answer in mind (i.e. kids long sleeves generally look shorter than the body of the garment compared to adults). If the garment is a tank top with thin spaghetti straps, when does NOT answer.

Check here if the Image is Broken:

Step 2: Categorize the Assorted Garments

Search Classes

Domain Name	Scientific Name	Conservation Status
The dog	Canis lupus familiaris	The modern wolves are not closely related to the wolves that were first domesticated – when dogs that are the direct ancestor of the dog is related

Context

The domestic dog (Canis lupus familiaris) when considered a subspecies of the wolf or Canis lupus when considered a distinct species is a member of the family Canidae. The domestic dog is one of the most common and widespread domestic animals. The dog and the wild gray wolf are sister taxa as modern wolves are not closely related to the ancient wolf that was first domesticated, which probably took place around 15,000 years ago. The dog has been bred by humans for thousands of years specifically for certain behaviors, sensory capabilities, and physical attributes. Their long association with humans has led to a unique bond between the two. Dogs can be domesticated and have been selectively bred over millennia for various behaviors, sensory capabilities, and physical attributes. Their long association with humans has led to a unique bond between the two. Dogs can be trained to perform a variety of tasks that would be inadequate for other canid species. Dogs vary widely in shape, size and colors. They perform many roles for humans, such as hunting, herding, pulling loads, acting as personal assistants, providing police and military, companionship and, more recently, acting disabled people and therapeutic roles. This influence on human society has given them the nickname of "man's best friend".

Annotations

- Clicking with Once a span is selected, apply an annotation by clicking on one of the classes to the left.
- Hold click and drag across spans to select multiple spans.
- Double click a span to invert it and all matching spans.
- To remove annotations from selected spans.
- To break up selected spans into smaller spans.
- To merge selected adjacent spans into larger spans.
- To break up the selected span in search engine.
- To merge to words.
- To merge to metrics.
- To expand/collapse filters.

Intent classification

Content categorization

Text annotation

Expert judges vs. crowdsourcing

- Expert judges
 - Each query-item pair is commonly assessed by a single person
 - Agreement is good because of “narrative”
- Crowdsourcing
 - Assessments are more noisy
 - Commonly, majority vote is taken
 - The number of labels collected for an item may be adjusted dynamically such that a majority decision is reached
- **Data is only as good as the guidelines!**

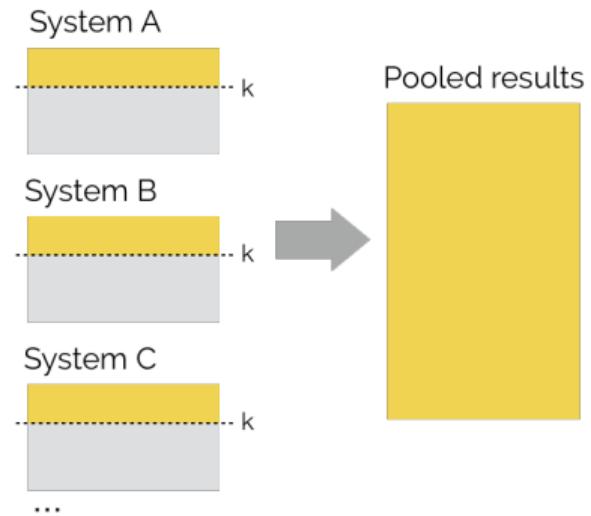
Discussion

Question

How can the relevance of all items be assessed in a large dataset for a given query?

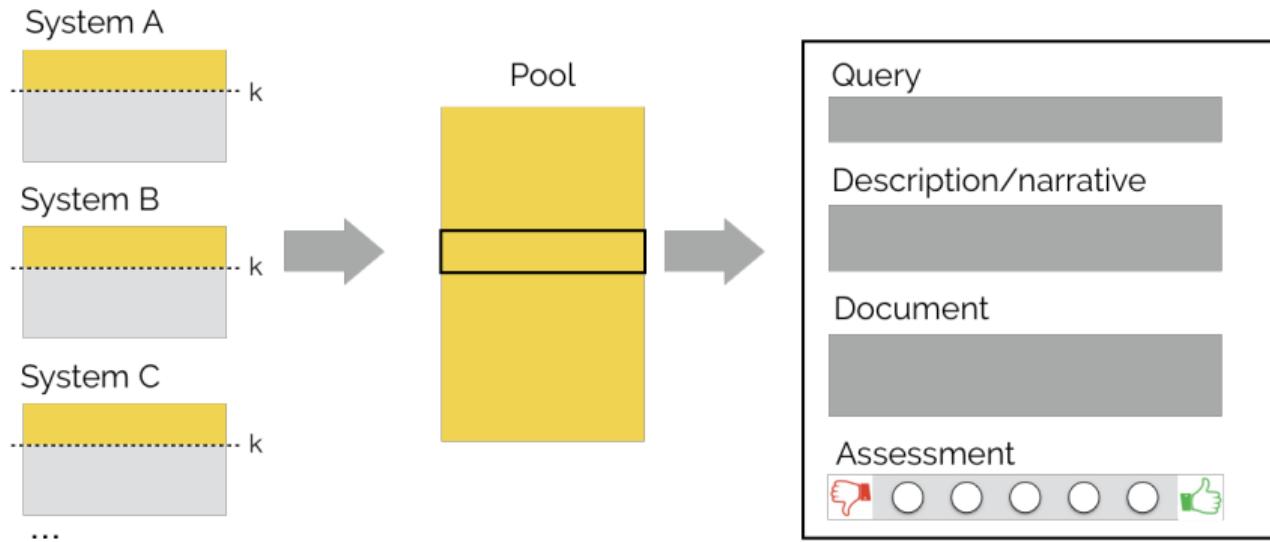
Pooling

- Exhaustive judgments for all documents in a collection is not practical
- Top- k results from different systems (algorithms) are merged into a pool
 - Duplicates are removed
 - Item order is randomized
- Produces a large number of relevance judgments for each query, although still incomplete
 - Not assessed items are assumed to be non-relevant



Pooling

- Relevance assessments are collected for all documents in the pool
 - Either using expert judges or crowd workers



Test collection based evaluation

- Ingredients of an IR test collection
 - ~~Dataset (corpus of documents or *information objects*)~~
 - ~~Test queries (set of *information needs*)~~
 - ~~Relevance assessments~~
 - **Evaluation measures**

IR evaluation measures

- Assessing the quality of a ranked list against the ground truth relevance labels
 - Commonly, a real number between 0 and 1
- **Important:** All measures are based on a (simplified) model of user needs and behavior
 - That is, the right measure depends on the particular task

Effectiveness measures

- A is the set of **relevant** documents
- B is the set of **retrieved** documents

	Relevant	Non-relevant
Retrieved	$ A \cap B $	$ \bar{A} \cap B $
Not retrieved	$ A \cap \bar{B} $	$ \bar{A} \cap \bar{B} $

Precision and recall analogously to before:

$$P = \frac{|A \cap B|}{|B|}$$

$$R = \frac{|A \cap B|}{|A|}$$

Discussion

Question

Precision and Recall are set-based metrics. How can we use them to evaluate ranked lists?

Evaluating rankings

Calculate recall and precision values at every rank position



= the relevant documents

Ranking #1



Recall 0.17 0.17 0.33 0.5 0.67 0.83 0.83 0.83 0.83 1.0

Precision 1.0 0.5 0.67 0.75 0.8 0.83 0.71 0.63 0.56 0.6

Ranking #2



Recall 0.0 0.17 0.17 0.17 0.33 0.5 0.67 0.67 0.83 1.0

Precision 0.0 0.5 0.33 0.25 0.4 0.5 0.57 0.5 0.56 0.6

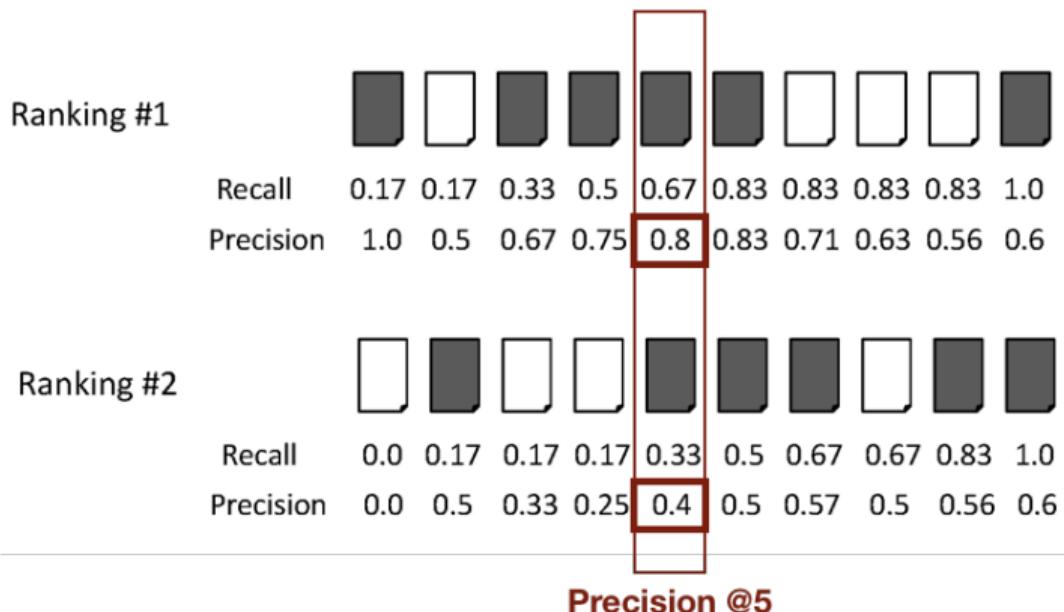
Evaluating rankings

- Calculating recall and precision values at every rank position produces a long list of numbers (see previous slide)
- Need to **summarize** the effectiveness of a ranking
- Various alternatives
 - Calculate recall and precision at fixed rank positions ($P@k$, $R@k$)
 - Calculate precision at standard recall levels, from 0.0 to 1.0 (requires interpolation)
 - Averaging the precision values from the rank positions where a relevant document was retrieved (AP)

Fixed rank positions

Compute precision/recall at a given rank position k ($P@k$, $R@k$)

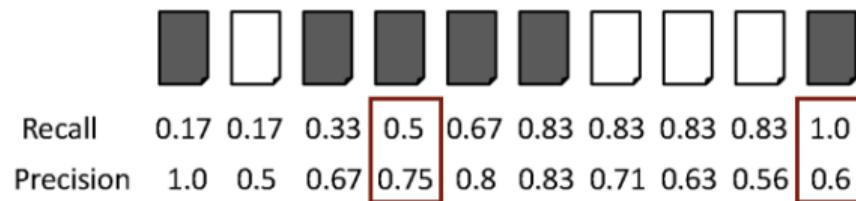
- This measure does not distinguish between differences in the rankings at positions 1 to k



Standard recall levels

Calculate precision at standard recall levels, from 0.0 to 1.0

- Each ranking is then represented using 11 numbers
- Values of precision at these standard recall levels are often not available, for example:



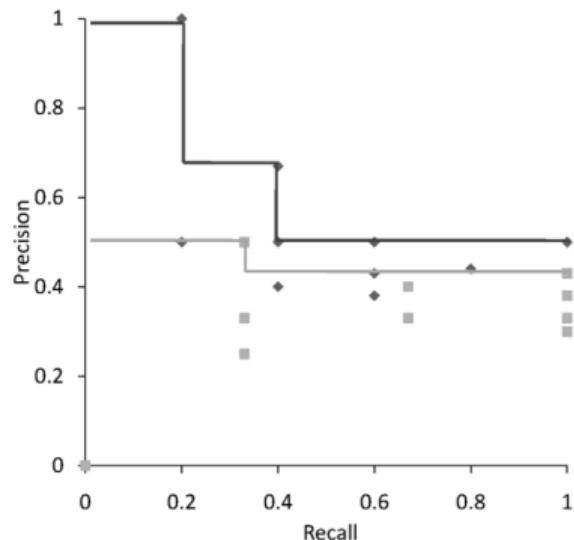
- *Interpolation* is needed

Interpolation

- To average graphs, calculate precision at standard recall levels:

$$P(R) = \max\{P' : R' \geq R \wedge (R', P') \in S\}$$

- where S is the set of observed (R, P) points
- Defines precision at any recall level as the maximum precision observed in any recall-precision point at a higher recall level
- Produces a step function



Average Precision

- Average the precision values from the rank positions where a relevant document was retrieved

$$AP = \frac{1}{|Rel|} \sum_{\substack{i=1, \dots, n \\ d_i \in Rel}} P(i) \rightarrow \text{Precision at rank i}$$

↓ ↓

Total number of relevant documents
According to the ground truth Only relevant documents contribute to the sum

- If a relevant document is not retrieved (in the top k ranks, e.g., $k = 1000$) then its contribution is 0.0
- AP is single number that is based on the ranking of all the relevant documents
- The value depends heavily on the highly ranked relevant documents

Average Precision



Ranking #1



Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	1.0
Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56

Ranking #2



Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

$$\text{Ranking } \#1: (1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6) / 6 = 0.78$$

$$\text{Ranking } \#2: (0.5 + 0.4 + 0.5 + 0.57 + 0.56 + 0.6) / 6 = 0.52$$

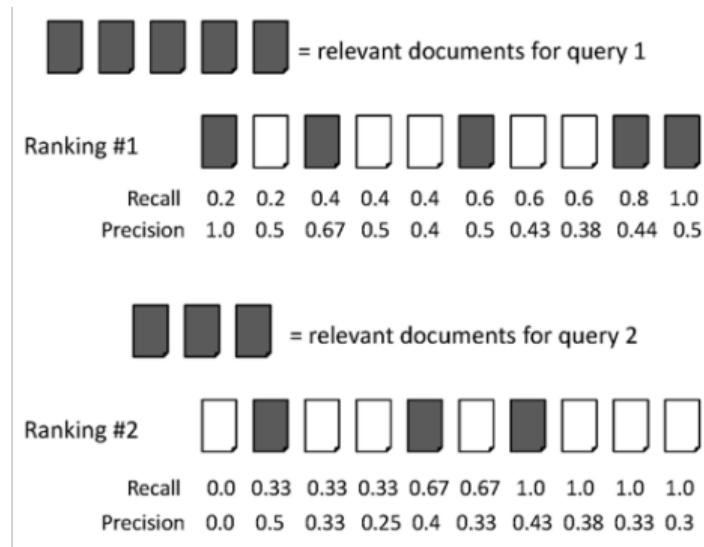
Averaging across queries

- So far: measuring ranking effectiveness on a **single query**
- Need: measure ranking effectiveness on a **set of queries**
- Average is computed over the set of queries

Mean Average Precision (MAP)

- Summarize rankings from multiple queries by averaging Average Precision
- Very succinct summary
- Most commonly used measure in research papers
- Assumes user is interested in finding many relevant documents for each query
- Requires many relevance judgments

Mean Average Precision



$$\text{average precision query 1} = (1.0 + 0.67 + 0.5 + 0.44 + 0.5)/5 = 0.62$$

$$\text{average precision query 2} = (0.5 + 0.4 + 0.43)/3 = 0.44$$

$$\text{mean average precision} = (0.62 + 0.44)/2 = 0.53$$

Focusing on top documents

- Users tend to look at only the top part of the ranked result list to find relevant documents
- Some search tasks have only one relevant document
 - E.g., navigational search, question answering
- Recall in those cases is not appropriate
 - Instead need to measure how well the search engine does at retrieving relevant documents at very high ranks

Focusing on top documents

- Precision at rank k (P@k)
 - k is typically 5, 10, 20
 - Easy to compute, average, understand
 - Not sensitive to rank positions less than k
- Reciprocal Rank (RR)
 - Reciprocal of the rank at which the first relevant document is retrieved
 - Mean Reciprocal Rank (MRR) is the average of the reciprocal ranks over a set of queries
 - Very sensitive to rank position

Mean Reciprocal Rank



= the relevant documents

Ranking #1



$$\text{Reciprocal rank (RR)} = 1/1 = 1.0$$

Ranking #2



$$\text{Reciprocal rank (RR)} = 1/2 = 0.5$$

$$\text{Mean reciprocal rank (MRR)} = (1.0 + 0.5) / 2 = 0.75$$

Exercise

E5-1 Evaluation measures

Exercise

E5-2 Interpolated precision

Graded relevance

- So far: relevance in binary
- What about graded relevance levels?

Discounted Cumulative Gain

- Popular measure for evaluating web search and related tasks
- Two assumptions:
 - Highly relevant documents are more useful than marginally relevant document
 - 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)

- DCG is the total gain accumulated at a particular rank p :

$$DCG_p = rel_1 + \sum_{i=2}^p \frac{rel_i}{\log_2 i}$$

- rel_i is the graded relevance level of the item retrieved at rank i
- Gain is accumulated starting at the top of the ranking and discounted by $1/\log(\text{rank})$
 - E.g., discount at rank 4 is $1/2$, and at rank 8 it is $1/3$
- Average over the set of test queries
- Note: search engine companies have their own (secret) variants

Discounted Cumulative Gain



Rank (i)	1	2	3	4	5	6	7	8	9	10
Gain	3	2	3	0	0	1	2	2	3	0
Discounted gain	3	2/1	3/1.59	0	0	1/2.59	2/2.81	2/3	3/3.17	0
Discounted cumulative gain (DCG@i)	3	5	6.89	6.89	6.89	7.28	7.99	8.66	9.61	9.61

How good is a DCG@10 value of 9.61?

Normalized Discounted Cumulative Gain (NDCG)

- DCG values are often normalized by comparing the DCG at each rank with the DCG value for the perfect (ideal) ranking
 - I.e., divide $DCG@i$ value with the ideal DCG value at rank i
 - Yields value between 0 and 1

<i>Ideal ranking</i>										
Rank (i)	1	2	3	4	5	6	7	8	9	10
Gain	3	3	3	2	2	2	1	0	0	0
Discounted cumulative gain ($DCG@i$)	3	6	7.89	8.89	9.75	10.52	10.88	10.88	10.88	10.88

Exercise

E5-3 NDCG

Retrieval evaluation

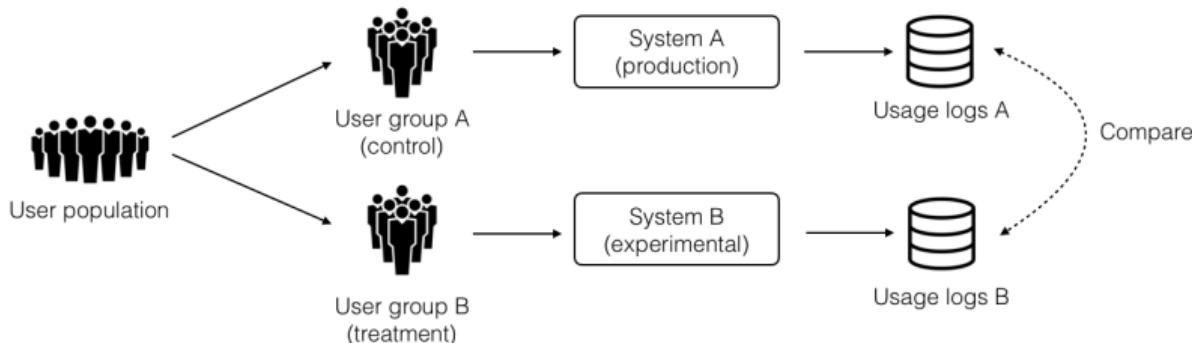
- Offline evaluation
- Online evaluation

Online evaluation

- **Idea:** See how normal users interact with a live retrieval system (“living lab”) when just using it
- Observe implicit behavior
 - Clicks, skips, saves, forwards, bookmarks, likes, etc.
- Try to infer differences in behavior from different flavors of the live system
 - A/B testing, interleaving

A/B testing

- Users are divided into two control (**A**) and treatment (**B**) groups
 - **A** uses the production system
 - **B** uses an experimental system
- Measure relative system performance based on usage logs



Interleaving

- Combine two rankings (A and B) into a single list
- Determine a winner on each query impression
 - Can be a draw too
- Aggregate wins on a large number of impressions to determine which ranker is better

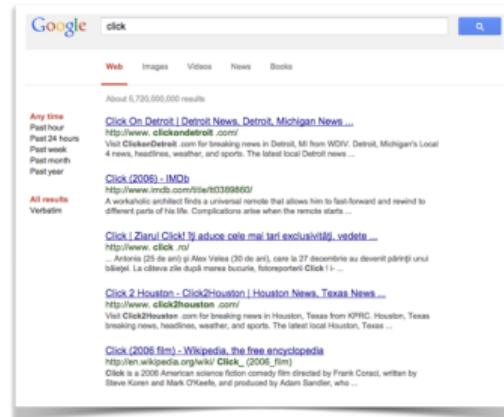


A/B testing vs. interleaving

- A/B testing
 - Between subject design
 - Can be used for evaluating any feature (new ranking algorithms, new features, UI design changes, etc.)
- Interleaving
 - Within subject design
 - Reduces variance (same users/queries for both A and B)
 - Needs 1 to 2 orders of magnitude less data
 - ~100K queries for interleaving in a mature web search engine ($\gg 1M$ for A/B testing)
 - Limited to evaluating ranked lists

Measures in online evaluation

- Inferred from observable user behavior
- Clicks
- Mouse movement
- Browser action
 - Bookmark, save, print, ...
- Time
 - Dwell time, time on SERP, ...
- Explicit judgment
 - Likes, favorites, ...
- Query reformulations
- ...



Challenges in online evaluation

- Simple measures break!

weather san francisco

40,000,000 RESULTS Any time ▾

[Weather in San Francisco, California](#)
bing.com/weather Data from AccuWeather

56°F
Cloudy 76°F | 50°F

Sun Mon Tue Wed Thu

58° / 49° 57° / 47° 62° / 46° 63° / 47° 66° / 47°

[10 Day Weather Forecast for San Francisco - weather.com](#)
www.weather.com/weather/10day/San+Francisco/CA-USCA0087 ~ Oct 26, 2013 - Rain or shine? Be prepared with the most accurate 10 day forecast, complete with highs, lows, chance of precipitation and more from weather.com

[San Francisco, CA Weather Forecast from Weather Underground](#)
www.wunderground.com/weather-forecast/US/CA/San_Francisco.html ~ Oct 26, 2013 - Get the latest forecast for weather in San Francisco CA along with updates about temperature, humidity, weather radar and more on Weather ...

[San Francisco Weather - AccuWeather Forecast for CA 94103](#)
www.acuweather.com ~ North America | United States | California
Get the San Francisco weather forecast. Access hourly, 10 day and 15 day forecasts along with up to the minute reports and videos for San Francisco, CA 94103 from ...

Instant answers
(satisfaction not observable)

Tom Hanks



Thomas Jeffrey "Tom" Hanks is an American actor, producer, writer, and director. Hanks is best known for his roles in Big, A League of Their Own, Sleepless in Seattle, Forrest Gump, Apollo 13, Saving Private Ryan, You've Got Mail, The Green Mile, Cast Away, and The Da Vinci Code, as ... +

[www.imdb.com](#) [en.wikipedia.org](#)

  91 KLUT

Born: Jul 9, 1956 (age 57) · Concord
Net worth: \$350 million USD (2012)
Spouses: Rita Wilson (1988) · Samantha Lewes (1978 - 1987)
Children: Colin Hanks · Elizabeth Hanks · Chet Hanks · Truman Theodore Hanks
Upcoming movies: Saving Mr. Banks
Siblings: Jim Hanks · Sandra Hanks · Larry Hanks

Movies and TV shows



Captain Phillips 2013
Cloud Atlas 2012
Forrest Gump 1994
Cast Away 2000
The Green Mile 1999

People also search for

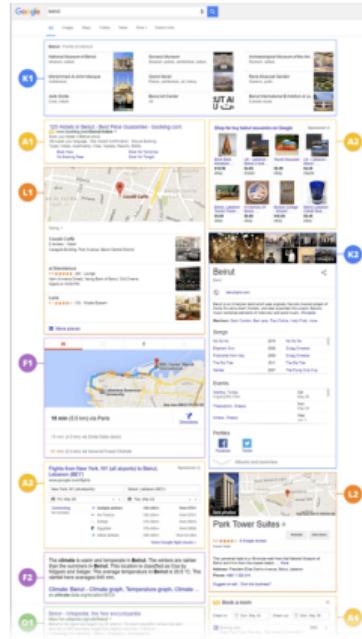


Rita Wilson spouse
Tom Cruise
Leonardo DiCaprio
Colin Hanks son
Sandra Bullock

Exploration
(more time/queries is not necessarily bad effort)

Challenges in online evaluation

- Whole page relevance
- Page is composed by a layered stack of modules
 - Web result ranking
 - ⇒ Result caption generation
 - ⇒ Answer triggering/ranking
 - ⇒ Knowledge panel composition
 - ⇒ Whole page composition
- Changes in modules lower in the stack have upstream effects



Pros and cons of online evaluation

- Advantages
 - No need for expensive dataset creation
 - Perfectly realistic setting: (most) users are not even aware that they are guinea pigs
 - Scales very well: can include millions of users
- Disadvantages
 - Requires a service with lots of users
 - Can be highly nontrivial how to interpret implicit feedback signals
 - Experiments are difficult to repeat

Offline vs. online evaluation

	Offline	Online
Basic assumption	Assessors tell you what is relevant	Observable user behavior can tell you what is relevant
Quality	Data is only as good as the guidelines	Real user data, real and representative information needs
Realisticity	Simplified scenario, cannot go beyond a certain level of complexity	Perfectly realistic setting (users are not aware that they are guinea pigs)
Assessment cost	Expensive	Cheap
Scalability	Doesn't scale	Scales very well
Repeatability	Repeatable	Not repeatable
Throughput	High	Low
Risk	None	High

Statistical significance testing

Statistical significance testing

- Comparison between two systems.
- Inherent noise in evaluation (e.g., variations in topics, assessors' behavior).
- Question: Is a result likely due to chance?

Essential ingredients of a significance test

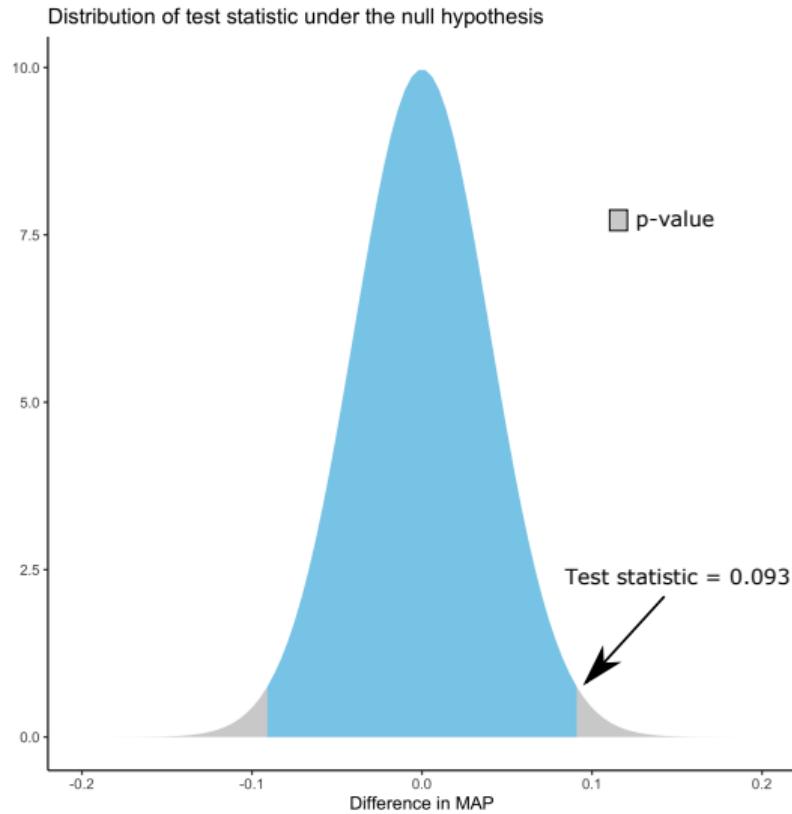
- A **test statistic or criterion** used to compare two systems. The difference in mean of an IR metric is commonly used.
- A **null hypothesis** H_0 and an **alternative hypothesis** H_1 .
- A distribution of the test statistic given H_0 .
- A **significance level** α used to determine if the comparison is statistically significant.
- **p-value** probability which determine whether there is evidence to reject H_0 .

$$\text{p-value} = P(T(X^*) \leq T(X_0) \mid H_0) + P(T(X^*) \geq T(X_0) \mid H_0) \quad (1)$$

Test statistic example

System A	System B	Difference (A-B)
0.2215	0.0765	0.145
0.3924	0.0426	0.3498
0.6540	0.5738	0.0802
0.5611	0.1571	0.404
0.9186	0.9881	-0.0695
0.1104	0.7164	-0.606
0.6086	0.7507	-0.1421
0.5062	0.4350	0.0712
0.9688	0.3959	0.5729
0.9950	0.8709	0.1241
Mean Average Precision (<i>MAP</i>)	0.5937	0.093

p-value example



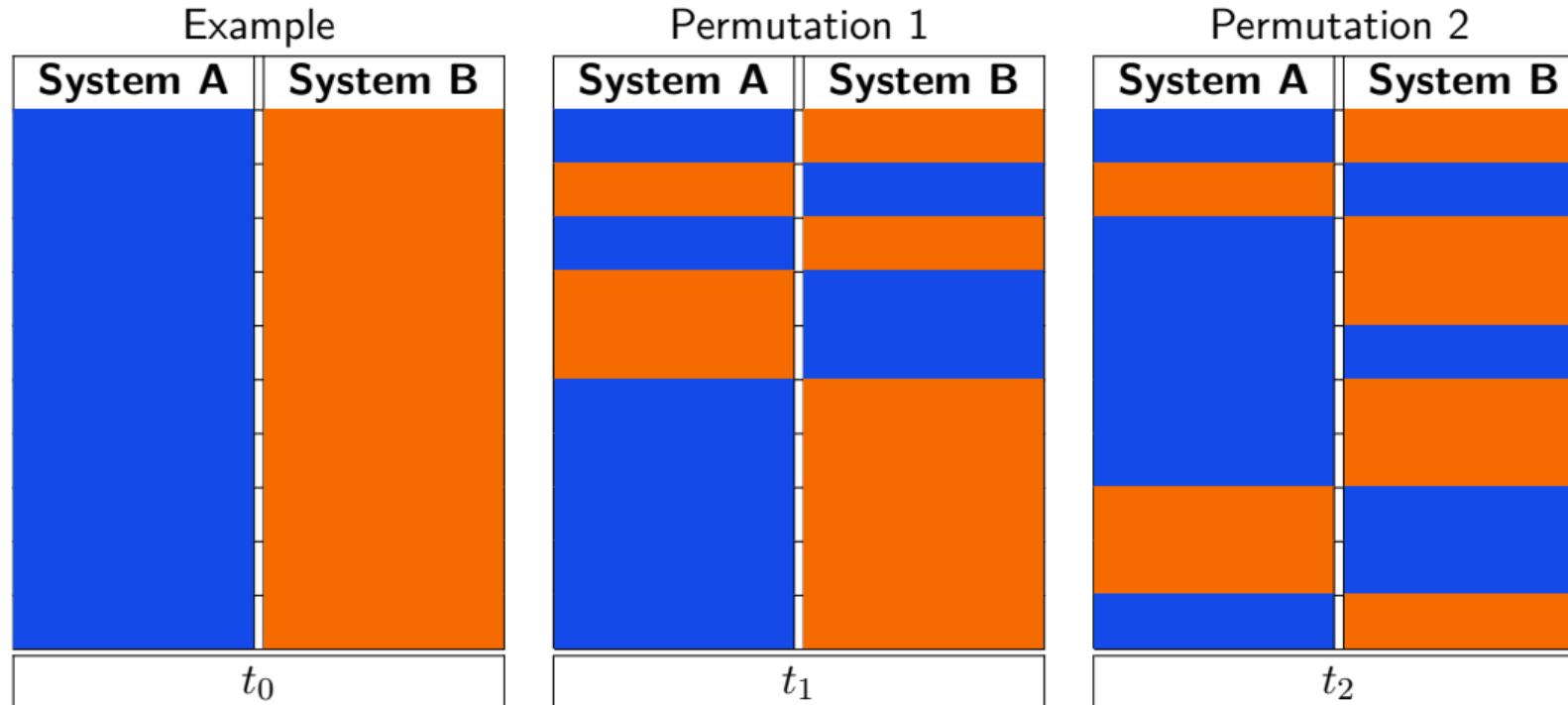
Commonly-used tests

- Randomization (permutation) test ⇐ our focus
- Student's paired sample t-test ⇐ our focus
- Wilcoxon signed rank test
- Bootstrap test
- Sign test
- ...

Randomization (permutation) test

- Null hypothesis H_0 : systems A and B are identical.
- alternative hypothesis H_1 : systems A and B are not identical.
- Test statistic of your choice. For this example, we use the mean difference.
- Distribution of test statistic T
 - Create all or a sample of permutations.
 - Record test statistic for each permutation.
- Compute p-value

Randomization (permutation) test



$T = [t_0, t_1, t_2, \text{etc.}, t_p]$, with p as number of permutations.

Student's paired sample t-test

- Hypothesis for paired sample t-test
 - $H_0 : \bar{x}_A = \bar{x}_B$, systems A and B are random samples from the same normal distribution.
 - $H_1 : \bar{x}_A \neq \bar{x}_B$
- $t = \frac{\bar{x}_D}{\frac{s_D}{\sqrt{n}}}$, with \bar{x}_D and s_D as the average and standard deviation of the differences between all pairs.
- Compute p-value

Randomization test vs. Student's t-test

	Randomization test	Student's t-test
Test statistic	Any	Difference of means
Normality assumption	No	Yes

Exercise

E5-4 Statistical significance testing

Summary

- Ingredients of offline test collections
- Collecting relevance assessments (expert judges/crowdsourcing, pooling, binary vs. graded relevance)
- Set retrieval measures (precision, recall, F1)
- Ranked retrieval measures (AP, RR, NDCG)
- Evaluating rankings for multiple queries
- Online evaluation (A/B testing and interleaving)
- Statistical significance testing

Reading

- Text Data Management and Analysis (Zhai&Massung)
 - Chapter 9
- Smucker, Mark D., James Allan, and Ben Carterette. "A comparison of statistical significance tests for information retrieval evaluation." *Proceedings of the sixteenth ACM Conference on information and knowledge management*. 2007.

Knowledge Bases and Entity Retrieval

[DAT640] Information Retrieval and Text Mining

Ivica Kostric & Krisztian Balog
University of Stavanger

September 12, 2022



CC BY 4.0

In this module

1. Entity-oriented search
2. Elasticsearch
3. Entity retrieval

Entity-oriented search

Question

What do you think *semantic search* means?

Examples of semantic search

Query: buy books

Google search results for "buy books" (2005):

Results 1 - 10 of about 259,000,000 for [buy books](#) (0.15 seconds)

Sponsored Links:

- [Compare book prices](#) (www.lookbookstores.com) Compare prices at dozens of stores Save up to 90% on books & textbooks
- [Buy books](#) (CollegeBoard.com) College Planning, SAT Preparation, Financial Aid - Books & More
- [Bookliquidator.com](#) New Books At Below Wholesale Prices No Minimums, We Dropship Also.
- [Used Books](#) (www.bookfinder4u.com) Compare New, Used, Text Book Prices Save up to 90% Today!
- [Welcome to Amazon](#) Signup and join now Buy stuff and save
- [Buy Books and Gifts](#) An awesome new shopping mall of 500 stores for books and holiday gifts

Regular search results:

- [Amazon.com Online Shopping for Electronics, Apparel, Computers ...](#) Online shopping from the earth's biggest selection of books, magazines, music, DVDs, ... Redeem or buy a gift certificate • Visit our Help department. ...
- [www.amazon.com/exec/obidos/subscribe/home.html - 56k - 30 Sep 2005 - Cached - Similar pages](#)
- [Amazon.com Books New & used textbooks, biographies, children's ...](#) Online shopping for millions of new & used books on thousands of topics at everyday low ... Redeem or buy a gift certificate • Visit our Help department. ...
- [www.amazon.com/exec/obidos/hg/browse/-/283156 - 56k - 30 Sep 2005 - Cached - Similar pages](#)
- [Barnes & Noble.com - Home](#) Browse All Books By Subject ... Special Offer: Buy 2, Get the 3rd Free. Buy two paperbacks and get a third for free! Browse our special collection of Modern ...
- [www.barnesandnoble.com/ - 54k - 30 Sep 2005 - Cached - Similar pages](#)
- [Powell's Books - Used, New, and Out of Print](#) New, used, and out of print books; many categories with detailed descriptions and illustrations of selected books.

Google result page in 2005

Google search results for "buy books" (2018):

About 5,160,000,000 results (0.64 seconds)

Adibris online bookstore | Norges største bokutvalg | adibris.com

More than 10 million titles - Quick deliveries and low prices. Over 10 million titles. Trygg e-handel.

Nyheter

Diverse bokutgivere fra hele landet og internasjonalt.

Massa nyttig informasjon om bøker og boklitteratur.

Aktuelt i media

Det siste med boknytter fra alle medier.

Se hvilke bøker som synes i media

Book Depository | Free Delivery Worldwide | bookdepository.com

Discover 18 Million Books with Free Delivery on All Orders. Over 18 Million Titles. Leading Online Bookstore. Everyday Low Prices. Free Delivery To Norway. Types: Romance, Crime & Thriller, Children's Books, Food & Drink, Travel & Holiday, Fiction, Fantasy Books, Classics Books, Travel & Holiday Guides. The Bargain Shop - Coming Soon

Map showing locations of bookstores in Stavanger, Norway:

- A** **Cormac Books Stavanger**
Cormac Books Stavanger
4.5 km - Stavanger 51 89 17 11
Open - Closes 7PM
- B** **Notabene Tvedtenteret**
Book Store - Tvedtenteret
4.7 km - Stavanger 52 90 05 80
Open - Closes 8PM
- C** **Notabene Arkaden Torgterrassen**
Book Store - Arkaden Torgterrassen
4.4 km - Stavanger 51 89 09 88
Open - Closes 8PM

Google result page today

Examples of semantic search

Query: *flight berlin chicago*

Google search results for "flight berlin chicago":

- Best Fares to Berlin, Germany**: Best Fares to Berlin, Germany. Sign Up For Best Fare Alerts Track Best Fares to ... Chicago, IL - O'Hare (ORD), \$356.00*. Chicago, IL - all (CHI), \$356.00* ... search.travel.yahoo.com/search/bfsearch/intl=us&dc=BER&xt_main=Berlin&source=37k - Cached - Similar pages
- Cheap Flights from Chicago to Berlin, Chicago Flights | Travelzoo ...**: Find Chicago Flights through Travelzoo SuperSearch. SuperSearch recommends the best airline for your itinerary flights.travelzoo.com/Air/Germany/Berlin_BER/USA_IU/Chicago_CHI/ - 10k - Cached - Similar pages
- Cheap airline tickets to Berlin, Germany departing from Chicago ...**: Cheap airline tickets to Berlin, Germany from Chicago, Illinois. Compare cheap airline tickets for flights to Berlin, Germany departing from Chicago, ... www.cheapflights.com/flights/Berlin/Chicago/ - 57k - Cached - Similar pages
- Cheap airline tickets to Berlin, Germany**: Cheap airline tickets to Berlin, Germany at Cheapflights.com. Compare flights to Berlin, Germany from the USA. www.cheapflights.com/flights/Berlin/ - 35k - 30 Sep 2005 - Cached - Similar pages
- Gridskipper, the Urban Travel Guide**: Scouting the world for discount flights, chic hotels - and pretty people. Gridskipper, the decadent travel guide.

Google result page in 2005

Google search results for "flight berlin chicago":

- Berlin-Chicago from €189 - eDreams.com**: ★★★ 4.1 Rating for edreams.com 3.6 - 2,641 reviews Compare flight prices and book with confidence. Book Your Flight Best Price Guarantee. Customer Service 7/7. Cheap Flights from €189. More than 760 Airlines. Flight + Hotel Offers. Book Now & Get Best Deals Take Off with eDreams.com and Save. We Find the Best Deals For You!
- Flights from Berlin, Germany (all airports) to Chicago, USA (all airports)**: www.google.flights
- | Airline | Flight | Departure | Arrival | Price |
|------------------|-----------|----------------|-----------------|----------------|
| WOW | 17h 15min | Mon, October 8 | Tue, October 23 | from NOK 3,531 |
| Aer Lingus | 14h 35min | | | from NOK 5,030 |
| LOT | 14h 30min | | | from NOK 5,407 |
| Turkish Airlines | 17h 30min | | | from NOK 5,969 |
| Other airlines | 11h 56min | | | from NOK 6,617 |
- Cheap Flights from Berlin to Chicago from \$357 | (BER - CHI) - KAYAK**: https://www.kayak.com / Flights / Worldwide / Europe / Germany * Fly from Berlin to Chicago on WOW air from \$357, Icelandair from \$461, KLM from \$517, Delta from \$720... Search and find deals on flights to Chicago.

Google result page today

Semantic search

A broad view on semantic search:

Definition

Semantic search encompasses a variety of methods and approaches aimed at aiding users in their information access and consumption activities, by understanding their context and intent.

- “Search with meaning”
 - Beyond literal matches
 - Understanding what the query actually means
 - Searching for *things* instead of *strings*
- Our notion of semantics: references to meaningful, i.e., machine understandable structures

Entity-oriented search

Entity-oriented search to refer to a broad range of information access tasks where entities are used as information objects, instead of or in addition to documents

Definition

Entity-oriented search is the search paradigm of organizing and accessing information centered around entities, and their attributes and relationships.

- Note: entity-oriented search is a subset of semantic search

Examples of entity-oriented search

Google

Web Images Maps News Videos More Search tools

About 78,200,000 results (0.59 seconds)

Lisbon - Official Website - visitlisboa.com
www.visitlisboa.com/TouristInfo ▾
Discover Golf, Music, Surf And Food Discover Everything About Lisbon!

Lisbon - Wikipedia, the free encyclopedia
<https://en.wikipedia.org/wiki/Lisbon> ▾ Wikipedia ▾
Lisbon (/lɪzˈbən/; Portuguese: Lisboa, IPA: [liʒˈboɐ]) is the capital and the largest city of Portugal, with a population of 552,700 within its administrative limits ...
Belém Tower - Tagus - Belém - Alcântara

Images for lisbon Report images



[More images for lisbon](#)

Lisbon, Portugal - Lonely Planet
www.lonelyplanet.com/portugal/lisbon ▾ Lonely Planet ▾
Spread across steep hillsides that overlook the Rio Tejo, Lisbon offers all the delights you'd expect of Portugal's star attraction, yet with half the fuss of other ...
Top things to do in Lisbon - Best places to stay in Lisbon - Portugal image gallery

Lisbon Tourism: Best of Lisbon, Portugal - TripAdvisor
www.tripadvisor.com ... Central Portugal > Lisbon District ▾ TripAdvisor ▾
Lisbon Tourism: TripAdvisor has 486539 reviews of Lisbon Hotels, Attractions, and Restaurants making it your best Lisbon resource.



Lisbon

Capital of Portugal

Lisbon, Portugal's hilly capital, is a coastal city known for its cafe culture and soulful Fado music. From imposing São Jorge Castle, the view encompasses the old city's pastel-colored buildings, Tagus Estuary and the Ponte 25 de Abril suspension bridge. Nearby, the National Azulejo Museum displays 5 centuries of decorative ceramic tiles. And just outside Lisbon is a string of Atlantic beaches, from Cascais to Estoril.

Area: 32.74 mi²

Weather: 61°F (16°C), Wind NE at 7 mph (11 km/h), 82% Humidity

Local time: Tuesday 10:41 AM

Hotels: 3-star averaging \$50, 5-star averaging \$140. [View hotels](#)

Population: 530,847 (2012) [UNdata](#)

Points of interest

[View 15+ more](#)



Examples of entity-oriented search

A screenshot of a Google search results page for the query "lisbon things to see". The results are filtered to show "Lisbon / Points of interest". A grid of nine images is displayed, each with a caption below it: Belém Tower, Jerónimos Monastery, São Jorge Castle, Lisbon Oceanarium, Praça do Comércio, Calouste Gulbenkian Foundation, Museu Calouste Gulbenkian, Lisbon Zoo, and Santa Justa L.

Things To Do In Lisbon - visitlisboa.com

[www.visitlisboa.com/ThingsToDo](#) ▾

Find Out What To Visit And Where To Sleep. Everything About Lisbon In
Lisbon: Associação de Turismo de Lisboa

Lisbon Things To See - City Tours, Day Trips, and more

[www.viator.com/lisbon](#) ▾

4.4 ★★★★★ rating for viator.com

Book Lisbon Things To Do on Viator.

24/7 Live Support - Verified Reviews & Photos - Low Price Guarantee

Viator.com has 1,456,772 followers on Google+

Sintra Tours - Lisbon Tours - Lisbon Fado Show - Porto Wine Tours

Best Things to Do in Lisbon - holidaylettings.co.uk

[www.holidaylettings.co.uk/rentals](#) ▾

Find, Book & Enjoy The Best Rentals Book Securely or Enquire Online Now

Late Deals - Holiday Destinations - Payment Protection - Holiday Ideas

The Top 10 Things to Do in Lisbon - TripAdvisor - Lisbon ...

[www.tripadvisor.com/Attractions-g189158-Activities-Lisbon...](#) ▾ TripAdvisor =

Hotels near Oceanario de Lisboa. Hotels near Torre de Belém. Hotels near Castelo de São Jorge. Hotels near Mosteiro dos Jerónimos. Hotels near Alfama. Hotels near Tram 28. Hotels near Bairro Alto. Hotels near Museu da Fundação Calouste Gulbenkian. Oceanário de Lisboa - Miradouro da Senhora do Monte - Mosteiro dos Jerónimos

Best Things to Do in Lisbon | U.S.News Travel



Lisbon

Capital of Portugal

Lisbon, Portugal's hilly capital, is a coastal city known for its cafe culture and soulful Fado music. From imposing São Jorge Castle, the view encompasses the old city's pastel-colored buildings, Tagus Estuary and the Ponte 25 de Abril suspension bridge. Nearby, the National Azulejo Museum displays 5 centuries of decorative ceramic tiles. And just outside Lisbon is a string of Atlantic beaches, from Cascais to Estoril.

Area: 32.74 mi²

Weather: 61°F (16°C), Wind NE at 7 mph (11 km/h), 82% Humidity

Hotels: 3-star averaging \$50, 5-star averaging \$140. [View hotels](#)

Local time: Tuesday 10:42 AM

Population: 530,847 (2012) [Unidata](#)

Colleges and Universities: [University of Lisbon](#), More

Examples of entity-oriented search

Google

All Maps Videos Shopping News More Search tools

About 3,980 results (0.29 seconds)

KLM Flight 1799
On-time - departs in 10 hours 50 mins

AMS  MUC

Departs Amsterdam, Tuesday, May 10	Arrives Munich, Tuesday, May 10				
Time	Terminal	Gate	Time	Terminal	Gate
6:40 PM	1	C7	8:05 PM	1	-

Showing airport times Feedback

KLM flight KL1799 - Flightradar24
<https://www.flightradar24.com/data/flight/kl1799> ▾ Flightradar24 ▾
KL1799 (KLM) - Live flight status, scheduled flights, flight arrival and departure times, flight tracks and playback, flight route and airport.

KLM (KL) #1799 FlightAware
<https://flightaware.com/live/flight/KLM1799> ▾ FlightAware ▾
KLM (KL) #1799 Flight Tracker (KLM1799) Flight Tracker (en route flights, arrivals, departures, history) with live maps and aircraft photos.

KL 1799 Flight Status - FlightStats
www.flightsstats.com/FlightStatus/flightStatusByFlight.do?... ▾ FlightStats ▾
Check the current status of flight (KL) KLM 1799 complete with live maps, weather and more.

KL1799 / KLM1799 — KLM Royal Dutch Airlines — Plane Fin...
<https://planefinder.net/data/flight/KL1799> ▾
Plane Finder Data has the latest real-time information on flight KL1799 / KLM1799 (KLM Royal Dutch Airlines).

KL1799 schedule. (KLM flight: Amsterdam -> Munich)
info.flightradar24.com/flight/KLM_KL_1799 ▾
KL 1799 Non-stop Embraer EMB 170 / EMB 190 (EMJ) 1:25 Effective from 2016-10-30
... KL 1799 Non-stop Fokker 70 (F70) 1:25 Effective 2016-10-31 through ...

Why?

- From a **user perspective**
 - Entities are natural units for organizing information
 - We care about and mostly think in terms of real-world things and their connections
- From a **machine perspective**
 - Entities allow for a better understanding of search queries, of document content, and even of users (e.g., their context and preferences)
 - Entities enable search engines to be more intelligent

Entity-oriented search

- Defining entities
- Entity-oriented search tasks
- Meet the data: Wikipedia
- Meet the data: Knowledge bases

What is an entity?



people



locations



organizations



products

What is an entity?

Commonly accepted definition:

Definition

An *entity* is an object or concept in the real world that can be distinctly identified.

- Issues

- What does the “real world” mean? (Is “Superman” an entity or not?)
- Answering this will likely lead to a long philosophical discussion about “existence”

What is an entity?

Pragmatic, data-oriented definition:

Definition

An *entity* is a uniquely identifiable object or thing, characterized by its name(s), type(s), attributes, and relationships to other entities.

Our universe is restricted to some particular registry of entities:

Definition

An *entity catalog* is a collection of entries, where each entry is identified by a unique ID and contains the name(s) of the corresponding entity.

Named entities vs. concepts

- Two main classes of entities may be distinguished
 - *Named entities* are real-world objects that can be denoted by a proper noun
 - For example, specific persons, locations, organizations, products, events, etc.
 - *Concepts* are abstract objects, including, but not limited to
 - Mathematical and philosophical concepts (e.g., "distance," "axiom," "quantity")
 - Physical concepts and natural phenomena (e.g., "gravity," "force," "wind")
 - Psychological concepts (e.g., "emotion," "thought," "identity"), and social concepts (e.g., "authority," "human rights," "peace")
- This distinction is mostly of a philosophical nature. From a technical perspective, the exact same methods may be used for names entities and concepts.

Properties of entities

- **Unique identifier**
 - There must be a one-to-one correspondence between each entity identifier (ID) and the (real-world or fictional) object it represents
 - For example, social security number, product EAN, MAC address, etc.
- **Name(s)**
 - Names do not uniquely identify entities; multiple entities may share the same name
 - The same entity may be known by more than a single name (e.g., “Barack Obama,” “President Obama”)
 - Alternative names are called *surface forms* or *aliases*
- **Type(s)**
 - Entities may be categorized into multiple *entity types*
 - Types can also be thought of as containers (semantic categories) that group together entities with similar properties
 - Analogy to object-oriented programming: an entity of a type is like an instance of a class
 - Entity types are often organized in a hierarchical structure (*type taxonomy*)

Properties of entities (2)

- **Attributes**
 - Entities are characterized by attributes
 - Different types of entities typically have different sets of attributes
 - People: date and place of birth, weight, height, parents, spouses, etc.
 - Places: latitude, longitude, population, postal code(s), country, continent, etc.
 - Attributes always have *literal values*
- **Relationships**
 - May be seen as “typed links” between entities (or attributes where the value is another entity)
 - For example, parents of a person, capital of a country, manufacturer of a product, etc.

Entity-oriented search

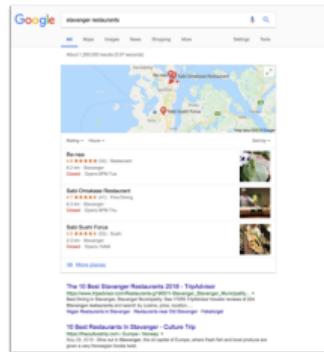
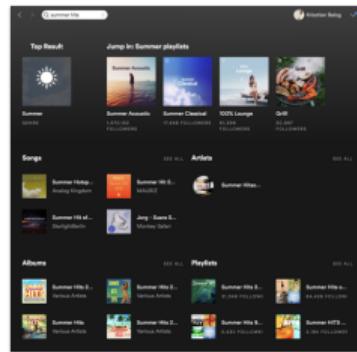
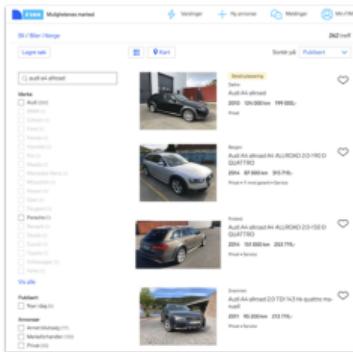
- Defining entities
- Entity-oriented search tasks
- Meet the data: Wikipedia
- Meet the data: Knowledge bases

Entity-oriented search tasks

- Entities as the unit of retrieval
 - Entity retrieval
- Entities for knowledge representation
 - Entity linking
 - Knowledge base population
- Entities for an enhanced user experience
 - Query assistance
 - Recommendations

Entity retrieval

- Task: given a search query, return a ranked list of entities (instead of documents)



Entity linking

- Task: recognize mentions of entities in text and assign to these unique identifiers from a knowledge repository

The screenshot shows the DBpedia Spotlight web application. At the top is the logo "DBpedia Spotlight" with a stylized sunburst icon above it. Below the logo is a search bar containing the text: "First documented in the 13th century, Berlin was the capital of the Kingdom of Prussia (1701–1918), the German Empire (1871–1918), the Weimar Republic (1919–33) and the Third Reich (1933–45). Berlin in the 1920s was the third largest municipality in the world. After World War II, the city became divided into East Berlin -- the capital of East Germany -- and West Berlin, a West German exclave surrounded by the Berlin Wall from 1961–89. Following German reunification in 1990, the city regained its status as the capital of Germany, hosting 147 foreign embassies."

Below the text are several user interface elements:

- A "Confidence:" slider with a value of 0.5.
- A "Language:" dropdown set to "English".
- A checkbox labeled "n-best candidates" which is unchecked.
- Two buttons: "SELECT TYPES..." and "ANNOTATE".
- A "BACK TO TEXT" button at the bottom right.

Architecture of an entity-oriented search system

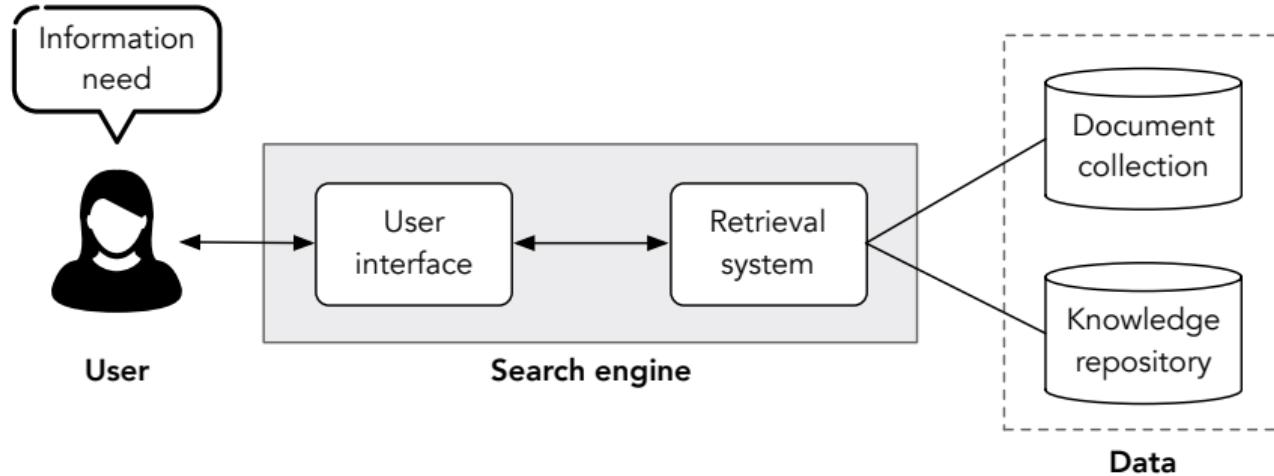


Figure: Illustration is taken from (Balog, 2018) [Fig. 1.3]

Representing properties of entities

Information about entities can be represented and stored in semi-structured or in structured form

Definition

A *knowledge repository* (KR) is a catalog of entities that contains entity type information, and (optionally) descriptions or properties of entities, in a semi-structured or structured format.

- Classic example: Wikipedia
 - Each article in Wikipedia is an entry that describes a particular entity
 - Articles are also assigned to categories (which can be seen as entity types)
 - Wikipedia articles also contain information about attributes and relationships of entities, but not in a structured form

Representing properties of entities

To organize and store information about entities in a structured form, entities may be represented as a set of statements (facts or assertions)

Definition

A *knowledge base* (KB) is a structured knowledge repository that contains a set of facts (assertions) about entities.

- Note: all knowledge bases are also knowledge repositories, but the reverse is not true
- Conceptually, entities in a knowledge base may be seen as nodes of a graph, with the relationships between them as (labeled) edges
 - When this graph nature is emphasized, a knowledge base may also be referred to as a *knowledge graph* (KG)

Representing properties of entities

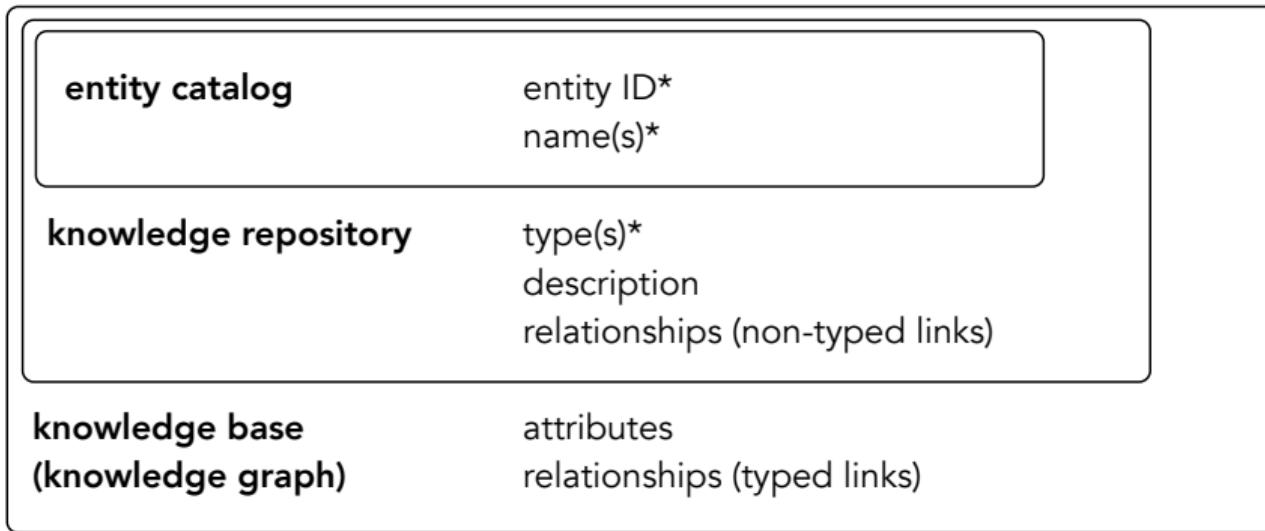


Figure: Illustration is taken from (Balog, 2018) [Fig. 1.2]

Entity-oriented search

- Defining entities
- Entity-oriented search tasks
- Meet the data: Wikipedia
- Meet the data: Knowledge bases

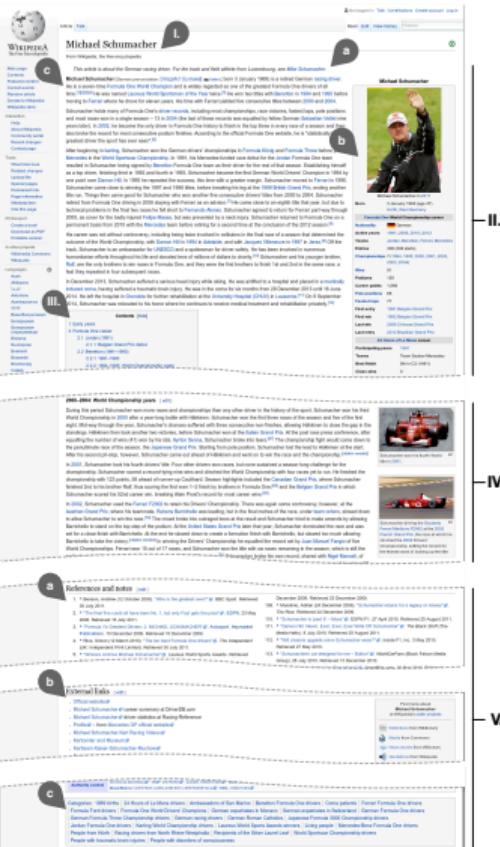
Question

Why is Wikipedia relevant for entity-oriented search?

- One of the most popular web sites in the world and a trusted source of information for many people
- Content is created through the collaborative effort of a community of users, facilitated by a *wiki* platform
- Available in nearly 300 languages, although English is by far the most popular, with over five million articles

Wikipedia as a knowledge repository

- Most of Wikipedia's entries can be considered as (semi-structured) representations of entities



The anatomy of a Wikipedia article

- Title
- Lead section
 - Disambiguation links
 - Infobox
 - Introductory text
- Table of contents
- Body content
- Appendices and bottom matter
 - References and notes
 - External links
 - Categories

Entity-oriented search

- Defining entities
- Entity-oriented search tasks
- Meet the data: Wikipedia
- Meet the data: Knowledge bases

Knowledge base

- A data repository for storing entities and their properties in structured format
- A set of assertions about the world, describing specific entities and their relationships
- Conceptually, it forms a graph (“knowledge graph”)

Resource Description Framework (RDF)

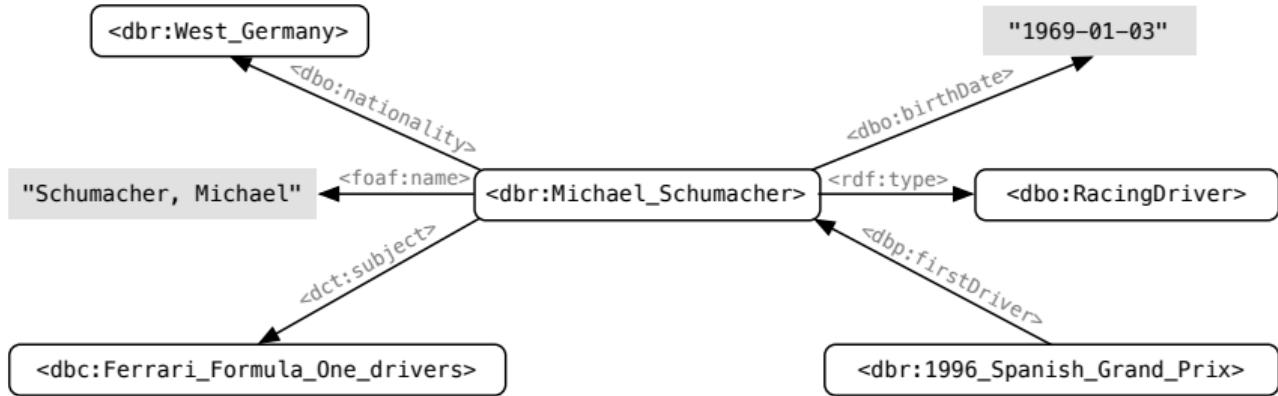
- A language designed to describe “things” (which are referred to as *resources*)
- Each resource is assigned a *Uniform Resource Identifier* (URI), making it uniquely and globally identifiable
- Each RDF statement is a triple, consisting of subject, predicate, and object components
 - **Subject:** always a URI, denoting a resource
 - **Predicate:** always a URI, corresponding to a relationship or property of the subject resource
 - **Object:** either a URI (referring to another resource) or a literal

Example

Michael Schumacher (born 3 January, 1969) is a retired German racing driver, who raced in Formula One for Ferrari.

subject	predicate	object
<dbr:Michael_Schumacher>	<foaf:name>	"Schumacher, Michael"
<dbr:Michael_Schumacher>	<dbo:birthPlace>	<dbr:West_Germany>
<dbr:Michael_Schumacher>	<dbo:birthDate>	"1969-01-03"
<dbr:Michael_Schumacher>	<rdf:type>	<dbo:RacingDriver>
<dbr:Michael_Schumacher>	<dct:subject>	<dbc:Ferrari_Formula_One_drivers>

Example



Related technologies

- RDF describes the instance level in the knowledge base
- RDFS and OWL are vocabularies for ontological modeling
 - An ontology is a means to formalizing knowledge. Building blocks of an ontology include classes, instances, relations, attributes, restrictions, and rules and axioms.
- Serializations for RDF data: Notation-3, Turtle, N-Triples, RDFa, and RDF/JSON
- SPARQL is a structured query language for retrieving and manipulating RDF data
- Triplestores are special-purpose databases designed for storing and querying RDF data

Public knowledge bases

- Cyc
 - Started in 1984 with the goal to manually build a knowledge base of everyday common knowledge
 - ... still building and far from complete
 - “one of the most controversial endeavors of the artificial intelligence history”

Public knowledge bases

- DBpedia
 - Extracted from Wikipedia (mostly from infoboxes) using a set of manually constructed mapping rules
 - Community effort, users collaboratively create and edit the mapping rules
 - Available in multiple languages
 - Contains over 5 million entities (English)

Example

About: Stavanger

An Entity of Type : [city](#), from Named Graph : [http://dbpedia.org](#), within Data Space : [dbpedia.org](#)

Stavanger /sta'væŋər/ (Norwegian pronunciation: [sta'vɑŋər]) is a city and municipality in Norway. The city is the third-largest urban zone and metropolitan area in Norway (through conurbation with neighbouring Sandnes) and the administrative centre of Rogaland county. The municipality is the fourth most populous in Norway. Located on the Stavanger Peninsula in Southwest Norway, Stavanger counts its official founding year as 1125, the year the Stavanger Cathedral was completed. Stavanger's core is to a large degree 18th- and 19th-century wooden houses that are protected and considered part of the city's cultural heritage. This has caused the town centre and inner city to retain a small-town character with an unusually high ratio of detached houses, and has contributed signif

Property	Value
dbo:PopulatedPlace/areaMetro	▪ 2598.0
dbo:PopulatedPlace/areaTotal	▪ 71.0
dbo:PopulatedPlace/areaUrban	▪ 77.98
dbo:abstract	▪ Stavanger /sta'væŋər/ (Norwegian pronunciation: [sta'vɑŋər]) is a city and municipality in Norway. The city is the third-largest urban zone and metropolitan area in Norway (through conurbation with neighbouring Sandnes) and the administrative centre of Rogaland county. The municipality is the fourth most populous in Norway. Located on the Stavanger Peninsula in Southwest Norway, Stavanger counts its official founding year as 1125, the year the Stavanger Cathedral was completed. Stavanger's core is to a large degree 18th- and 19th-century wooden houses that are protected and considered part of the city's cultural heritage. This has caused the town centre and inner city to retain a small-town character with an unusually high ratio of detached houses, and has contributed significantly to spreading the city's population growth to outlying parts of Greater Stavanger. The city's rapid population growth in the late 20th century was primarily a result of Norway's booming offshore oil industry. Today the oil industry is a key industry in the Stavanger region and the city is widely referred to as the Oil Capital of Norway. The largest company in the Nordic region, Norwegian energy company Statoil is headquartered in Stavanger. Multiple educational institutions for higher education are located in Stavanger. The largest of these is the University of Stavanger. Domestic and international military installations are located in Stavanger, among these is the North Atlantic Treaty Organisation's Joint Warfare Center. Other international establishments, and especially local branches of foreign oil and gas companies, contribute further to a significant foreign population in the city. Immigrants make up 11.3% of Stavanger's population. Stavanger has since the early 2000s consistently had an unemployment rate significantly lower than the Norwegian and European average. In 2011, the unemployment rate was less than 2%. The city is also among those that frequent various lists of expensive cities in the world, and Stavanger has even been ranked as the world's most expensive city by certain indexes. Stavanger is served by international airport Stavanger Airport, Sola, which offers flights to cities in most major European countries, as well as a limited number of intercontinental charter flights. The airport was named most punctual European regional airport by flightstats.com in 2010. Every two years, Stavanger organizes the Offshore Northern Seas (ONS), which is the second largest exhibition and conference for the energy sector. Gladmat food festival is also held each year and is considered to be one of Scandinavia's leading food festivals. The city is also known for being one of the nation's premier culinary clusters. Stavanger 2008 European Capital of Culture (en)

Public knowledge bases

- Wikidata
 - Operated by the Wikimedia Foundation
 - Its goal is to provide the same information as Wikipedia, but in a structured format
 - Wikidata considers “claims” not “facts”
 - Each claim must be supported by a reference
 - Claims can contradict each other and coexist, thereby allowing opposing views to be expressed (e.g., different political positions)

Example

Items Discussion Read View history Search Wikidata

English Not logged in Talk Contributions Create account Log in

Stavanger (Q25416)

municipality in Rogaland, Norway
Stavanger, Rogaland, Norway | Stavanger, Norway

In more languages

Configure

Language	Label	Description	Also known as
English	Stavanger	municipality in Rogaland, Norway	Stavanger, Rogaland, Norway Stavanger, Norway
Hungarian	Stavanger	No description defined	
Danish	Stavanger	kommune i Rogaland i Norge	
Norwegian Bokmål	Stavanger	kommune i Rogaland i Norge	St. Svitnus by Stavanger kommune Sankt Svitnus by

All entered languages

Statements

instance of municipality of Norway

+ 0 references + add reference + add value

image Vaagen-modif.jpg

800 x 509, 267 KB

+ 0 references + add reference

Wikidata (65 entries) edit

- af Stavanger
- ar ستافنر
- bar-smp Stavangeris
- be_x-old| স্টাবেণ্র
- be স্টাবেণ্র
- bg Ставангер
- bpy ସ্টାବେଣର
- br Stavanger
- bs Stavanger
- ca Stavanger
- cab Stavanger (munisipyo)
- ce სტავანერი
- cs Stavanger
- cy Stavanger
- da Stavanger
- de Stavanger
- el Σταύρος
- en Stavanger
- eo Stavanger
- es Stavanger
- et Stavanger
- eu Stavanger
- fa ستافنر
- fi Stavanger
- fo Stavanger
- frr Stavanger
- fr Stavanger
- fy Stavanger
- gl Stavanger
- ha Stavanger
- he סטונגראן
- hr Stavanger
- hu Stavanger

Proprietary knowledge bases

- Google Knowledge Graph
 - "... the knowledge graph is one of Google's biggest search milestones of the last decade..."—Amit Singhal, Google's director of search
- Facebook Entity Graph
- Microsoft Satori
- ...

Connecting knowledge bases

- The same entity may be present in multiple knowledge bases
- A special predicate `<owl:sameAs>` can be used to connect URLs across different knowledge bases

subject	predicate	object
<code><dbr:Michael_Schumacher></code>	<code><owl:sameAs></code>	<code><fb:m.053w4></code>
<code><dbr:Michael_Schumacher></code>	<code><owl:sameAs></code>	<code><>wikidata:Q9671></code>

The Web of Data

- Increasingly more data is being exposed on the Web in the form of semantic annotations
 - Microdata, RDFa, JSON-LD
- Strong incentive for websites for marking up their content with semantic metadata: It allows search engines to better understand their content
- Standardization: development of schema.org
 - A common vocabulary used by major search providers (including Google, Microsoft, and Yandex) for describing commonly used entity types (including people, organizations, events, products, books, movies, recipes, etc.)

Example

Easy Chicken Satay Recipe - Allrecipes.com



allrecipes.com/recipe/132929/easy-chicken-satay/ ▾

★★★★★ Rating: 4,7 - 282 reviews - 2 hrs 45 mins - 418 cal

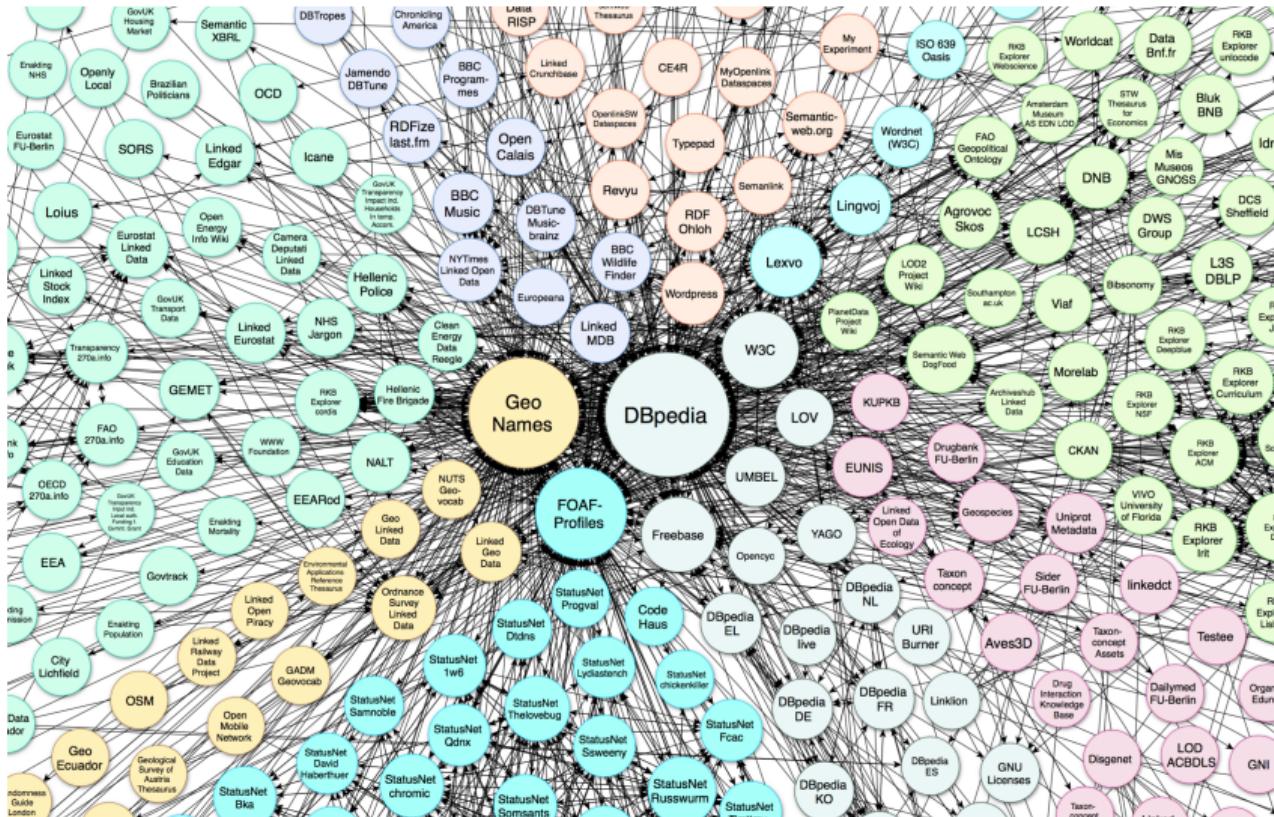
Bring 1 cup coconut milk, 1 tablespoon curry powder, peanut butter, **chicken** stock, and 1/4 cup brown sugar to a simmer in a saucepan over medium-high heat. Simmer for 5 minutes, stirring constantly, until smooth and thickened. Remove from heat and stir in lime juice and soy sauce; season to taste with salt.

```
<section class="ar_recipe_index full-page" itemscope
        itemtype="http://schema.org/Recipe">
    <link href="http://allrecipes.com/recipe/132929/easy-chicken-satay/"
          itemprop="url" />
    <meta itemprop="mainEntityOfPage" content="True" />
```

The Semantic Web

- Historically, data made available in RDF format was referred to as Semantic Web data
- One of the founding principles behind the Semantic Web is that data should be interlinked
- The term *Linked Data (LD)* refers to a set of best practices for publishing structured data on the Web
 - This is facilitated by the special “same-as” predicate
 - A knowledge base published using LD principles should be called *Linked Dataset*
- These “same-as” links connect all Linked Data into a single global data graph
- *Linked Open Data (LOD)* (a casual synonym for the Web of Data) emphasizes the fact that Linked Data is released under an open license

Linked Open Data



Elasticsearch

Elasticsearch

- An open source search engine built on top of Apache Lucene
- It is distributed - indices can be divided into shards and each shard can have zero or more replicas
- All of its functionality is available through a RESTful API

Exercise

E6-0 Elasticsearch Basics

Exercise

E6-1 Term statistics

Entity retrieval

Ad hoc entity retrieval

Entity retrieval is the task of answering queries with a ranked list of entities¹

Definition

Given a keyword query q and an entity catalog \mathcal{E} , *ad hoc entity retrieval* is the task of returning a ranked list of entities $\langle e_1, \dots, e_k \rangle, e_i \in \mathcal{E}$ with respect to each entity's relevance to q . The relevance of entities is inferred based on a collection of unstructured and/or (semi-)structured data.

¹Ad hoc refers to the standard form of retrieval in which the user, motivated by an ad hoc information need, initiates the search process by formulating and issuing a query

Example queries

martin luther king

disney orlando

Apollo astronauts who walked on the Moon

Winners of the ACM Athena award

EU countries

Hybrid cars sold in Europe

birds cannot fly

Who developed Skype?

Which films starring Clint Eastwood did he direct himself?

Main strategy

- Build on work on document retrieval
- Create an entity description or “profile” document is to be compiled for each entity in the catalog
 - Specifically, a fielded *entity document*
- Those entity description documents can be ranked the same way as documents

Entity retrieval

- Constructing term-based entity representations
- Ranking term-based entity representations

Term-based entity representations

- The key statistic is *term count*, $c(t; e)$: the number of times term t appears in the description constructed for entity e
- Other components of bag-of-words models can be derived analogously to document retrieval, e.g.,:
 - *Entity length*: $l_e = \sum_{t \in \mathcal{V}} c(t; e)$
 - *Term frequency*: $TF(t, e) = \frac{c(t; e)}{l_e}$
 - *Entity frequency*: $EF(t) = |\{e \in \mathcal{E} : c(t; e) > 0\}|$
 - *Inverse entity frequency*: $IEF(t) = \log \frac{|\mathcal{E}|}{EF(t)}$
 - where \mathcal{E} is the entity catalog and \mathcal{V} is the vocabulary of terms

Data collection

- Unstructured documents
- Semi-structured documents
- Structured knowledge bases

From unstructured documents

- Input: documents that are annotated with entities
 - Document-level annotations (e.g., tags)
 - Mention-level annotations (e.g., as in Wikipedia)

The **2004 Belgian Grand Prix** (formally the **Formula 1 Belgian Grand Prix 2004**)^[2] was a **Formula One motor race** held on 29 August 2004, at the **Circuit de Spa-Francorchamps**, near the town of **Spa, Belgium**. It was Race 14 of 18 in the **2004 FIA Formula One World Championship**. The race was contested over 44 laps and was won by **Kimi Räikkönen**, taking his and **McLaren's** only race win of the season from tenth place on the grid. Second place for **Michael Schumacher** won his seventh world championship, after beating third-placed **Rubens Barrichello**.

Figure: Excerpt from https://en.wikipedia.org/wiki/2004_Belgian_Grand_Prix.

- Output: (pseudo) term counts $\tilde{c}(t; e)$

From unstructured documents

- General formula, using documents as a bridge between terms and entities:

$$\tilde{c}(t; e) = \sum_{d \in \mathcal{D}} c(t, e; d) w(e, d)$$

- where
 - $c(t, e; d)$: number of co-occurrences between a term and an entity in a particular document
 - $w(e, d)$: strength of the *association* between the entity and the document

Using document-level annotations

- Entity description == the contents of all documents concatenated that are tagged with the given entity
- Formally:
 - Term count in the document if it mentions the entity, and are zero otherwise:

$$c(t, e; d) = \begin{cases} c(t; d), & e \in d \\ 0, & e \notin d . \end{cases}$$

- Binary document-entity weights:

$$w(e, d) = \begin{cases} 1, & e \in d \\ 0, & e \notin d . \end{cases}$$

Using mention-level annotations

- Assume that entity occurrences have been replaced with unique identifiers that behave as regular terms
- Simplest approach: consider terms that co-occur with the entity within a fixed window size of w
- Formally:

$$c(t, e; d) = \sum_{i=1}^{l_d} 1_d(i, t) \sum_{\substack{j=1 \\ |i-j| \leq w}}^{l_d} 1_d(j, e)$$

- where
 - $1_d(i, t)$ returns 1 if the term at position i in d is t , and 0 otherwise
 - $1_d(i, e)$ returns 1 if entity e appears at position i in d , and 0 otherwise

Using mention-level annotations

- Intuition: terms closer to the mention of an entity should be given more importance than terms appearing farther away
- Formally expressed using proximity kernels

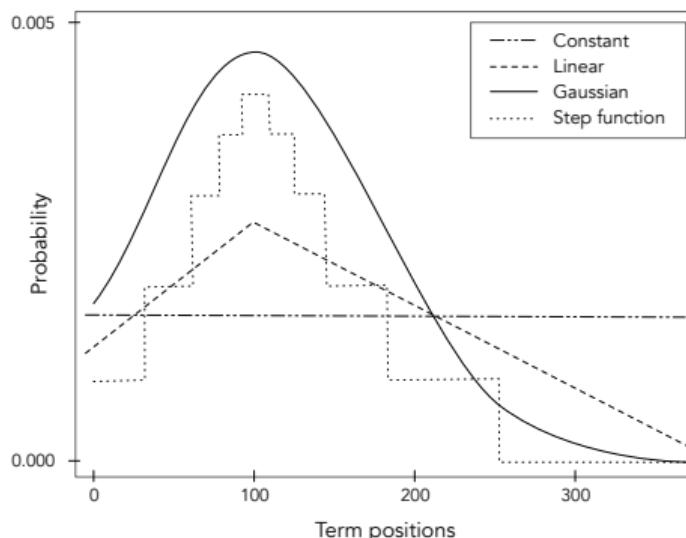


Figure: Illustration of proximity kernels from Balog (2018) [Fig. 3.3].

From semi-structured documents

- E.g., Wikipedia article, IMDB page, LinkedIn profile, ...
- Field content is typically extracted using wrappers (template-based extractors)

Example

FULL CAST AND CREW | TRIVIA | USER REVIEWS | IMDbPro | MORE ▾ SHARE

 The Matrix (1999)

R | 2h 16min | Action, Sci-Fi | 31 March 1999 (USA)

 8.7 /10
1,235,392 |  Rate This

A computer hacker learns from mysterious rebels about the true nature of his reality and his role in the war against its controllers.

Directors: [Lana Wachowski](#) (as The Wachowski Brothers), [Lilly Wachowski](#) (as The Wachowski Brothers)

Writers: [Lilly Wachowski](#) (as The Wachowski Brothers), [Lana Wachowski](#) (as The Wachowski Brothers)

Stars: [Keanu Reeves](#), [Laurence Fishburne](#), [Carrie-Anne Moss](#) | [See full cast & crew »](#)

73 Metascore | [Reviews](#) From metacritic.com | 3,655 user | 312 critic |  Popularity 286 (♦ 24)

Top Rated Movies #18 | Won 4 Oscars. Another 33 wins & 43 nominations. [See more awards »](#)

Figure: Web page of the movie The Matrix from IMDb (<http://www.imdb.com/title/tt0133093/>).

Example

Name	The Matrix
Genre	Action, Sci-Fi
Synopsis	A computer hacker learns from mysterious rebels about the true nature of his reality and his role in the war against its controllers.
Directors	Lana Wachowski (as The Wachowski Brothers), Lilly Wachowski (as The Wachowski Brothers)
Writers	Lilly Wachowski (as The Wachowski Brothers), Lana Wachowski (as The Wachowski Brothers)
Stars	Keanu Reeves, Laurence Fishburne, Carrie-Anne Moss
Catch-all	The Matrix Action, Sci-Fi A computer hacker learns from mysterious rebels about the true nature of his reality and his role in the war against its controllers. Lana Wachowski (as The Wachowski Brothers), Lilly Wachowski (as The Wachowski Brothers) Lilly Wachowski (as The Wachowski Brothers), Lana Wachowski (as The Wachowski Brothers) Keanu Reeves, Laurence Fishburne, Carrie-Anne Moss

Question

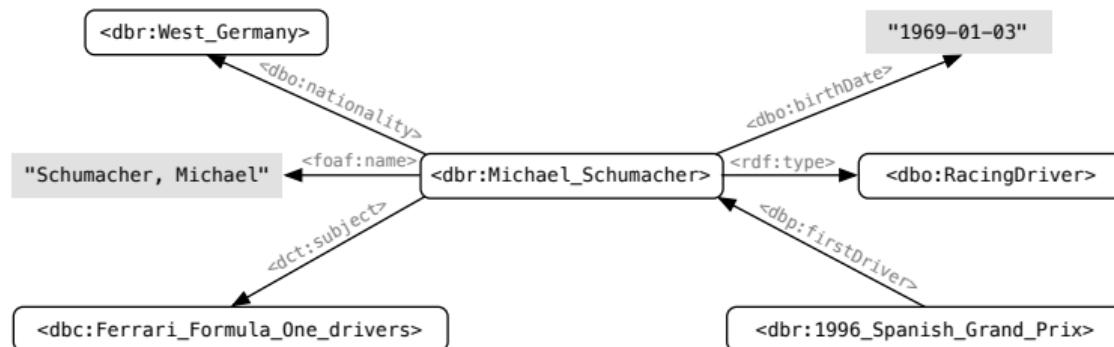
What is the role of the catch-all field?

Catch-all field

- Amasses the contents of all fields
 - Can help to quickly filter entities (e.g., in first-pass retrieval)
 - Fields are often sparse; combining field-level scores with an entity-level (“catch-all” score) often improve performance

From structured knowledge bases

- Assemble text from all SPO triples that are about a given entity
 - Note that the entity may also stand as object



Question

How to turn SPO triples into a fielded document?

Issue #1

- The number of potential fields is huge (in the 1000s)
 - The representation of an entity is sparse (each entity has only a handful of predicates)
 - Estimating field weights becomes problematic
- Solution: *predicate folding*
 - Grouping predicates together into a small set of predefined categories
 - Grouping may be based on predicate type or (manually determined) importance

Commonly used fields

- **Name** contains the name(s) of the entity
 - The two main predicates mapped to this field are <foaf:name> and <rdfs:label>
 - One might follow a simple heuristic and additionally consider all predicates ending with “name,” “label,” or “title”
- **Name variants** (aliases) may be aggregated in a separate field
 - In DBpedia, such variants may be collected via Wikipedia redirects (via <dbo:wikiPageRedirects>) and disambiguations (using <dbo:wikiPageDisambiguates>)
- **Attributes** includes all objects with literal values, except the ones already included in the name field
 - In some cases, the name of the predicate may also be included along with the value, e.g., “founding date 1964” (vs. just the value part, “1964”)

Commonly used fields (2)

- **Types** holds all types (categories, classes, etc.) to which the entity is assigned
 - Commonly, <rdf:type> is used for types
 - In DBpedia, <dct:subject> is used for assigning Wikipedia categories, which may also be considered as entity types
- **Outgoing relations** contains all URI objects, i.e., names of entities (or resources in general) that the subject entity links to
 - If the *types* or *name variants* fields are used then those predicates are excluded
 - Values might be prefixed with the predicate name, e.g., “spouse Michelle Obama”
- **Incoming relations** is made up of subject URIs from all SPO triples where the entity appears as object
- **Top predicates** may be considered as individual fields
 - E.g., top-100 most frequent DBpedia predicates
- **Catch-all** is a field that amasses all textual content related to the entity

Issue #2

- Object values are either URIs or literals
- While literals can be treated as regular text, URIs are not suitable for text-based search
 - Some URIs are “user-friendly”: http://dbpedia.org/resource/Audi_A4
 - Others are not: <http://rdf.freebase.com/ns/m.030qmx>
- *URI resolution* is the process of finding the corresponding human-readable name/label for a URI

URI resolution

- Goal: find the name/label for a URI
- The specific predicate that holds the name of a resource depends on the RDF vocabulary used
 - Commonly, <foaf:name> or <rdfs:label> are used
- Given an SPO triple, for example

```
<dbr:Audi_A4> <rdf:type> <dbo:MeanOfTransportation>
```

- The corresponding resource's name is contained in the object element of this triple:

```
<dbo:MeanOfTransportation> <rdfs:label> "mean of transportation"
```

Example

Name	Audi A4
Name variants	Audi A4 ... Audi A4 Allroad
Attributes	The Audi A4 is a compact executive car produced since late 1994 by the German car manufacturer Audi, a subsidiary of the Volkswagen Group [...] ... 1996 ... 2002 ... 2005 ... 2007
Types	Product ... Front wheel drive vehicles ... Compact executive cars ... All wheel drive vehicles
Outgoing relations	Volkswagen Passat (B5) ... Audi 80
Incoming relations	Audi A5
<foaf:name>	Audi A4
<dbo:abstract>	The Audi A4 is a compact executive car produced since late 1994 by the German car manufacturer Audi, a subsidiary of the Volkswagen Group [...]
Catch-all	Audi A4 ... Audi A4 ... Audi A4 Allroad ... The Audi A4 is a compact executive car produced since late 1994 by the German car manufacturer Audi, a subsidiary of the Volkswagen Group [...] ... 1996 ... 2002 ... 2005 ... 2007 ... Product ... Front wheel drive vehicles ... Compact executive cars ... All wheel drive vehicles ... Volkswagen Passat (B5) ... Audi 80 ... Audi A5

Exercise

E6-2 DBpedia Trivia

Entity retrieval

- Constructing term-based entity representations
- Ranking term-based entity representations

Models for entity ranking

- Unstructured retrieval models
 - LM, BM25, **SDM**
- Fielded retrieval models
 - MLM, BM25F, **PRMS**, **FSDM**

Markov random field (MRF) models

- Models so far employed a bag-of-words representation of both entities and queries
 - The order of terms is ignored
- The *Markov random field* (MRF) model provides a sound theoretical framework for modeling term dependence
 - Term dependencies are represented as a Markov random field (undirected graph G)
 - The MRF ranking function is computed as a linear combination of feature functions over the set of cliques² in G :

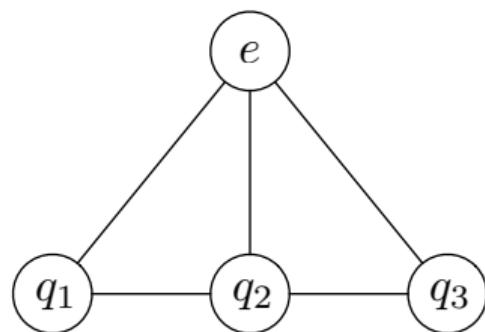
$$P_{\Lambda}(e|q) \stackrel{\text{rank}}{=} \sum_{c \in \mathcal{C}_G} \lambda_c f(c)$$

- MRF approaches belong to the more general class of linear feature-based models

²A clique is a subset of vertices of an undirected graph, such that every two distinct vertices are adjacent.

Sequential Dependence Model (SDM)

- The *sequential dependence model* (SDM) is one particular instantiation of the MRF model
 - SDM assumes dependence between neighboring query terms
 - Strikes a good balance between effectiveness and efficiency
- Graph consists of an entity node e and query nodes q_i
- Two types of cliques
 - Query term and the entity (unigram matches)
 - Two query terms and the entity; two variants:
 - The query terms occur contiguously (ordered bigram match)
 - They do not (unordered bigram match)



Sequential Dependence Model (SDM)

- The SDM ranking function is given by a weighted combination of three feature functions
 - Query terms (f_T)
 - Exact match of query bigrams (f_O)
 - Unordered match of query bigrams (f_U)

$$\text{score}(e, q) = \lambda_T \sum_{i=1}^n f_T(q_i, e) + \lambda_O \sum_{i=1}^{n-1} f_O(q_i, q_{i+1}, e) + \lambda_U \sum_{i=1}^{n-1} f_U(q_i, q_{i+1}, e)$$

- The query is represented as a sequence of terms $q = \langle q_1, \dots, q_n \rangle$
- Feature weights are subject to the constraint $\lambda_T + \lambda_O + \lambda_U = 1$
 - Recommended default setting: $\lambda_T = 0.85$, $\lambda_O = 0.1$, and $\lambda_U = 0.05$

Feature functions

- Feature functions are based on language modeling estimates using Dirichlet prior smoothing
- *Unigram matches* are based on smoothed entity language models:

$$f_T(q_i, e) = \log P(q_i | \theta_e)$$

Feature functions (cont'd)

- *Ordered bigram matches:*

$$f_O(q_i, q_{i+1}, e) = \log \left(\frac{c_o(q_i, q_{i+1}, e) + \mu P_o(q_i, q_{i+1} | \mathcal{E})}{l_e + \mu} \right)$$

- $c_o(q_i, q_{i+1}, e)$ denotes the number of times the terms q_i, q_{i+1} occur in this exact order in the description of e
- l_e is the length of the entity's description (number of terms)
- \mathcal{E} is the entity catalog (set of all entities)
- μ is the smoothing parameter
- The background language model is a maximum likelihood estimate:

$$P_o(q_i, q_{i+1} | \mathcal{E}) = \frac{\sum_{e \in \mathcal{E}} c_o(q_i, q_{i+1}, e)}{\sum_{e \in \mathcal{E}} l_e} .$$

Feature functions (cont'd)

- *Unordered bigram matches:*

$$f_U(q_i, q_{i+1}, e) = \log \left(\frac{c_w(q_i, q_{i+1}, e) + \mu P_w(q_i, q_{i+1} | \mathcal{E})}{l_e + \mu} \right) ,$$

- $c_w(q_i, q_{i+1}, e)$ counts the co-occurrence of terms q_i and q_{i+1} in e , within an unordered window of w term positions
- Typically, a window size of 8 is used (corresponds roughly to sentence-level proximity)
- l_e is the length of the entity's description (number of terms)
- \mathcal{E} is the entity catalog (set of all entities)
- μ is the smoothing parameter
- The background language model is a maximum likelihood estimate:

$$P_w(q_i, q_{i+1} | \mathcal{E}) = \frac{\sum_{e \in \mathcal{E}} c_w(q_i, q_{i+1}, e)}{\sum_{e \in \mathcal{E}} l_e} .$$

Example

Entity description	a b c a b c c d e f b c f e g
Query (q)	a b c
<i>Ordered bigram matches:</i> $c_o(q_i, q_{i+1}, e)$	
$c_o(a, b, e)$	2
$c_o(b, c, e)$	3 \Rightarrow a b c a b c c d e f b c f e g a b c a b c c d e f b c f e g a b c a b c c d e f b c f e g
<i>Unordered bigram matches:</i> $c_w(q_i, q_{i+1}, e)$ ($w = 5$)	
$w_o(a, b, e)$	4
$w_o(b, c, e)$	7 \Rightarrow a b c a b c c d e f b c f e g a b c a b c c d e f b c f e g a b c a b c c d e f b c f e g a b c a b c c d e f b c f e g a b c a b c c d e f b c f e g a b c a b c c d e f b c f e g a b c a b c c d e f b c f e g

Question

How would you implement the SDM scoring function on top of Elasticsearch?

Exercise

E6-3 Counting Bigram Matches

Models for entity ranking

- ~~Unstructured retrieval models~~
 - ~~LM, BM25, SDM~~
- Fielded retrieval models ⇌
 - MLM, BM25F, **PRMS**, **FSDM**

Mixture of Language Models (MLM)

- Idea: Build a separate language model for each field, then take a linear combination of them

$$P(t|\theta_d) = \sum_i w_i P(t|\theta_{d_i})$$

- where
 - i corresponds to the field index
 - w_i is the field weight (such that $\sum_i w_i = 1$)
 - $P(t|\theta_{d_i})$ is the field language model

Probabilistic Retrieval Model for Semistructured data (PRMS)

- Extension to MLM for dynamic field weighting
- To key ideas
 - Instead of using a fixed (static) field weight for all terms, field weights are determined dynamically on a term-by-term basis
 - Field weights can be established based on the term distributions of the respective fields
- Replace the static weight w_i with a *mapping probability* $P(f|t)$

$$P(t|\theta_d) = \sum_f P(f|t)P(t|\theta_{df})$$

- Note: we now use field f instead of index i when referring to fields

Estimating the mapping probability

- By applying Bayes' theorem and using the law of total probability:

$$P(f|t) = \frac{P(t|f)P(f)}{P(t)} = \frac{P(t|f)P(f)}{\sum_{f' \in \mathcal{F}} P(t|f')P(f')}$$

- where
 - $P(f)$ is a prior that can be used to incorporate, for example, domain-specific background knowledge, or left to be uniform
 - $P(t|f)$ is conveniently estimated using the background language model of that field $P(t|C_f)$

Example

$t = ``\text{Meg}''$		$t = ``\text{Ryan}''$		$t = ``\text{war}''$		$t = ``\text{redemption}''$	
f	$P(f t)$	f	$P(f t)$	f	$P(f t)$	f	$P(f t)$
cast	0.407	cast	0.601	genre	0.927	title	0.983
team	0.381	team	0.381	title	0.070	location	0.017
title	0.187	title	0.017	location	0.002	year	0.000

Table: Example mapping probabilities computed on the IMDB collection, taken from Kim et al., 2009.

Fielded Sequential Dependence Model (FSDM)

- Idea: base the feature function estimates on term/bigram frequencies combined across multiple fields (in the spirit of MLM and BM25F)
- The *fielded sequential dependence model* (FSDM) we present here combines SDM and MLM
- *Unigram matches* are MLM-estimated probabilities:

$$f_T(q_i, e) = \log \sum_{f \in \mathcal{F}} w_f^T P(t | \theta_{f_e})$$

- w_f^T are the field mapping weights (for each field)

Feature functions (cont'd)

- *Unordered bigram matches:*

$$f_O(q_i, q_{i+1}, e) = \log \sum_{f \in \mathcal{F}} w_f^O \frac{c_o(q_i, q_{i+1}, f_e) + \mu_f P_o(q_i, q_{i+1} | f_e)}{l_{f_e} + \mu_f}$$

- *Ordered bigram matches:*

$$f_U(q_i, q_{i+1}, e) = \log \sum_{f \in \mathcal{F}} w_f^U \frac{c_u^w(q_i, q_{i+1}, f_e) + \mu_f P_u^w(q_i, q_{i+1} | f_e)}{l_{f_e} + \mu_f}$$

- All background models and smoothing parameters are made field-specific
 - But the same smoothing parameter (μ_f) may be used for all types of matches
- w_f^O and w_f^U are the field mapping weights (for each field)
 - May be based on the field mapping probability estimates from PRMS

Summary

- Entities
 - Named entities vs. concepts
 - Properties of entities (unique IDs, names, types, attributes, relationships)
 - Entity-oriented search vs. semantic search
 - Entity catalog, knowledge repository, knowledge base/graph
- Data sources
 - Wikipedia article structure
 - Resource Description Framework (RDF)
 - Knowledge bases (DBpedia, Wikidata, LOD)
- The ad hoc entity retrieval task
- Constructing entity description documents
 - From unstructured, semi-structured, and structured sources
 - Working with SPO triples (predicate folding and URI resolution)
- Entity ranking
 - Using standard document retrieval models (LM, BM25, MLM, BM25F)
 - Sequential Dependence Models (SDM) and fielded variant (FSDM)
 - Probabilistic Retrieval Model for Semi-Structured Data (PRMS)

Reading

- Entity-Oriented Search (Balog)
 - Chapters 1 and 2
 - Chapter 3, until 3.3.2 (inclusive)

Entity Linking

[DAT640] Information Retrieval and Text Mining

Ivica Kostric & Krisztian Balog
University of Stavanger

September 19, 2022



CC BY 4.0

In this module

1. Entity linking
2. Entity linking and retrieval

Entity linking

Entity linking

- Task: recognizing entity mentions in text and linking them to the corresponding entries in a knowledge base (KB)
 - Limited to recognizing entities for which a target entry exists in the reference KB; each KB entry is a candidate
 - It is assumed that the document provides sufficient context for disambiguating entities

Entity linking in action



The DBpedia Spotlight interface shows entity linking results for the text "First documented in the 13th century, Berlin was the capital of the Kingdom of Prussia (1701–1918), the German Empire (1871–1918), the Weimar Republic (1919–33) and the Third Reich (1933–45). Berlin in the 1920s was the third largest municipality in the world. After World War II, the city became divided into East Berlin -- the capital of East Germany -- and West Berlin, a West German enclave surrounded by the Berlin Wall from 1961–89. Following German reunification in 1990, the city regained its status as the capital of Germany, hosting 147 foreign embassies." The interface includes a confidence slider at 0.5, language set to English, and options for n-best candidates, select types, and annotate.

Confidence: 0.5

Language: English

n-best candidates [SELECT TYPES...](#) [ANNOTATE](#)

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

[BACK TO TEXT](#)

Entity linking in action



The DBpedia Spotlight interface shows entity linking results for the text: "First documented in the 13th century, Berlin was the capital of the Kingdom of Prussia (1701–1918), the German Empire (1871–1918), the Weimar Republic (1919–33) and the Third Reich (1933–45). Berlin in the 1920s was the third largest municipality in the world. After World War II, the city became divided into East Berlin -- the capital of East Germany -- and West Berlin, a West German enclave surrounded by the Berlin Wall from 1961–89. Following German reunification in 1990, the city regained its status as the capital of Germany, hosting 147 foreign embassies." The interface includes a confidence slider set at 0.5, a language dropdown set to English, and a checkbox for 'n-best candidates' which is unchecked. Buttons for 'SELECT TYPES...' and 'ANNOTATE' are also present. A 'BACK TO TEXT' button is located at the bottom right.

Confidence: 0.5

Language: English

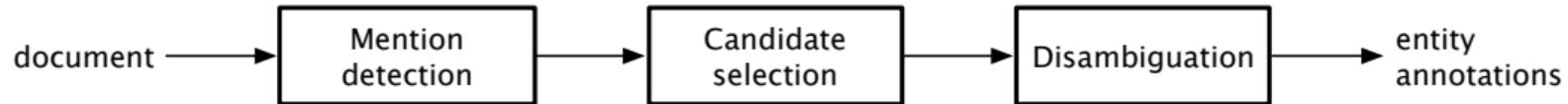
n-best candidates

SELECT TYPES... ANNOTATE

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

BACK TO TEXT

Anatomy of an entity linking system

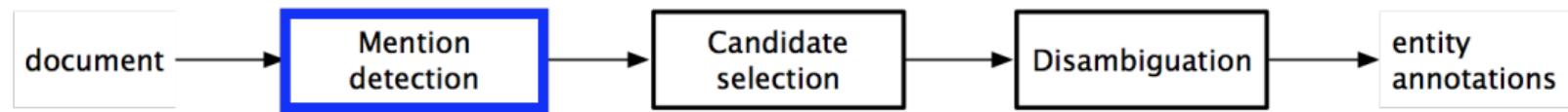


- **Mention detection:** Identification of text snippets that can potentially be linked to entities
- **Candidate selection:** Generating a set of candidate entities for each mention
- **Disambiguation:** Selecting a single entity (or none) for each mention, based on the context

Entity linking

- Mention detection
- Candidate selection
- Disambiguation
- Evaluation

Mention detection



Mention detection

- Goal: Detect all “linkable” phrases
- Challenges
 - Recall oriented
 - Do not miss any entity that should be linked
 - Find entity name variants
 - E.g. “jlo” is name variant of Jennifer Lopez
 - Filter out inappropriate ones
 - E.g. “new york” matches >2k different entities

Common approach

1. Build a dictionary of entity surface forms
 - o Entities with all names variants
2. Check all document n-grams against the dictionary
 - o The value of n is set typically between 6 and 8
3. Filter out undesired entities
 - o Can be done here or later in the pipeline

Example

Surface form dictionary

Surface form S Entities \mathcal{E}_S

Empire	British_Empire Empire_(magazine) First_French_Empire Galactic_Empire_(Star_Wars) Holy_Roman_Empire Roman_Empire ...
Empire State	Empire_State_(band) Empire_State_Building Empire_State_Film_Festival
Empire State Building	Empire_State_Building
...	...
Times Square	Times_Square Times_Square_(Hong_Kong) Times_Square_(IRT_42nd_Street_Shuttle) ...
...	...

Home to the **Empire State Building**, **Times Square**, **Statue of Liberty** and other iconic sites,
New York City is a fast-paced, globally influential center of art, culture, fashion and finance.

Surface form dictionary construction from Wikipedia

- **Page title**
 - Canonical (most common) name of the entity

The screenshot shows a web browser displaying the English Wikipedia article for "United Kingdom". The title "United Kingdom" is highlighted with a red box. The page content discusses the United Kingdom of Great Britain and Northern Ireland, its location off the north-western coast of Europe, and its status as a sovereign state. It mentions the country's population, capital city (London), and its place in the European Union. The page also notes the relationships between the four countries of the UK and its historical formation.

United Kingdom - Wikipedia, the free encyclopedia

Article Talk

United Kingdom

From Wikipedia, the free encyclopedia

This article is about the country. For the island, see [Great Britain](#). For other uses, see [United Kingdom \(disambiguation\)](#).

The United Kingdom of Great Britain and Northern Ireland, commonly known as the United Kingdom (UK) or Britain. Lying off the north-western coast of the European mainland, it includes the island of Great Britain—a term encompassing England, Scotland and Wales—and many smaller islands.^[3] Northern Ireland is the land border with another state—the Republic of Ireland.^[nb 6] Apart from this land border, the UK is surrounded by sea: to its east, the English Channel to its south and the Celtic Sea to its south-southwest. The Irish Sea lies between Great Britain and Ireland. The British Isles as a whole have an area of 93,800 square miles (243,000 km²), making it the 80th-largest sovereign state in the world.

The United Kingdom is the 22nd-most populous country, with an estimated 64.5 million inhabitants.^[4] It is a constitutional monarchy and a unitary parliamentary democracy. Its capital city is London, an important global city and financial centre with 10,310,000, the fourth-largest in Europe and second-largest in the European Union.^[11] The current monarch is Queen Elizabeth II. The UK consists of four countries: England, Scotland, Wales, and Northern Ireland.^[12] The latter three each with varying powers,^{[14][15]} based in their capitals, Edinburgh, Cardiff, and Belfast, respectively. The nearby and [Bailiwick of Jersey](#) are not part of the United Kingdom, being Crown dependencies with the British Government's international representation.^[16]

The relationships among the countries of the United Kingdom have changed over time. Wales was annexed by the Acts of Union of 1536 and 1543. A treaty between England and Scotland resulted in 1707 in a unified Kingdom of Great Britain, which joined with the Kingdom of Ireland to form the United Kingdom of Great Britain and Ireland. In 1922, five-sixths of Ireland seceded to form the present formulation of the United Kingdom of Great Britain and Northern Ireland.^[nb 7] The UK has fourteen overseas territories, remnants of the British Empire which, at its height in the 1920s, encompassed almost a quarter of the world's land area.

Surface form dictionary construction from Wikipedia

- Page title
- Redirect pages
 - Alternative names that are frequently used to refer to an entity

The screenshot shows a Wikipedia article page for "United Kingdom". The page title is "United Kingdom" in bold black text. Below it, a red box highlights the text "(Redirected from UK)". The main content area starts with a summary line: "This article is about the country. For the island, see Great Britain. For other uses, see United Kingdom (disambiguation)." The text continues with a detailed description of the United Kingdom's geography, history, and political structure. On the left side, there is a sidebar with links to "Main page", "Contents", "Featured content", "Current events", "Random article", "Donate to Wikipedia", "Wikipedia store", "Interaction", "Help", "About Wikipedia", "Community portal", "Recent changes", "Contact page", and "Tools".

United Kingdom

(Redirected from UK)

This article is about the country. For the island, see Great Britain. For other uses, see United Kingdom (disambiguation).

The United Kingdom of Great Britain and Northern Ireland, commonly known as the United Kingdom (UK) or Britain. Lying off the north-western coast of the European mainland, it includes the island of Great Britain—a whole country—plus the north-eastern part of the island of Ireland and many smaller islands.^[8] Northern Ireland is the land border with another state—the Republic of Ireland.^[nb 6] Apart from this land border, the UK is surrounded by the North Sea to its east, the English Channel to its south and the Celtic Sea to its south-southwest. The Irish Sea lies between the two islands. The country has an area of 93,800 square miles (243,000 km²), making it the 80th-largest sovereign state in the world.

The United Kingdom is the 22nd-most populous country, with an estimated 64.5 million inhabitants.^[4] It is a constitutional monarchy and a unitary parliamentary state of governance.^{[9][10]} Its capital city is London, an important global city and financial centre with 10,310,000, the fourth-largest in Europe and second-largest in the European Union.^[11] The current monarch is Queen Elizabeth II. The UK consists of four countries: England, Scotland, Wales, and Northern Ireland.^[12] The latter three each with varying powers,^{[14][15]} based in their capitals, Edinburgh, Cardiff, and Belfast, respectively. The nearby Isle of Wight and the Bailiwick of Jersey are not part of the United Kingdom, being Crown dependencies with the British Government's international representation.^[16]

The relationships among the countries of the United Kingdom have changed over time. Wales was annexed by the Acts of Union of 1536 and 1543. A treaty between England and Scotland resulted in 1707 in a unified Kingdom of Great Britain, which included the Kingdom of Ireland to form the United Kingdom of Great Britain and Ireland. In 1922, five-sixths of Ireland seceded to form the Irish Free State, leaving the present formulation of the United Kingdom of Great Britain and Northern Ireland.^[nb 7] The UK has fourteen C

Surface form dictionary construction from Wikipedia

- Page title
- Redirect pages
- **Disambiguation pages**
 - List of entities that share the same name



The screenshot shows a web browser window displaying the Wikipedia disambiguation page for "United Kingdom". The URL in the address bar is en.wikipedia.org. The page title is "United Kingdom (disambiguation)". The main content area includes a DNA helix icon and the text "European Science Photo Competition 2015". Below the title, it states: "The United Kingdom is a sovereign state located off the north-western coast of continental Europe." It also notes that "United Kingdom" may also refer to:

- United Kingdom (album)
- Political union, a state formed out of smaller states
 - United Kingdom of Great Britain and Ireland, the United Kingdom from 1801 to 1922
 - United Kingdom of Great Britain, a sovereign state from 1707 to 1800
 - United Kingdom of England, a state from the 10th century to 1707
 - Kingdom of the Netherlands, a sovereign state with territory in western Europe and the Caribbean
 - United Kingdom of the Netherlands, the Kingdom of the Netherlands from 1815 to 1830
 - United Kingdom of Libya, a sovereign state from 1951 to 1969
 - Kingdom of Portugal, a state from 1139 to 1910
 - United Kingdom of Portugal, Brazil and the Algarves, the Kingdom of Portugal from 1815 to 1825

Below this, there is a "See also" section with links to "Terminology of the British Isles", "United Kingdoms", and "British (disambiguation)".

Surface form dictionary construction from Wikipedia

- Page title
- Redirect pages
- Disambiguation pages
- **Anchor texts**
 - of links pointing to the entity's Wikipedia page

The screenshot shows a Mac OS X desktop with a window open to the English Wikipedia article for 'Ireland'. The window title bar says 'en.wikipedia.org'. The main content area displays the article's text, starting with a summary about the island in Europe. A red box highlights the word 'United Kingdom' in the text. The left sidebar contains a navigation menu with links like 'Main page', 'Contents', and 'Interaction'.

Ireland
From Wikipedia, the free encyclopedia

This article is about the island in Europe. For the sovereign state of the same name, [Ireland](#). For other uses, see [Ireland \(disambiguation\)](#).

Ireland (Irish: *Éire* [eːɾə] (listen); Ulster-Scots: *Airlann* [ɑːrlən]) is an island separated from Great Britain to its east by the North Channel, the Irish Sea, and St George's Channel. It is the second-largest island of the British Isles, the third-largest in Europe, and the twentieth-largest island in the world. Politically, Ireland is divided between the Republic of Ireland (officially named Ireland), which covers most of the island, and Northern Ireland, which is part of the United Kingdom, in the northeast corner. The population of Ireland was about 6.4 million, ranking it the second-most populous island in the world. Just under 4.6 million live in the Republic of Ireland and just over 1.8 million live in Northern Ireland.

The island's geography comprises relatively low-lying mountains surrounding a central plain, with rivers extending inland. The island has lush vegetation, a product of its mild but changeable climate. Thick woodlands covered the island until the Middle Ages. As that is wooded in Ireland is about 11% of the total, compared with a European average of 33%. The climate is temperate, with twenty-six extant mammal species native to Ireland.^[10] The Irish climate is very moderate, with temperatures ranging from 5°C to 15°C. As a result, winters are milder than expected for such a northerly area. However, the island's position in the Atlantic Ocean means that it experiences more extreme weather than those in Continental Europe. Rainfall and cloud cover are abundant.

The earliest evidence of human presence in Ireland is dated at 10,500 BC.^[12] Gaelic Ireland was first mentioned in the 1st century AD by the Roman historian Tacitus.

Surface form dictionary construction from Wikipedia

- Page title
- Redirect pages
- Disambiguation pages
- Anchor texts
- **Bold texts from first paragraph**
 - generally denote other name variants of the entity



Surface form dictionary construction from other sources

- Anchor texts from external web pages pointing to Wikipedia articles
- Problem of *synonym discovery*
 - Expanding acronyms
 - Leveraging search results or query-click logs from a web search engine
 - ...

Filtering mentions

- Objective is to filter our mentions that are unlikely to be linked to any entity
- **Keyphraseness**

$$P(\text{keyphrase}|m) = \frac{|D_{\text{link}}(m)|}{|D(m)|}$$

- $|D_{\text{link}}(m)|$ is the number of Wikipedia articles where m appears as an anchor text of a link
- $|D(m)|$ is the number of Wikipedia articles that contain m

Filtering mentions (cont'd)

- **Link probability**

$$P(\text{link}|m) = \frac{\text{link}(m)}{\text{freq}(m)}$$

- $\text{link}(m)$ is the number of times mention m appears as an anchor text of a link
- $\text{freq}(m)$ is the total number of times mention m occurs in Wikipedia (as a link or not)

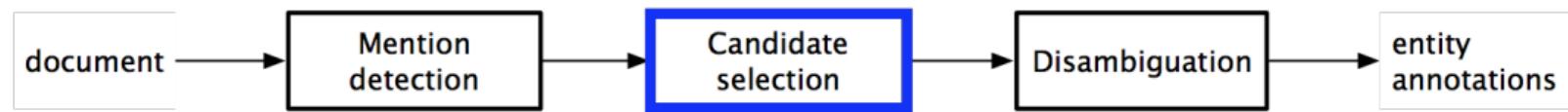
Overlapping entity mentions

- Dealing with them in this phase
 - E.g., by dropping a mention if it is subsumed by another mention
- Keeping them and postponing the decision to a later stage (candidate selection or disambiguation)

Entity linking

- Mention detection
- Candidate selection
- Disambiguation
- Evaluation

Candidate selection



Candidate selection

- Goal: Narrow down the space of disambiguation possibilities
- Balances between precision and recall (effectiveness vs. efficiency)
- Often approached as a ranking problem
 - Keeping only candidates above a score/rank threshold for downstream processing

Commonness

- Perform the ranking of candidate entities based on their overall popularity, i.e., “most common sense”

$$P(e|m) = \frac{n(m, e)}{\sum_{e' \in \mathcal{E}} n(m, e')}$$

- $n(m, e)$ the number of times entity e is the link destination of mention m
- Can be pre-computed and stored in the entity surface form dictionary
- Follows a power law with a long tail of extremely unlikely senses; entities at the tail end of the distribution can be safely discarded
 - E.g., 0.001 is a sensible threshold

Example

Entity	Commonness
e	$P(e m)$

Times_Square	0.940
Times_Square_(film)	0.017
Times_Square_(Hong_Kong)	0.011
Times_Square_(IRT_42nd_Street_Shuttle)	0.006
...	...



Home to the **Empire State Building**, **Times Square**, **Statue of Liberty** and other iconic sites,
New York City is a fast-paced, globally influential center of art, culture, fashion and finance.

Example #2

Bulgaria's best **World Cup** performance was in the **1994 World Cup** where they beat **Germany**, to reach the semi-finals, losing to Italy, and finishing in fourth ...

Entity	Commonness
FIFA_World_Cup	0.2358
FIS_Apline_Ski_World_Cup	0.0682
2009_FINA_Swimming_World_Cup	0.0633
World_Cup_(men's_golf)	0.0622
...	

Entity	Commonness
1998_FIFA_World_Cup	0.9556
1998_IAAF_World_Cup	0.0296
1998_Alpine_Skiing_World_Cup	0.0059
...	

Entity	Commonness
Germany	0.9417
Germany_national_football_team	0.0139
Nazi_Germany	0.0081
German_Empire	0.0065
...	

- Commonness works in many of the cases, but not in all
- Other entities help to disambiguate which entity is being referred to

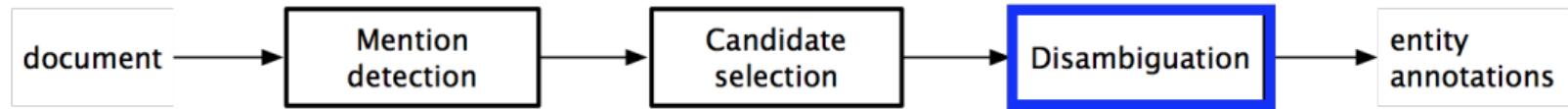
Exercise

E7-1 Entity linking

Entity linking

- Mention detection
- Candidate selection
- Disambiguation
- Evaluation

Disambiguation



Disambiguation

- Baseline approach: most common sense
- Consider additional types of evidence
 - **Prior importance** of entities and mentions
 - **Contextual similarity** between the text surrounding the mention and the candidate entity
 - **Coherence** among all entity linking decisions in the document
- Combine these signals
 - Using supervised learning or graph-based approaches
- Optionally perform pruning
 - Reject low confidence or semantically meaningless annotations

Prior importance features

- **Context-independent features**
 - Neither the text nor other mentions in the document are taken into account
- Keyphraseness
- Link probability
- Commonness

Prior importance features (cont'd)

- **Link prior**

- Popularity of the entity measured in terms of incoming links

$$P_{link}(e) = \frac{|\mathcal{L}_e|}{\sum_{e' \in \mathcal{E}} |\mathcal{L}_{e'}|}$$

- $|\mathcal{L}_e|$ is the total number of incoming links entity e has

- **Page views**

- Popularity of the entity measured in terms traffic volume

$$P_{pageviews}(e) = \frac{\text{pageviews}(e)}{\sum_{e' \in \mathcal{E}} \text{pageviews}(e')}$$

- $\text{pageviews}(e)$ is the total number of page views (measured over a certain time period)

Contextual features

- Compare the surrounding *context* of a mention with the (textual) representation of the given candidate entity
- Context of a mention
 - Window of text (sentence, paragraph) around the mention
 - Entire document
- Entity's representation
 - Wikipedia entity page, first description paragraph, terms with highest TF-IDF score, etc.
 - Entity's description in the knowledge base

Contextual similarity

- Commonly: bag-of-words representation
- **Cosine similarity**

$$sim_{cos}(m, e) = \frac{\vec{d}_m \cdot \vec{d}_e}{\|\vec{d}_m\| \|\vec{d}_e\|}$$

- Many other options for measuring similarity
 - Dot product, KL divergence, Jaccard similarity
- Representation does not have to be limited to bag-of-words
 - Concept vectors (named entities, Wikipedia categories, anchor text, keyphrases, etc.)

Entity-relatedness features

- It can reasonably be assumed that a document focuses on one or at most a few topics
- Therefore, entities mentioned in a document should be topically related to each other
- Capturing *topical coherence* by developing some measure of *relatedness* between (linked) entities
 - Defined for pairs of entities

Wikipedia Link-based Measure (WSM)

- Often referred to simply as *relatedness*
- A close relationship is assumed between two entities if there is a large overlap between the entities linking to them

$$WLM(e, e') = 1 - \frac{\log(\max(|\mathcal{L}_e|, |\mathcal{L}_{e'}|)) - \log(|\mathcal{L}_e \cap \mathcal{L}_{e'}|)}{\log(|\mathcal{E}|) - \log(\min(|\mathcal{L}_e|, |\mathcal{L}_{e'}|))}$$

- \mathcal{L}_e is the set of entities that link to e
- $|\mathcal{E}|$ is the total number of entities

Wikipedia Link-based Measure (WSM)

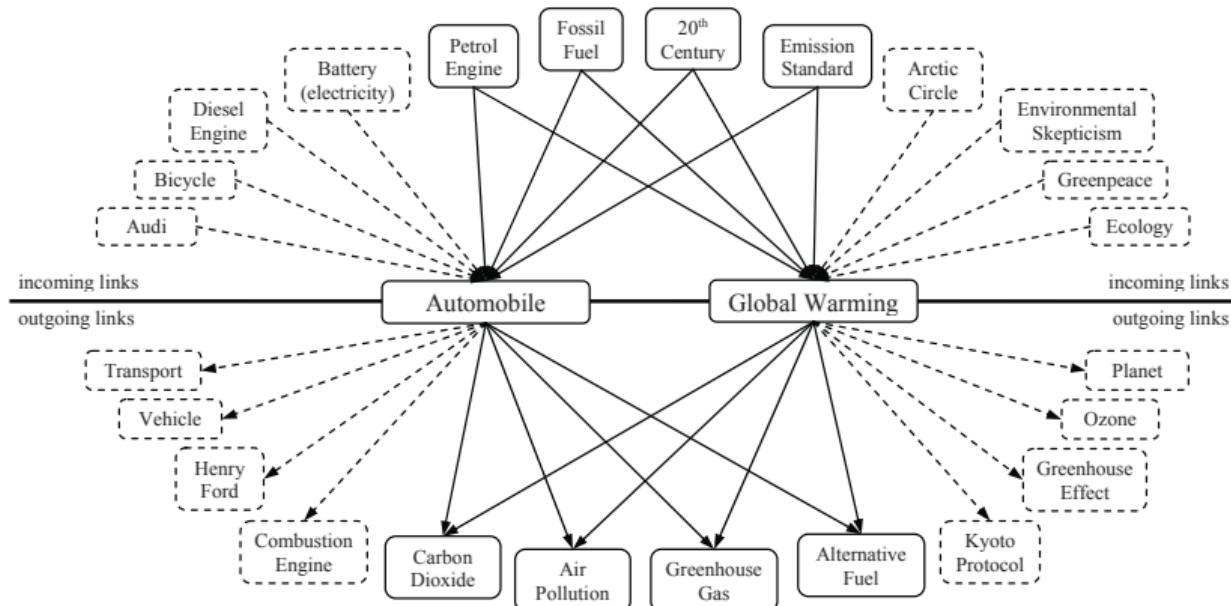


Figure: Image taken from Milne and Witten (2008). An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links. In: AAAI WikiAI Workshop.

Entity-relatedness features

- Numerous ways to define relatedness
 - Consider not only incoming, but also outgoing links or the union of incoming and outgoing links
 - Jaccard similarity, Pointwise Mutual Information (PMI), or the Chi-square statistic, etc.
- A relatedness function does not have to be symmetric
 - E.g., the relatedness of the UNITED STATES given NEIL ARMSTRONG is intuitively larger than the relatedness of NEIL ARMSTRONG given the UNITED STATES
 - **Conditional probability**

$$P(e'|e) = \frac{|\mathcal{L}_{e'} \cap \mathcal{L}_e|}{|\mathcal{L}_e|}$$

- Having a single relatedness function is preferred, to keep the disambiguation process simple
- Various relatedness measures can effectively be combined into a single score using a machine learning approach

Disambiguation approaches

- Consider *local compatibility* (including prior evidence) and *coherence* with the other entity linking decisions
- Overall objective function:

$$\Gamma^* = \arg \max_{\Gamma} \left(\sum_{(m,e) \in \Gamma} \phi(m, e) + \psi(\Gamma) \right)$$

- $\phi(m, e)$ is the local compatibility between the mention and the assigned entity
 - $\psi(\Gamma)$ is the coherence function for all entity annotations in the document
 - Γ is a solution (set of mention-entity pairs)
- **This optimization problem is NP-hard!**
 - Need to resort to approximation algorithms and heuristics

Disambiguation strategies

- **Individually**, one-mention-at-a-time
 - Rank candidates for each mention, take the top ranked one (or NIL)
 - Interdependence between entity linking decisions may be incorporated in a pairwise fashion

$$\Gamma(m) = \arg \max_{e \in \mathcal{E}_m} score(e, m)$$

- **Collectively**, all mentions in the document jointly

Disambiguation approaches

Approach	Context	Entity interdependence
Most common sense	none	none
Individual local disambiguation	text	none
Individual global disambiguation	text & entities	pairwise
Collective disambiguation	text & entities	collective

Individual local disambiguation

- Early entity linking approaches
- Local compatibility score can be written as a linear combination of features

$$\phi(e, m) = \sum_i \lambda_i f_i(e, m)$$

- $f_i(e, m)$ can be either a context-independent or a context-dependent feature
- Learn the “optimal” combination of features from training data using machine learning

Individual global disambiguation

- Consider what other entities are mentioned in the document
- True global optimization would be NP-hard
- Good approximation can be computed efficiently by considering pairwise interdependencies for each mention independently
 - Pairwise entity relatedness scores need to be aggregated into a single number (how coherent the given candidate entity is with the rest of the entities in the document)

TAGME (Ferragina & Scaiella, 2010)

- Combine the two most important features (*commonness* and *relatedness*) using a voting scheme
- The score of a candidate entity for a particular mention:

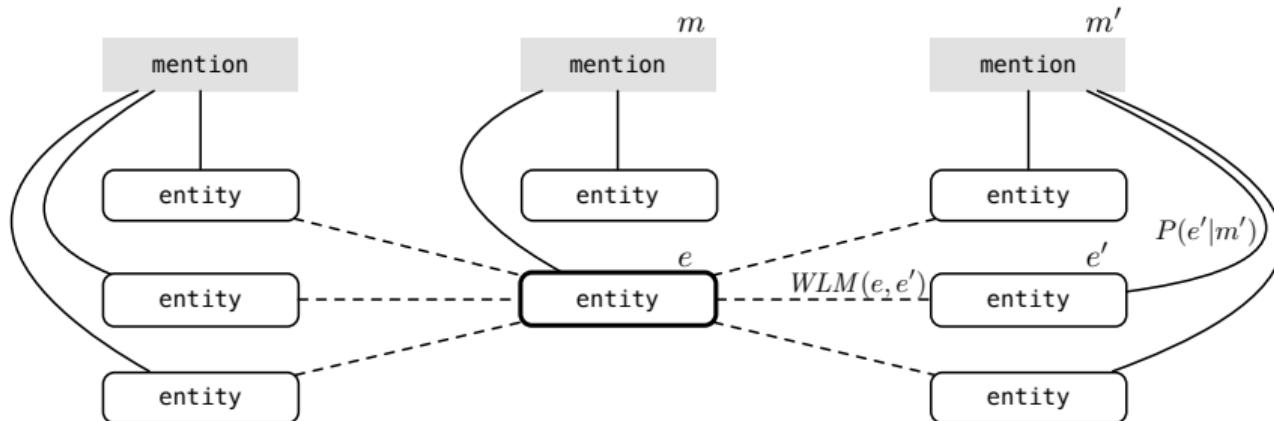
$$score(e, m) = \sum_{\substack{m' \in \mathcal{M}_d \\ m' \neq m}} vote(m', e)$$

- The vote function estimates the agreement between e and all candidate entities of all other mentions in the document

TAGME (voting mechanism)

- Average relatedness between each possible disambiguation, weighted by its commonness score

$$vote(m', e) = \frac{\sum_{e' \in \mathcal{E}_{m'}} WLM(e, e') P(e'|m')}{|\mathcal{E}_{m'}|}$$



TAGME (final score)

- Final decision uses a simple but robust heuristic
 - The top entities with the highest score are considered for a given mention and the one with the highest commonness score is selected

$$\Gamma(m) = \arg \max_{e \in \mathcal{E}_m} \{P(e|m) : e \in \text{top}_\epsilon[\text{score}(e, m)]\}$$

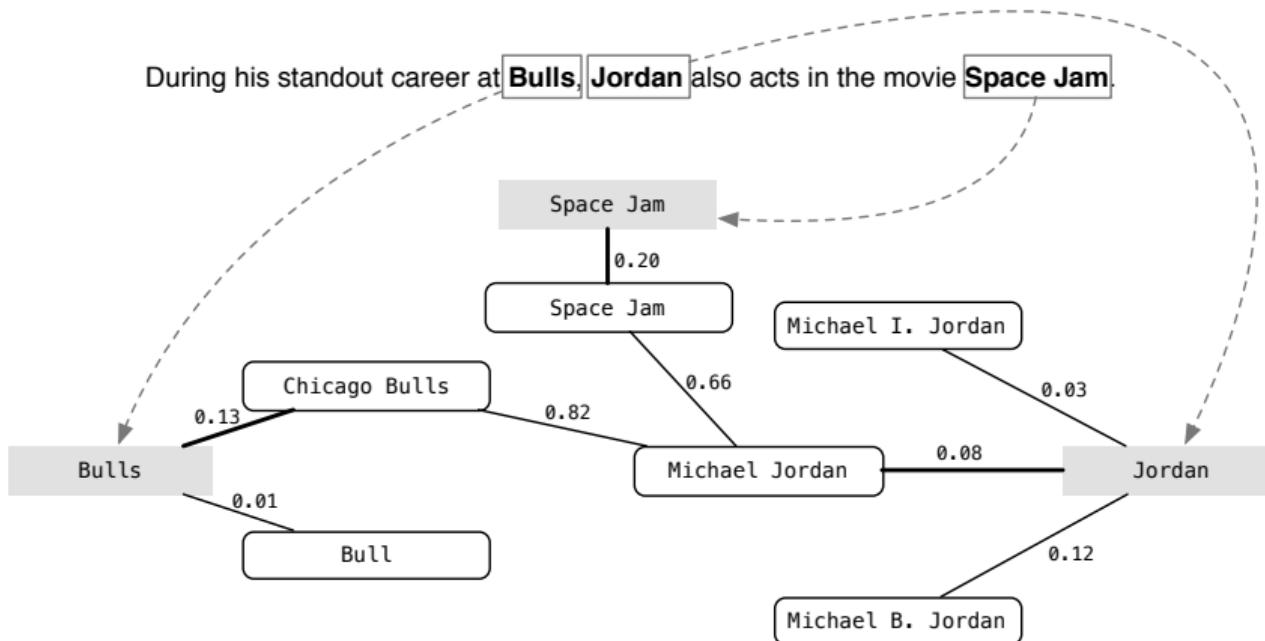
- Note that *score* merely acts as a filter
 - Only entities in the top ϵ percent of the scores are retained ($\epsilon = 0.3$)
 - Out of the remaining entities, the most common sense of the mention will be finally selected

Collective disambiguation

- Graph-based representation
- **Mention-entity edges** capture the local compatibility between the mention and the entity
 - Measured using a combination of context-independent and context-dependent features
- **Entity-entity edges** represent the semantic relatedness between a pair of entities
 - Common choice is *relatedness* (WLM)
- Use these relations jointly to identify a single referent entity (or none) for each of the mentions

Example

During his standout career at **Bulls**, **Jordan** also acts in the movie **Space Jam**.



AIDA (Hoffart et al., 2011)

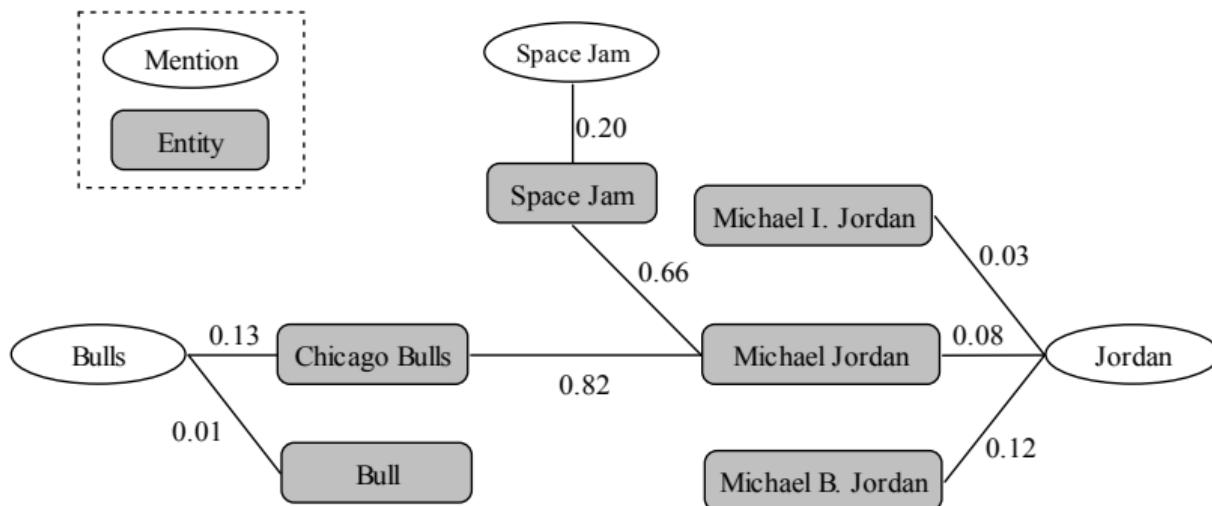
- Problem formulation: find a dense subgraph that contains all mention nodes and exactly one mention-entity edge for each mention
- Greedy algorithm iteratively removes edges

AIDA algorithm

- Start with the full graph
- Iteratively remove the entity node with the lowest *weighted degree* (along with all its incident edges), provided that each mention node remains connected to at least one entity
 - Weighted degree of an entity node is the sum of the weights of its incident edges
- The graph with the highest *density* is kept as the solution
 - The density of the graph is measured as the minimum weighted degree among its entity nodes

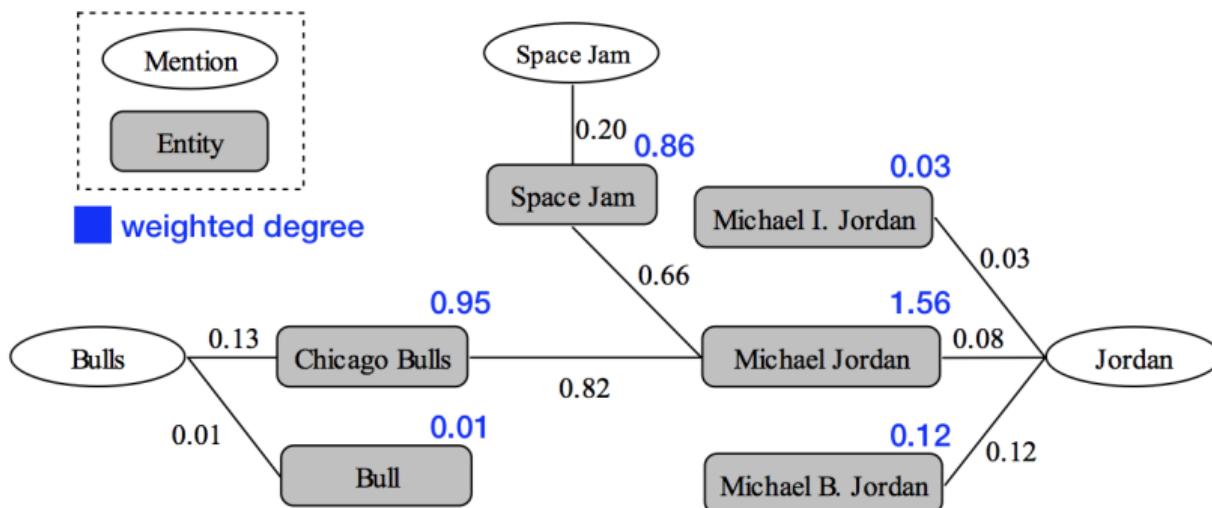
Example iteration #1

- Which entity should be removed first?



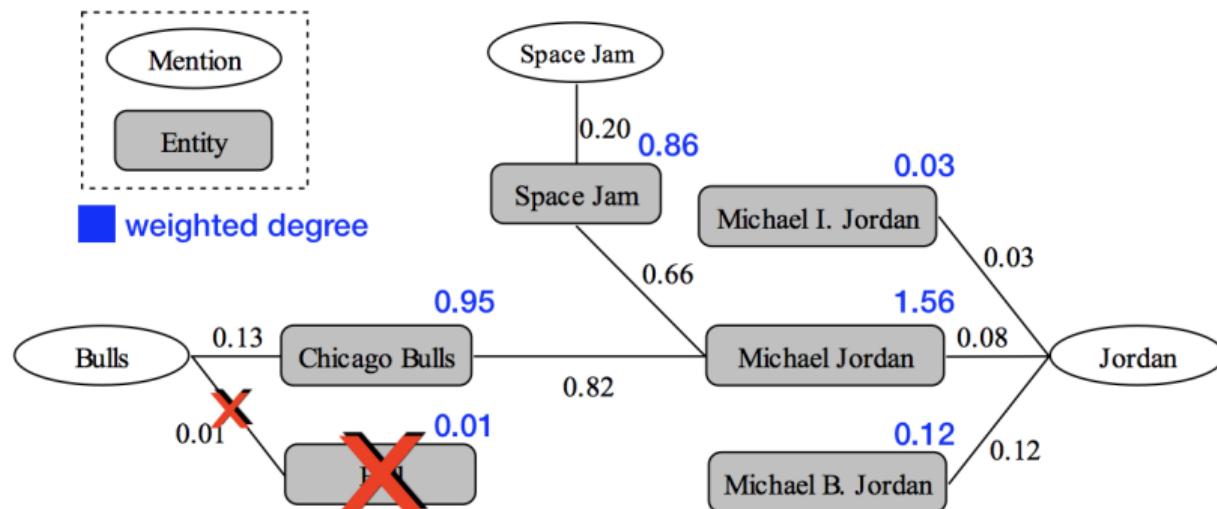
Example iteration #1

- Which entity should be removed first?



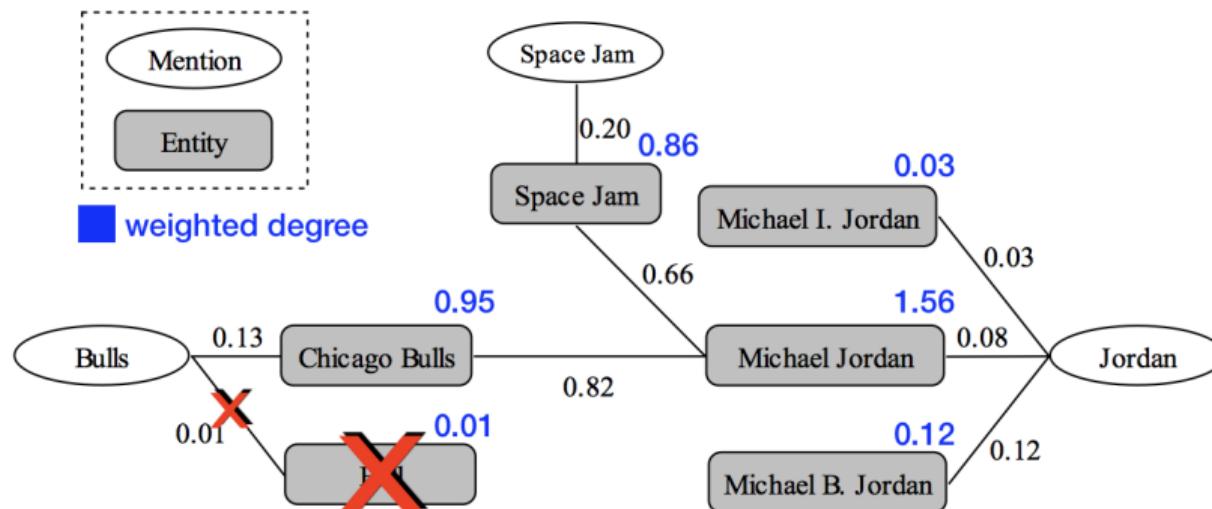
Example iteration #1

- Which entity should be removed first?



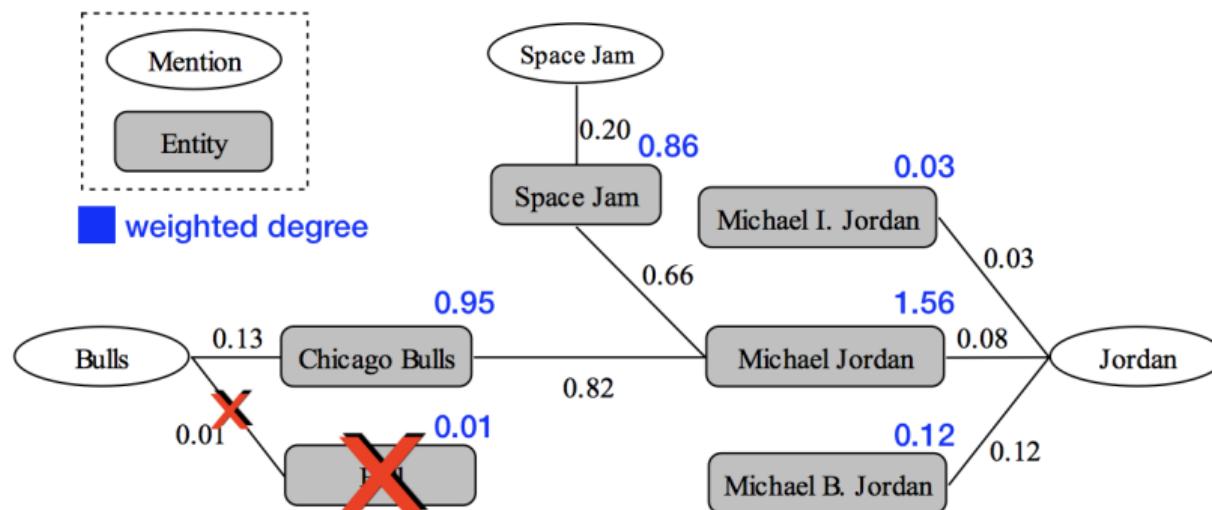
Example iteration #1

- What is the density of the graph?



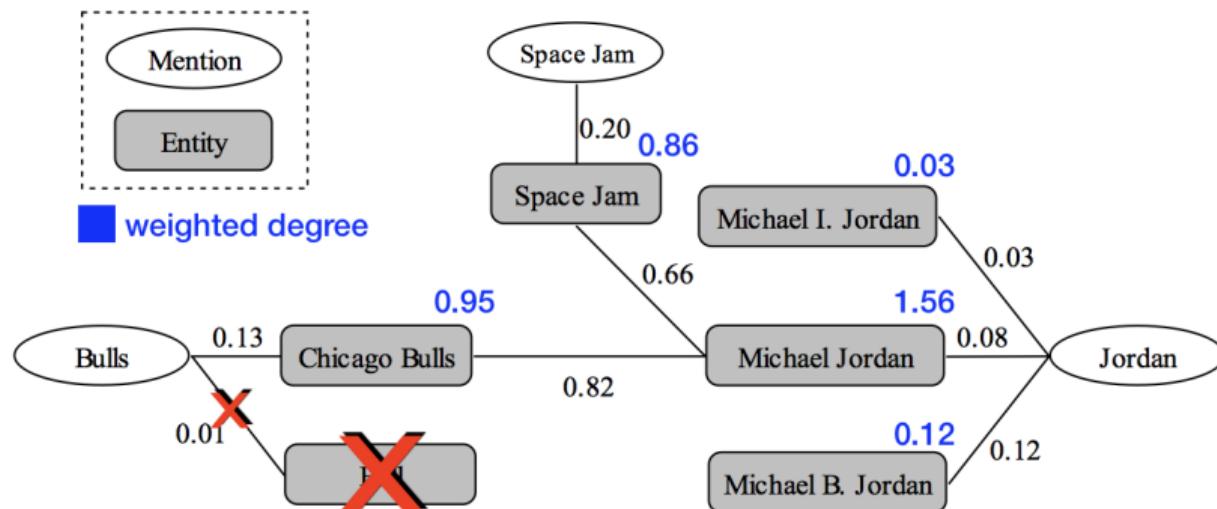
Example iteration #1

- What is the density of the graph? **0.03**



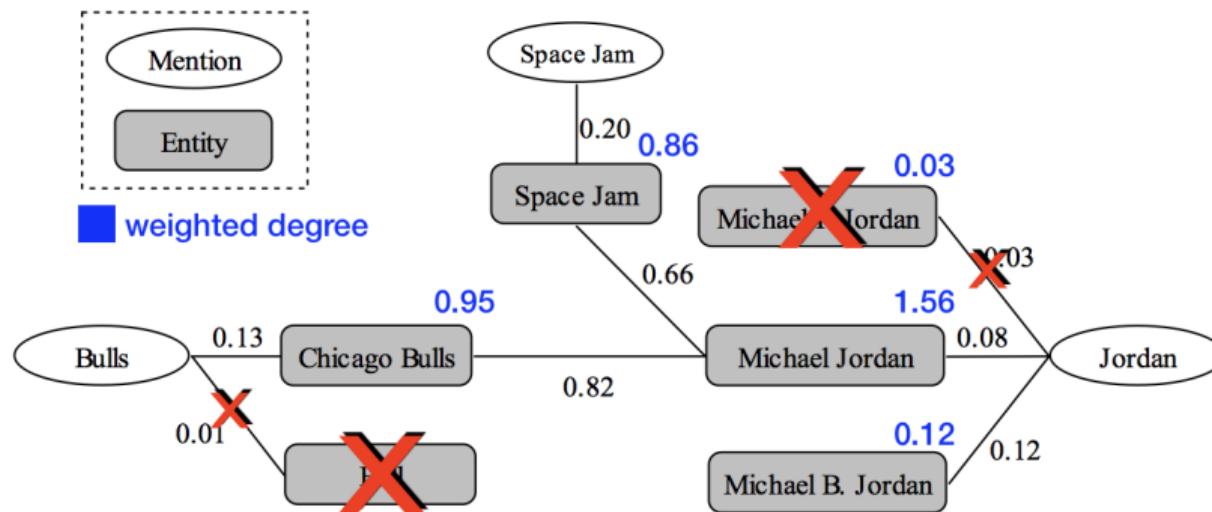
Example iteration #2

- Which entity should be removed next?



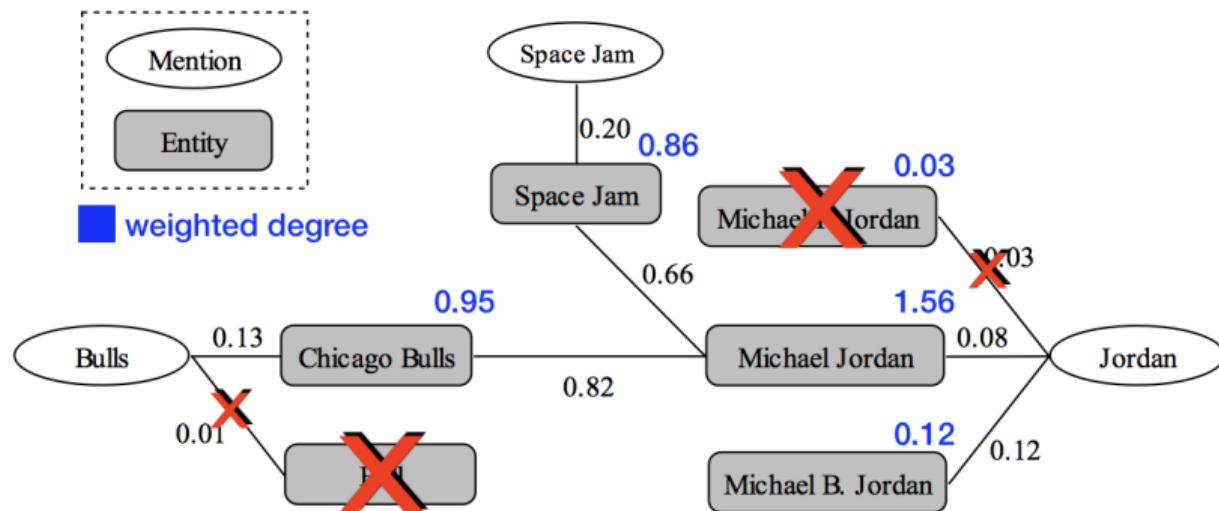
Example iteration #2

- Which entity should be removed next?



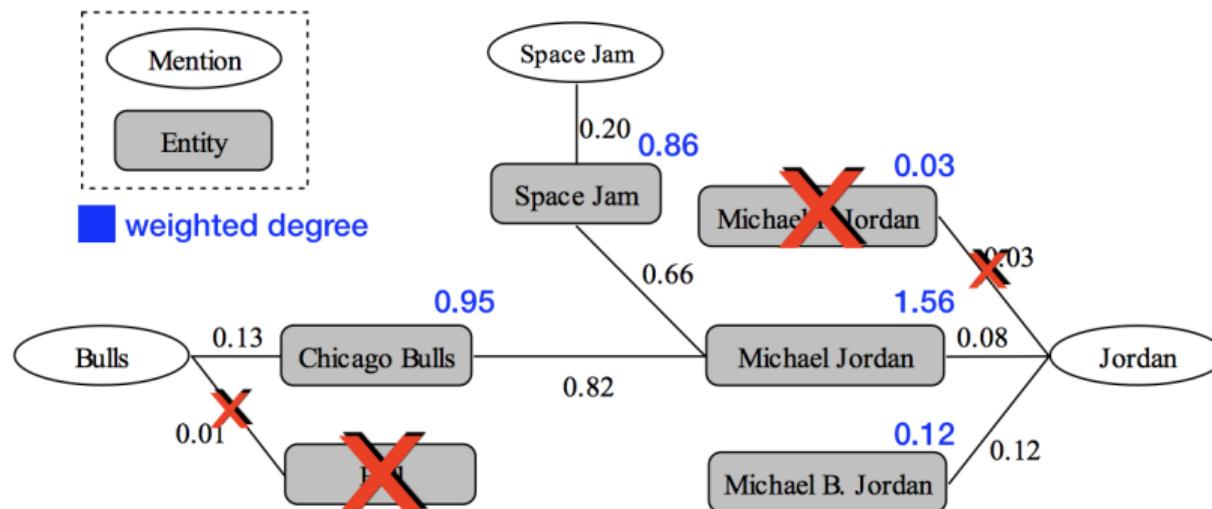
Example iteration #2

- What is the density of the graph?



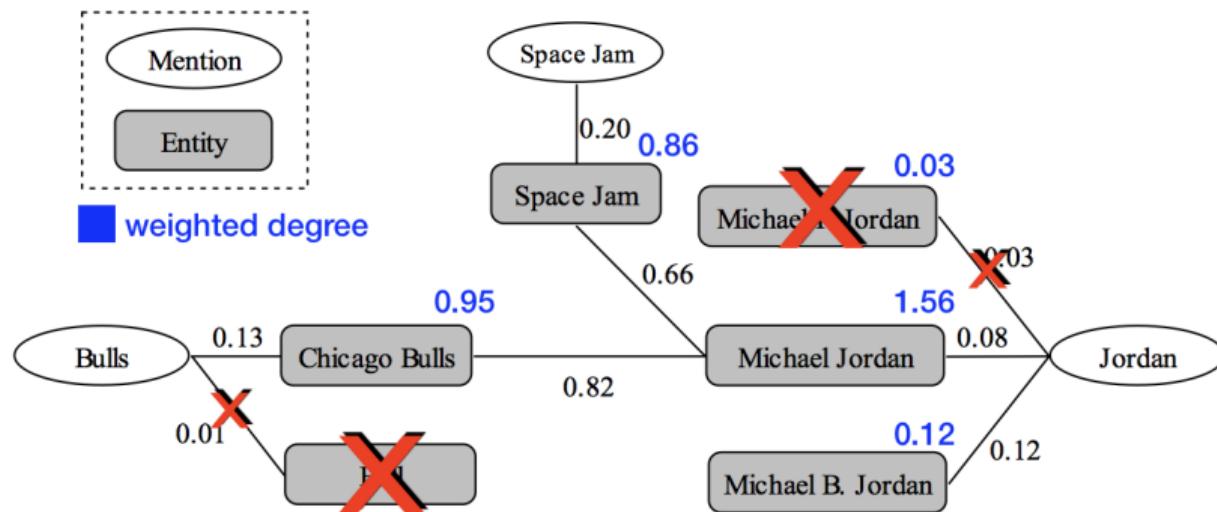
Example iteration #2

- What is the density of the graph? **0.12**



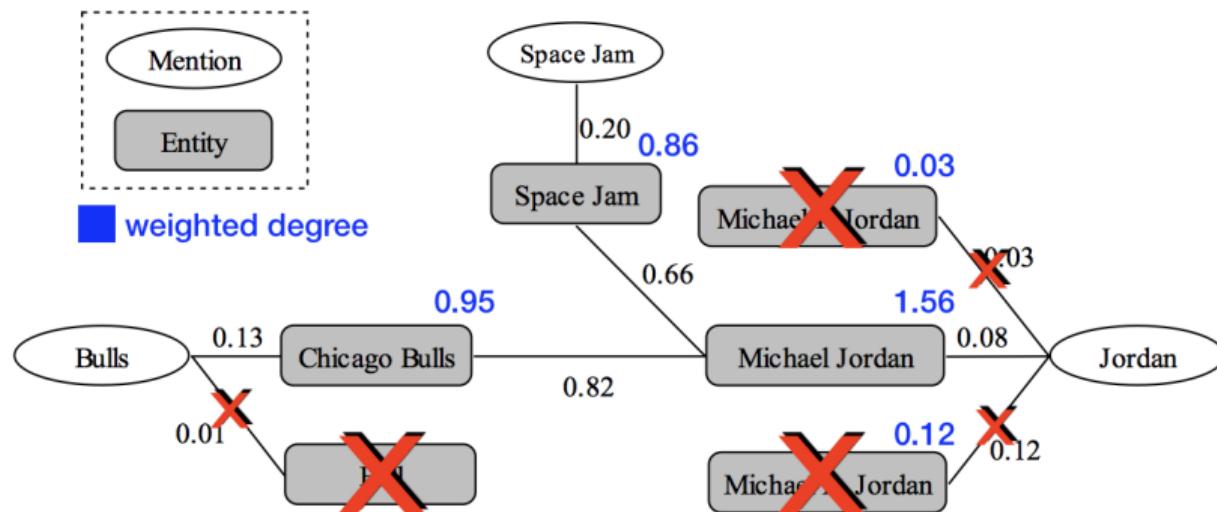
Example iteration #3

- Which entity should be removed next?



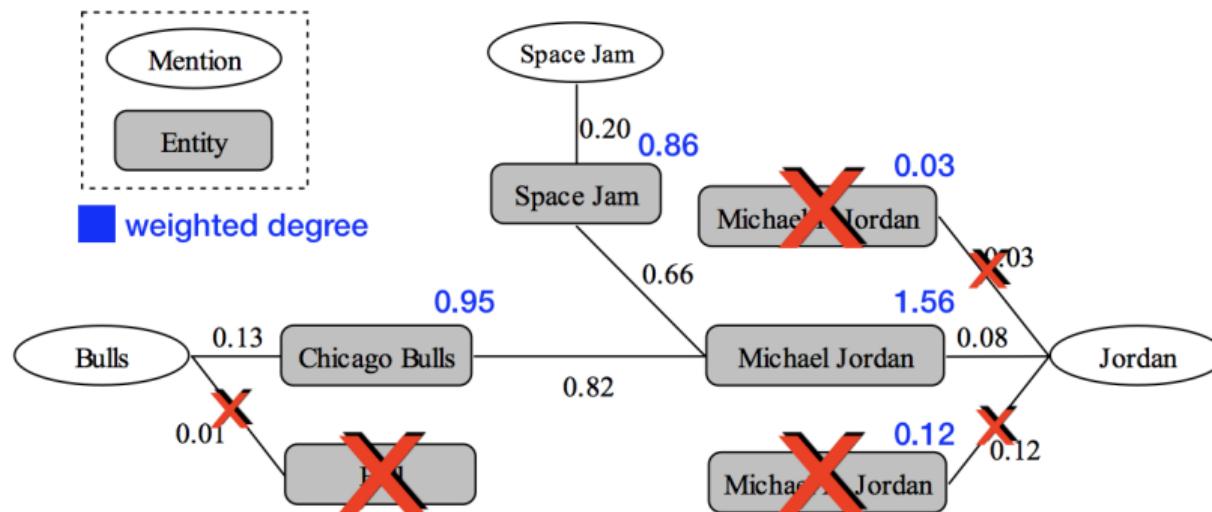
Example iteration #3

- Which entity should be removed next?



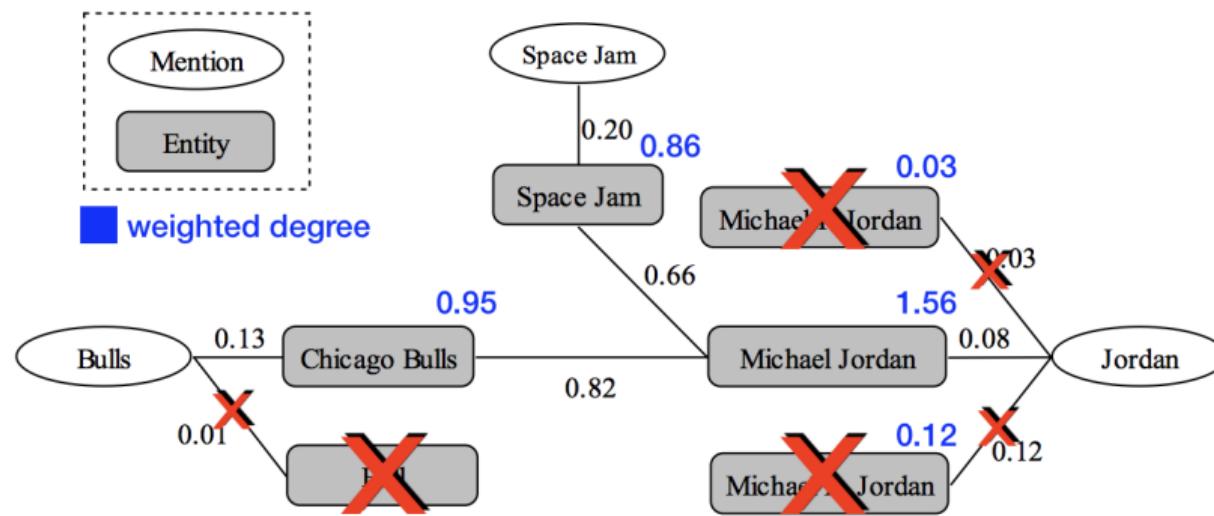
Example iteration #3

- What is the density of the graph?



Example iteration #3

- What is the density of the graph? **0.86**



AIDA pre- and post-processing

- Pre-processing phase: remove entities that are “too distant” from the mention nodes
- At the end of the iterations, the solution graph may still contain mentions that are connected to more than one entity; deal with this in post-processing
 - If the graph is sufficiently small, it is feasible to exhaustively consider all possible mention-entity pairs
 - Otherwise, a faster local (hill-climbing) search algorithm may be used

Pruning

- Discarding meaningless or low-confidence annotations produced by the disambiguation phase
- Simplest solution: use a confidence threshold
- More advanced solutions
 - Machine learned classifier to retain only entities that are “relevant enough” (human editor would annotate them)
 - Optimization problem: decide, for each mention, whether switching the top ranked disambiguation to NIL would improve the objective function

Entity linking

- Mention detection
- Candidate selection
- Disambiguation
- Evaluation

Evaluation (end-to-end)

- Comparing the system-generated annotations against a human-annotated gold standard
- Evaluation criteria
 - **Perfect match:** both the linked entity and the mention offsets must match
 - **Relaxed match:** the linked entity must match, it is sufficient if the mention overlaps with the gold standard

Evaluation with relaxed match

Example #1

ground truth	system annotation
<p>Košice is the biggest city in eastern Slovakia and in 2013 was the European Capital of Culture together with Marseille, France. It is situated on the river Hornád at the eastern reaches of the Slovak Ore Mountains, near the border with Hungary.</p> 	<p>Košice is the biggest city in eastern Slovakia and in 2013 was the European Capital of Culture together with Marseille, France. It is situated on the river Hornád at the eastern reaches of the Slovak Ore Mountains, near the border with Hungary.</p>  

Example #2

ground truth	system annotation
<p>Košice is the biggest city in eastern Slovakia and in 2013 was the European Capital of Culture together with Marseille, France. It is situated on the river Hornád at the eastern reaches of the Slovak Ore Mountains, near the border with Hungary.</p>  <p>Slovak Ore Mountains Mountain range in Slovakia</p>  	<p>Košice is the biggest city in eastern Slovakia and in 2013 was the European Capital of Culture together with Marseille, France. It is situated on the river Hornád at the eastern reaches of the Slovak Ore Mountains, near the border with Hungary.</p> <p>Slovakia Country in Europe</p> 

Evaluation metrics

- Set-based metrics
 - **Precision:** fraction of correctly linked entities that have been annotated by the system
 - **Recall:** fraction of correctly linked entities that should be annotated
 - **F-measure:** harmonic mean of precision and recall
- Metrics are computed over a collection of documents
 - Micro-averaged: aggregated across mentions
 - Macro-averaged: aggregated across documents

Evaluation metrics

- **Micro-averaged**

$$P_{mic} = \frac{|\mathcal{A}_{\mathcal{D}} \cap \hat{\mathcal{A}}_{\mathcal{D}}|}{|\mathcal{A}_{\mathcal{D}}|}$$

$$R_{mic} = \frac{|\mathcal{A}_{\mathcal{D}} \cap \hat{\mathcal{A}}_{\mathcal{D}}|}{|\hat{\mathcal{A}}_{\mathcal{D}}|}$$

- $\mathcal{A}_{\mathcal{D}}$ include all annotations for a set \mathcal{D} of documents
- $\hat{\mathcal{A}}_{\mathcal{D}}$ is the collection of reference annotations for \mathcal{D}

- **Macro-averaged**

$$P_{mac} = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \frac{|\mathcal{A}_d \cap \hat{\mathcal{A}}_d|}{|\mathcal{A}_d|}$$

$$R_{mac} = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \frac{|\mathcal{A}_d \cap \hat{\mathcal{A}}_d|}{|\hat{\mathcal{A}}_d|}$$

- \mathcal{A}_d are the annotations generated by the entity linking system
- $\hat{\mathcal{A}}_d$ denote the reference (ground truth) annotations for a single document d

- **F1 score**

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

Component-based evaluation

- The pipeline architecture makes the evaluation of entity linking systems especially challenging
 - The main focus is on the disambiguation component, but its performance is largely influenced by the preceding steps
- Fair comparison between two approaches can only be made if they share all other elements of the pipeline

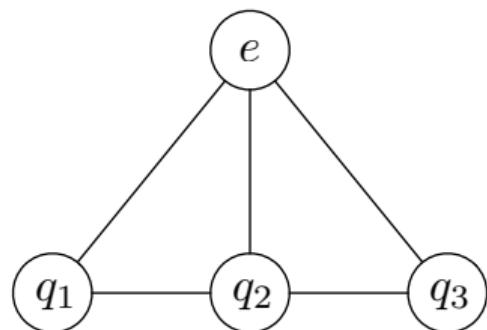
Exercise

E7-2 Entity linking evaluation

Entity linking and retrieval

Recap - Entity retrieval

- Two instantiations of Markov random field (MRF) models for modeling term dependence
 - SDM - Unstructured retrieval model
 - FSDM - Fielded retrieval model
- The approaches use unigram and bigram matches between terms
 - So far it is all term based.



Semantically enriched entity retrieval

- Working definition: semantics == structure
- Goal: enrich term-based representations by leveraging structured information about entities
- Important: this semantic enrichment needs to happen both on the query and on the entity side, as well as reflected in the matching (scoring function)

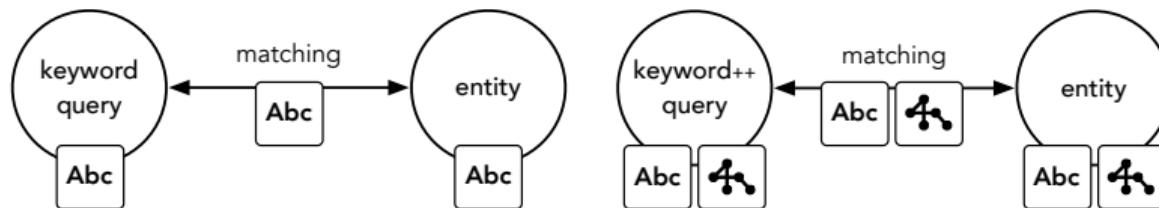


Figure: (Left) Ranking entities using term-based representations. (Right) Semantically enriched entity ranking by incorporating structure. Illustration is taken from Balog (2018) [Fig 4.1].

Question

How to incorporate entity linking information in entity retrieval?

Dual term-based and entity-based entity representations

Idea: preserve these entity references by employing a *dual entity representation*: on top of the traditional term-based representation, each entity is additionally represented by means of its associated entities.

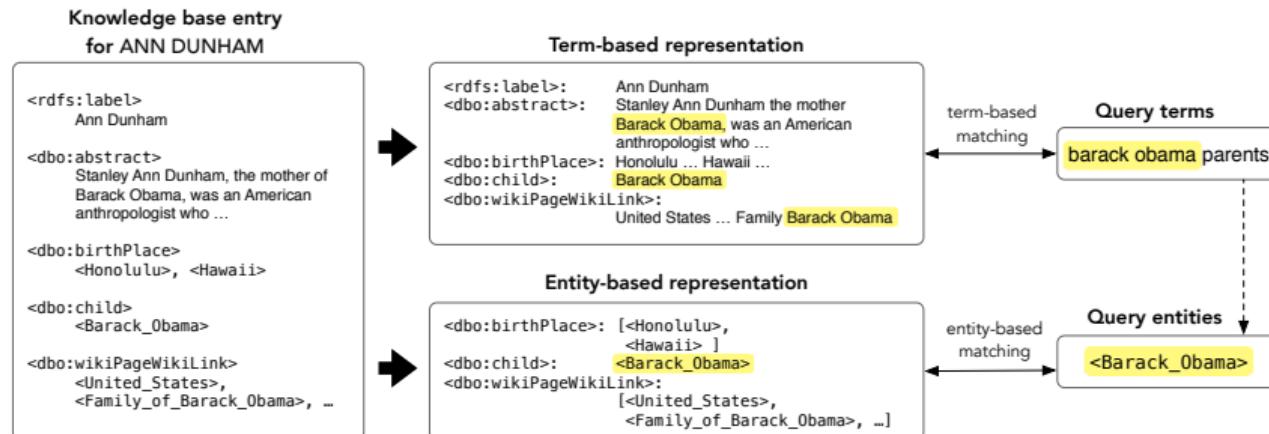
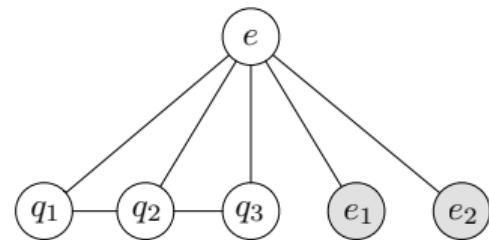


Figure: Illustration is based on (Hasibi et al., 2016).

Entity linking incorporated retrieval (ELR)

- Entity-specific extension applied on top of the MRF framework
 - May be applied on top of any term-based retrieval model that can be instantiated in the MRF framework, but here we will focus on SDM
- The underlying graph representation of SDM is extended with query entity nodes
 - It is assumed that entities have already been identified (linked) in queries and in the descriptions of entities
- New type of clique: 2-cliques between the given entity (that is being scored) and the query entities



Ranking function SDM+ELR

- The SDM+ELR ranking function is given by a weighted combination of four feature functions
 - Query terms (f_T)
 - Exact match of query bigrams (f_O)
 - Unordered match of query bigrams (f_U)
 - Entity matches (f_E)**

$$\begin{aligned} \text{score}(e, q) = & \lambda_T \sum_{i=1}^n f_T(q_i, e) + \lambda_O \sum_{i=1}^{n-1} f_O(q_i, q_{i+1}, e) \\ & + \lambda_U \sum_{i=1}^{n-1} f_U(q_i, q_{i+1}, e) + \lambda_E \sum_{i=1}^m f_E(e_i; e) \end{aligned}$$

- Issue: the number of entities (m) varies per query! How can λ_E be trained?

Parameterizing weights

- Rewriting λ parameters as parameterized functions over each clique:

$$\lambda_T(q_i) = \lambda_T \frac{1}{n} ,$$

$$\lambda_O(q_i, q_{i+1}) = \lambda_O \frac{1}{n-1} ,$$

$$\lambda_U(q_i, q_{i+1}) = \lambda_U \frac{1}{n-1} ,$$

$$\lambda_E(e_i) = \lambda_E \frac{w(e_i, q)}{\sum_{j=1}^m w(e_j, q)} .$$

- The weight $w(e_i, q)$ reflects the confidence in that entity annotation (entity linker's confidence score)

Ranking function SDM+ELR

- Final ranking function using the parameterized weights:

$$\begin{aligned} \text{score}(e, q) = & \frac{\lambda_T}{n} \sum_{i=1}^n f_T(q_i; e) \\ & + \frac{\lambda_O}{n-1} \sum_{i=1}^{n-1} f_O(q_i, q_{i+1}; e) \\ & + \frac{\lambda_U}{n-1} \sum_{i=1}^{n-1} f_U(q_i, q_{i+1}; e) \\ & + \frac{\lambda_{\mathcal{E}}}{\sum_{j=1}^m w(e_j, q)} \sum_{i=1}^m w(e_i, q) f_{\mathcal{E}}(e_i; e) \end{aligned}$$

- Default setting: $\lambda_T = 0.8$, $\lambda_O = 0.05$, $\lambda_U = 0.05$, and $\lambda_{\mathcal{E}} = 0.1$

Feature function for entity matches

- Differences from term-based scoring
 - We assume that each entity appears at most once in each field
 - If a query entity appears in a field, then it shall be regarded as a “perfect match,” independent of what other entities are present in the same field
 - Unlike for terms-based representations, predicate folding is not performed ()

$$f_{\mathcal{E}}(e_i; e) = \log \sum_{f \in \tilde{\mathcal{F}}} w_f^{\mathcal{E}} \left((1 - \lambda) \mathbb{1}(e_i, f_{\tilde{e}}) + \lambda \frac{\sum_{e' \in \mathcal{E}} \mathbb{1}(e_i, f_{\tilde{e}'})}{|\{e' \in \mathcal{E} : f_{\tilde{e}'} \neq \emptyset\}|} \right)$$

- \tilde{e} denote the entity-based representation of entity e
- $\tilde{\mathcal{F}}$ denotes the set of fields in the entity-based representation
- $\mathbb{1}(e, f_{\tilde{e}})$ is a binary indicator function, which is 1 if e_i is present in the entity field $f_{\tilde{e}}$ and otherwise 0
- λ is the smoothing parameter (default: 0.1)
- Field weights $w_f^{\mathcal{E}}$ may be set manually or via dynamic mapping using PRMS

Exercise

E7-3 Entity linking incorporated retrieval

Summary

- The entity linking task
- Canonical entity linking architecture and main components
- Mention detection
 - Entity surface form dictionary and its construction from Wikipedia
 - Filtering mentions (keyphraseness and link probability)
- Candidate selection (using commonness)
- Disambiguation
 - Prior importance features (incl. keyphraseness, link probability, commonness, link prior, page views)
 - Contextual features (contextual similarity and entity-relatedness features, incl. WSM, a.k.a. relatedness)
 - TAGME method
 - AIDA algorithm
- Evaluation with perfect/relaxed match, evaluation measures
- Dual entity representation, Entity Linking incorporated Retrieval (ELR) model

Reading

- Entity-Oriented Search (Balog)
 - Chapter 4: Sections 4.1 and 4.2.2
 - Chapter 5, until 5.6 (inclusive) + Section 5.8.1

Assignment

- First assignment in the second set
 - **Indexing (A2.1) (4 points)** ⇐
 - Scoring (A2.2) (7 points)
 - Evaluation (A2.3) (4 points)
- Deadline: 30. September
- Only packages in the requirements.txt are allowed.

Semantic search

[DAT640] Information Retrieval and Text Mining

Ivica Kostric & Krisztian Balog
University of Stavanger

September 26, 2022



CC BY 4.0

In this module

1. Understanding information needs
2. Leveraging entities in document retrieval

Understanding information needs

Understanding information needs

- Goal: infer a semantically enriched representation of the information need
- Various techniques
 - Classifying the query according to higher-level goals or intent
 - Segmenting the query and interpreting its structure
 - Identify the types or categories of entities targeted ⇐ this lecture
 - Recognizing and disambiguating the mentioned entities ⇐ this lecture

Understanding information needs

- Identifying target entity types
- Entity linking in queries

Identifying target entity types

One way to understand the meaning behind a search query is to identify the entity types targeted by the query.

The screenshot shows the Amazon search interface with the query "gps mount" entered into the search bar. The results page displays 1-16 of 168,801 results. A sidebar on the left lists product categories under "Show results for". The main content area features two sponsored product listings:

- WizGear Universal Magnetic Car Mounts**: Sponsored by WizGear. The listing includes an image of a black magnetic car mount, a brief description, and a "Shop now" link.
- Macally Adjustable Automobile Cup Holder Phone Mount**: Sponsored by Macally. The listing includes an image of a black cup holder mount, compatibility information for various phones (iPhone 6s, Galaxy S5, Note 4, Fitbit), a price of \$13.99, Prime delivery, and a 4.5-star rating.

Figure: Product categories displayed on Amazon in response to the query “gps mount.”

Problem definition

Definition

Target entity type identification is the task of finding the target types of a given input query, from a type taxonomy, such that these types correspond to most specific types of entities that are relevant to the query. Target types cannot lie on the same branch in the taxonomy. If no matching entity type can be found in the taxonomy, then the query is assigned a special NIL-type.

Approach

- Cast as a ranking problem: Given a keyword query q , estimate the relevance of an entity type $y \in \mathcal{T}$, $\text{score}(y; q)$
 - \mathcal{T} is the set of possible types (i.e., *type taxonomy*)
 - Note: NIL-type detection is not considered
- Unsupervised and supervised approaches

Example type taxonomies

DBpedia Ontology

- Well-designed hierarchy
- 700+ types organized in 7 levels

- [Agent](#) (edit)
 - [Deity](#) (edit)
 - [Employer](#) (edit)
 - [Family](#) (edit)
 - [NobleFamily](#) (edit)
 - [FictionalCharacter](#) (edit)
 - [ComicsCharacter](#) (edit)
 - [AnimangaCharacter](#) (edit)
 - [لینمنگا_کردار](#) (edit)
 - [DisneyCharacter](#) (edit)
 - [MythologicalFigure](#) (edit)
 - [NarutoCharacter](#) (edit)
 - [SoapCharacter](#) (edit)
 - [مُضحكه_خیزکردار](#) (edit)
 - [Organisation](#) (edit)
 - [Broadcaster](#) (edit)
 - [BroadcastNetwork](#) (edit)
 - [RadioStation](#) (edit)

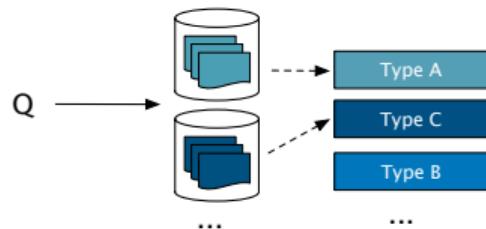
Wikipedia categories

- Not a well-defined “is-a” hierarchy, but a graph
 - A category may have multiple parent categories and there might be cycles along the path to ancestors
- Contains over 1.16M categories

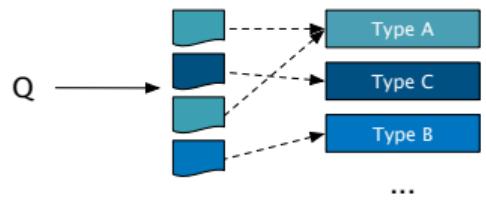
- ▼ [Formula One](#) (18 C, 57 P)
 - [British Formula One Championship](#) (1 C, 6 P)
 - [Formula One cars](#) (78 C, 37 P)
 - [Formula One circuits](#) (2 C, 80 P)
 - ▼ [Formula One constructors](#) (32 C, 132 P)
 - [Alex von Falkenhausen Motorenbau](#) (1 C, 2 P)
 - [Alfa Romeo in Formula One](#) (2 C, 7 P)
 - [All American Racers](#) (2 C, 1 P)
 - [Alpine F1 Team](#) (2 C, 2 P)
 - [Arrows Grand Prix International](#) (2 C, 10 P)
 - [Aston Martin in Formula One](#) (2 C, 6 P)
 - [Benetton Formula](#) (2 C, 28 P)
 - ▼ [BMW in Formula One](#) (2 C, 1 P)
 - [BMW \(1950s–60s\) Formula One drivers](#) (3 P)
 - [BMW Sauber Formula One cars](#) (4 P)

Unsupervised approaches

- **Type-centric model**
 - A term-based representation is built for each type
 - Types can then be ranked using existing document retrieval models
 - Analogous to constructing term-based entity representations from unstructured documents (cf. M6)



- **Entity-centric model**
 - Rank entities with respect to the query
 - Determine each type's relevance by considering the retrieval scores of entities of that type



Type-centric model

- A term-based representation (“type description” document) is constructed for each type
- Term pseudo-counts for a type are computed using:

$$\tilde{c}(t; y) = \sum_{e \in \mathcal{E}} c(t; e) w(e, y)$$

- $c(t; e)$ is the count of term t in the description of entity e
- $w(e, y)$ denotes the entity-type association weight; can be set uniformly across all entities that are typed with y :

$$w(e, y) = \begin{cases} \frac{1}{|\{e': y \in \mathcal{T}_{e'}\}|}, & y \in \mathcal{T}_e \\ 0, & y \notin \mathcal{T}_e \end{cases}$$

- Given the term-based type representation, as defined by $\tilde{c}(t; y)$, types can be ranked using any standard retrieval method

Entity-centric model

- First, entities are ranked based on their relevance to the query
 - $score(q; e)$ may be computed using any entity retrieval model discussed earlier
 - $\mathcal{E}_q(k)$ denotes the set of top- k ranked entities for query q
- Then, the score for a given type y is computed by aggregating the relevance scores of the top- k entities with that type:

$$score_{EC}(y; q) = \sum_{e \in \mathcal{E}_q(k)} score(e; q) w(e, y)$$

- $w(e, y)$ denotes the entity-type association weight (same as in type-centric model)

Supervised approach

- The type-centric model tends to return more specific types, while the entity-centric models favors more general types
- ⇒ Can the two be combined?
- Also, additional ranking signals may be incorporated in a supervised feature-based approach
- Types of features
 - Baseline (entity- and type-centric scores)
 - Type taxonomy (depth, #children, #siblings, etc.)
 - Type label (various ways of measuring the similarity between type label and query)

Evaluation

- Evaluated using standard rank-based measures
- Ground truth type annotations: $\hat{\mathcal{T}}_q$
- *Strict* evaluation: binary decision, i.e., each returned type y either matches one of the ground truth types (1) or not (0)
- However, not all types of mistakes are equally bad
 - For example, the target entity type is *racing driver*
 - Returning a more specific type (*rally driver*) or a more general type (*athlete*) is less of a problem than returning a type from a completely different branch of the taxonomy, like *organization* or *location*
- ⇒ Need for a more *lenient* evaluation that rewards near-misses

Lenient evalution

- First, the distance between types in the taxonomy $d(y, \hat{y})$ is set to
 - the number of steps between the two types, if they lie on the same branch (i.e., one of the types is a subtype of the other)
 - ∞ otherwise
- Then, the relevance level or gain of a type can be defined
 - in a *linear* fashion:

$$r(y) = \max_{\hat{y} \in \hat{T}_q} \left(1 - \frac{d(y, \hat{y})}{h} \right)$$

- or using an *exponential* decay function:

$$r(y) = \max_{\hat{y} \in \hat{T}_q} \left(b^{-d(y, \hat{y})} \right)$$

- where h is the depth of the type taxonomy and b is the base of the exponential function
- max is used in order to consider the closest match from the set of ground types \hat{T}_q
- The final measure is normalized discounted cumulative gain (NDCG), using the above (linear or exponential) relevance gain values

Exercise

E8-1 Target Entity Type Identification Evaluation

Understanding information needs

- Identifying target entity types
- Entity linking in queries

Entities in queries

- A large fraction of queries in Web search are reported to contain named entities (40-70%, depending on the study)
- Recognize entities in queries would lead to an improved search experience, e.g.,
 - Direct answers/knowledge panels
 - Better document ranking (next lecture)

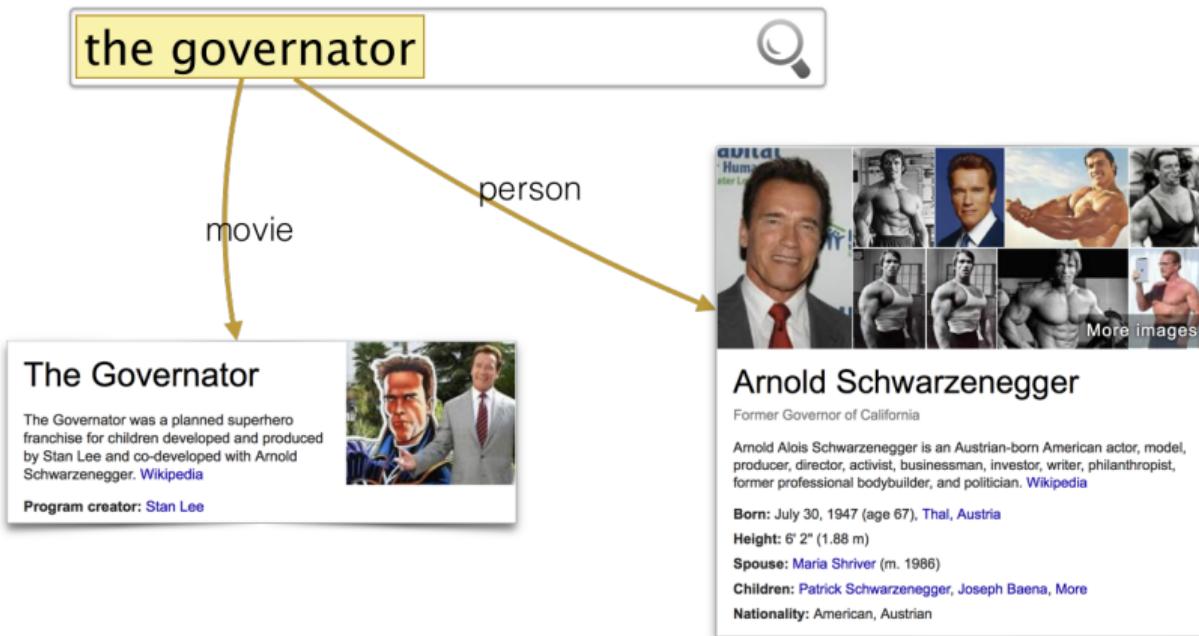
The screenshot shows a Google search results page for the query "Lisbon". The search bar at the top contains "Lisbon". Below the search bar, there are tabs for "Web", "Images", "Maps", "News", "Videos", "More", and "Search tools". The "Web" tab is selected. The results section starts with a snippet for the official website: "Lisbon - Official Website - visitlisboa.com". It includes a link, a snippet about the city being the capital and largest of Portugal, and a mention of Belém Tower. Below this is a section titled "Images for lisbon" with four thumbnail images of Lisbon. To the right of the images is a "Report images" link. Further down is a snippet from Lonely Planet: "Lisbon, Portugal - Lonely Planet". It includes a link, a snippet about the city's attractions like the Rio Tejo, and a "Top things to do in Lisbon - Best places to stay in Lisbon - Portugal image gallery" link. At the bottom of the snippet is a "TripAdvisor" link: "Lisbon Tourism: Best of Lisbon, Portugal - TripAdvisor". On the right side of the results, there is a sidebar with a map of Lisbon showing its districts like Alfama, Baixa, and Chiado. Below the map is a section titled "Lisbon" with the subtitle "Capital of Portugal". It provides basic stats: Area: 32.74 km², Weather: 61°F (16°C), Wind: NE at 7 mph (11 km/h), 82% Humidity, Local time: Tuesday, 10:41 AM, Hotels: 5-star averaging \$54, 5-star average \$140, and a "View hotels" link. It also lists Population: 530,847 (2012) and a "View population" link. The sidebar ends with a "Points of interest" section featuring small images of landmarks: Belém Tower, Jerónimos Monastery, São Jorge Castle, Lisbon Oceanarium, and Praça do Comércio.

Challenges

- Search queries are short
- Limited context
- Lack of proper grammar, spelling
- Multiple interpretations
- Needs to be fast

Example

Inherent ambiguity due to lack of context



Example

Ambiguity resolved

the governator movie

The Governor

The Governor was a planned superhero franchise for children developed and produced by Stan Lee and co-developed with Arnold Schwarzenegger. [Wikipedia](#)

Program creator: [Stan Lee](#)

Example

Multiple possible interpretations



Entity annotation tasks

- We distinguish between three entity annotation tasks in queries
 - *Named entity recognition* is the task of identifying mentions of named entities and tagging the mentions with their respective types
 - *Semantic linking* seeks to find a ranked list of entities that are semantically related to the query string
 - Unlike in entity retrieval, the goal is not to answer the user's underlying information need with a ranked list of entities, but to identify entities that are referenced (either explicitly or implicitly) in the query
 - *Interpretation finding* aims to discover all plausible meanings of the query; each interpretation consists of a set of non-overlapping and semantically compatible entity mentions, linked to a knowledge repository

Comparison of tasks

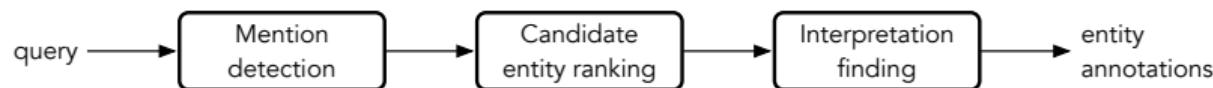
	Named Entity Recognition	Semantic Linking	Interpretation Finding
Result format	set/ranked list	ranked list	sets of sets
Explicit entity mentions?	yes	no	yes
Mentions can overlap	no	yes	no ²
Evaluation criteria	recognized entities ¹	relevant entities	interpretations
Evaluation measures	set/rank-based	rank-based	set-based
Examples			
"obama mother"	"obama"/PER	BARACK OBAMA ANN DUNHAM	{ { BARACK OBAMA } }
"new york pizza manhattan"	"new york"/LOC "manhattan"/LOC	NEW YORK CITY NEW YORK-STYLE PIZZA MANHATTAN MANHATTAN PIZZA	{ { NEW YORK CITY, MANHATTAN }, { NEW YORK-STYLE PIZZA, MANHATTAN } }
		...	

¹ Along with their respective types.

² Not within the same interpretation.

Interpretation finding

- Addresses the ambiguity of queries head-on: a query can legitimately have more than one interpretation
 - An interpretation is a set of non-overlapping linked entity mentions that are semantically compatible with the query text
- Pipeline architecture



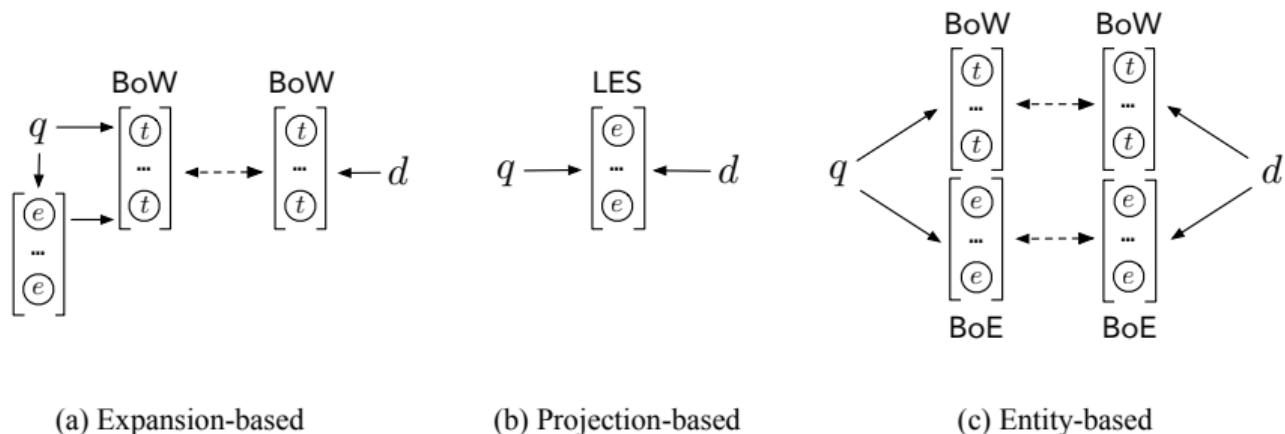
- *Mention detection* is performed similarly as it is done for documents (i.e., dictionary-based)
- *Candidate entity ranking* corresponds to the task of semantic linking
- *Interpretation finding* is the counterpart of the disambiguation component in conventional entity linking

Leveraging entities in document retrieval

Leveraging entities in document retrieval

- Traditionally, document retrieval methods compare query and document representations, which are based on terms (words)
- Entities facilitate a semantic understanding of both queries and documents
- Different approaches for leveraging entities
 - Expansion-based
 - Projection-based
 - Entity-based
- Prerequisite: Mapping queries to entities

Three main ways to leverage entities



- *Expansion-based*: utilize entities as a source of expansion terms to enrich the query
- *Projection-based*: treat entities as a latent layer and map both queries and documents to a latent entity space (LES)
- *Entity-based*: consider entity-based representations (bag-of-entities) in “duet” with traditional bag-of-words representations in the retrieval model

Leveraging entities in document retrieval

- Mapping queries to entities
- Projection-based methods
- Entity-based representations

Mapping queries to entities

- Goal: Identify a set of entities that may be semantically related to the query
- Shared by all approaches for leveraging entities
- Use the probability $P(e|q)$ to express the likelihood of entity e being related to query q
- The estimation may be based on:
 - entities mentioned in the query
 - entities retrieved directly from a knowledge base
 - entities retrieved indirectly, through pseudo-relevant documents

Entities mentioned in the query

- Entities can be identified and disambiguated using entity linking techniques
- Single entity with the highest annotation score

$$P(e|q) = \begin{cases} 1, & e = \operatorname{argmax}_{e \in \mathcal{E}_q} \operatorname{score}_{ELQ}(e; q) \\ 0, & \text{otherwise} \end{cases}$$

- \mathcal{E}_q is set of entities that have been identified in query q
- $\operatorname{score}_{ELQ}(e; q)$ is the associated confidence score
- Multiple entities

$$P(e|q) = \begin{cases} \frac{1}{Z} \operatorname{score}_{ELQ}(e; q), & \operatorname{score}_{ELQ}(e; q) > \gamma \\ 0, & \text{otherwise} \end{cases}$$

- γ is a threshold parameter
- Z is a normalization coefficient such that $0 \leq P(e|q) \leq 1$

Entities retrieved directly from a KB

- Query a knowledge base directly for relevant entities

$$P(e|q) = \begin{cases} \frac{1}{Z} score_{ER}(e; q), & e \in \mathcal{E}_q(k) \\ 0, & \text{otherwise} \end{cases}$$

- $score_{ER}(e; q)$ is the relevance score of entity e given q
- Z is a normalization coefficient
- Note that for pragmatical reasons only the top-k entities are considered

Entities retrieved indirectly

-
- Uses the top-ranked documents retrieved in response to the query
 - This is in the spirit of pseudo relevance feedback
 - Corresponds to the setting of ranking entities without direct representations

$$P(e|q) \propto \sum_{d \in \mathcal{D}_q(k)} P(e|d)P(d|q)$$

- $\mathcal{D}_q(k)$ denotes the set of top-k highest scoring documents retrieved in response to query q
- $P(d|q)$ corresponds to document d 's relevance to the query
- $P(e|d)$ is the probability of observing the entity in d

Entities retrieved indirectly

- $P(e|d)$ may be taken as a maximum-likelihood estimate

$$P(e|d) = \frac{c(e; d)}{\sum_{e' \in d} c(e'; d)}$$

- $c(e; d)$ is the number of times entity e occurs in document d
- The frequency of e across the document collection may also be taken into account by adding an IDF-like component

Leveraging entities in document retrieval

- Mapping queries to entities
- Projection-based methods
- Entity-based representations

Projection-based methods

- Addresses the inherent limitation of standard IR retrieval models to retrieve (relevant) documents that have no explicit term matches with the query
- The overall idea:
 - Construct a high-dimensional latent entity space where each dimension corresponds to one entity and then map both queries and documents to the latent space accordingly
- The relevance between a query and a document is estimated based on their projections to this latent entity space
- Allows to uncover hidden (latent) semantic relationships between queries and documents

Explicit Semantic Analysis

- The semantics of a given word are described by a vector storing the word's association strengths to a KR concept
- Primarily focused on using Wikipedia as the underlying knowledge repository
- Two components:
 - Concept-based indexing
 - Concept-based retrieval

Concept-based indexing

- The semantic representation of a given term t is a concept vector of length $|\mathcal{E}|$

$$t = \langle w(e_1, t), \dots, w(e_{|\mathcal{E}|}, t) \rangle$$

- Each element of the vector corresponds to an entity in the knowledge repository
- $w(e, t)$ quantifies the strength of the association between term t and the given entity. It is the normalized TF-IDF weight of t in the description of e

$$w(e, t) = \frac{\text{TFIDF}(t, e)}{\sqrt{\sum_{t' \in \mathcal{V}} \text{TFIDF}(t', e)^2}}$$

Concept-based indexing

- The semantic representation is computed by taking the centroid of the individual terms' concept vectors

$$w(e_j, z) = \frac{1}{l_z} \sum_{t \in \mathcal{Z}} c(t; z) w(e_j, t)$$

- l_z is the length of z
- $c(t; z)$ is the number of times term t appears in z
- Concept-based vectors are sparse, with most weights being zero
- they can be efficiently represented using an inverted index.

Concept-based retrieval

- semantic similarity between query q and document d may be computed by
 - mapping both query q and document d to the ESA concept space
 - taking cosine similarity of their concept vectors

$$score(q; d) = \cos(\mathbf{q}, \mathbf{d})$$

- issue with long documents where only a small part might be relevant to the current query
 - The solution is to split the document into passages where each passage $s \in d$ is represented by its own concept vector \mathbf{s} is matched against the query
 - The final retrieval score combines the full document's similarity score with that of the best performing passage

$$score_{ESA}(q; d) = \cos(\mathbf{q}, \mathbf{d}) + \max_{s \in d} \cos(\mathbf{q}, \mathbf{s})$$

Latent Entity Space model

- Based on a generative probabilistic framework.
- The document's retrieval score is a linear combination of the latent entity space score and the original query likelihood score

$$score_{LES}(q; d) = \alpha \underbrace{\sum_{e \in \mathcal{E}} P(q|e)P(e|d)}_{\text{LES score}} + (1 - \alpha)P(q|d)$$

- α is the interpolation parameter
- $P(q|d)$ is the query likelihood score
- $P(q|e)$ and $P(e|d)$ are query projection and document projection respectively

Query projection

- May be interpreted as the likelihood of the query q being generated by entity e
- One approach is to base it on the language model of the entity, θ_e

$$P(q|e) = \prod_{t \in q} P(t|\theta_e)^{c(t;q)}$$

- Another approach is to leverage the set of query entities \mathcal{E}_q in a pairwise manner

$$P(q|e) \propto \sum_{e' \in \mathcal{E}_q} sim(e, e') P(e'|q)$$

- $sim(e, e')$ may be any symmetric pairwise similarity measure
- $P(e'|q)$ is the query association probability for e'

Document projection

- The probability $P(e|d)$ may be interpreted as the projection of document d to the latent entity space
- May be estimated using existing document retrieval models, e.g., by computing entity likelihood (the probability of e generated from the language model of d)
- Example using cross-entropy between the document and entity language models:

$$P(e|d) = \exp(-CE(\theta_e||\theta_d)) = \exp\left(\sum_{t \in \mathcal{V}} P(t|\theta_e) \log(P(t|\theta_d))\right)$$

- Learning-to-Rank Model (Latent-ListMLE)
- Uses a combination of
 - Query-entity features (query projection)
 - Entity-document features (document projection)
 - Other features (e.g., entity popularity, document quality)

Features in EsdRank

Group	Description
<i>Query-entity features</i>	
$P(e q)$	Query-entity association probability (cf. Sect. 8.1)
$score_{ER}(e; q)$	Entity retrieval score (relevance of e given q)
$sim(\mathcal{T}_e, \mathcal{T}_q)$	Type-based similarity between the entity (\mathcal{T}_e) and the query (\mathcal{T}_q)
$maxSim(e, \mathcal{E}_q)$	Max. pairwise similarity between e and other query entities $e' \in \mathcal{E}_q$
$avgSim(e, \mathcal{E}_q)$	Avg. pairwise similarity between e and other query entities $e' \in \mathcal{E}_q$
<i>Entity-document features</i>	
$sim(e, d)$	Textual similarity between e and d
$sim(\mathcal{T}_e, \mathcal{T}_d)$	Type-based similarity between the entity (\mathcal{T}_e) and the document (\mathcal{T}_d)
$numMentioned(\mathcal{E}_e^i, d)$	Number of related entities (at most i hops from e) mentioned in d
<i>Other features</i>	
$IDF(e)$	IDF score of the entity (based on the number of documents that are annotated with e)
$quality(d)$	Quality score of d (document length, SPAM score, PageRank, etc.)

Note that many of these features are instantiated multiple times, using different similarity methods or parameter configurations

- ListMLE
 - it is a listwise learning-to-rank algorithm that uses a parametric model to estimate the probability of a document ranking being generated by a query
 - It employs maximum likelihood estimation (MLE) to find the parameters that maximize the likelihood of the best ranking
- Latent-ListMLE extends ListMLE by adding a latent layer of candidate entities in the generation process
- The probability of a document ranking $d = \langle d_1, \dots, d_k \rangle$ given q is

$$P(d|q; w, \theta) = \prod_{i=1}^k \sum_{e \in \mathcal{E}_{II}} P(d_i|e, \mathcal{D}_q(i, k)) P(e|q)$$

- $\mathcal{D}_q(i, k) = \{d_i, \dots, d_k\}$ is the set of documents that were not ranked in positions $1, \dots, i-1$
- The parameters of the model, w and θ , are learned using MLE and the EM algorithm

Leveraging entities in document retrieval

- Mapping queries to entities
- Projection-based methods
- Entity-based representations

Entity-based representations

- We make use of explicit entity annotations of documents
 - In contrast to projecting documents to a latent entity layer
- Several approaches
 - Entities may be blended with terms in a single representation layer or
 - a separate bag-of-entities representation may be introduced and combined with the traditional bag-of-terms representation

Entity-Based Document Language Models

- Consider individual terms as well as term sequences that have been annotated as entities
- Extend the vocabulary of terms (\mathcal{V}) with entities (\mathcal{E})
- The maximum likelihood estimate of vocabulary token x (may be a term or an entity, $x \in \mathcal{V} \cup \mathcal{E}$) given d is defined as

$$P(x|d) = \frac{c(x; d)}{l_d}$$

- $c(x; d)$ is the (pseudo) count of x in document d
- The representation length of the document is then given by $l_d = \sum_{x \in d} c(x; d)$

Entity-Based Document Language Models (cont'd)

- This maximum likelihood estimate is then smoothed with a background (collection-level) language model

$$P(x|\theta_d) = \frac{c(x; d) + \mu P(x|D)}{l_d + \mu}$$

- μ is a smoothing parameter
- The collection language model is also a maximum likelihood estimate, computed over the set \mathcal{D} of documents

$$P(x|D) = \frac{\sum_{d \in \mathcal{D}} c(x; d)}{\sum_{d \in \mathcal{D}} l_d}$$

Entity-Based Document Language Models (cont'd)

- How to compute the token pseudo-counts?

- **Hard confidence-level thresholding**

- Only consider those entity annotations that are above a given (pre-defined) threshold $\tau \in [0, 1]$

$$\tilde{c}(x; d) = \begin{cases} \lambda c(x; d), & x \in \mathcal{V} \\ (1 - \lambda) \sum_{i=1}^{l_d} \mathbb{1}(x_i = x, score_{EL}(x_i; d) \geq \tau), & x \in \mathcal{E} \end{cases}$$

- x_i refers to the token at position i in the document
 - $score_{EL}(x_i; d)$ is the entity linking confidence associated with that token
 - The λ parameter controls the relative importance given to term vs. entity tokens.

- **Soft confidence-level thresholding**

- Recognizes all entities that are linked in the document and weighs them by their corresponding confidence levels

$$\tilde{c}(x; d) = \begin{cases} \lambda c(x; d), & x \in \mathcal{V} \\ (1 - \lambda) \sum_{i=1}^{l_d} \mathbb{1}(x_i = x) score_{EL}(x_i; d), & x \in \mathcal{E} \end{cases}$$

Entity-Based Document Language Models (cont'd)

- The ranking of documents is based on the negative cross-entropy (CE) between the query and document language models

$$score_{ELM}(d; q) = -CE(\theta_q || \theta_d) = \sum_{x \in \mathcal{V} \cup \mathcal{E}} P(x|\theta_q) \log P(x|\theta_d)$$

- θ_q and θ_d are query and document language models respectively

Bag-of-Entities Representation

- So far, entity-based language models used a single representation layer, in which terms and entities are mixed together
- Now, the term-based and entity-based representations are kept apart and are used in "duet"
 - Queries and documents are represented in the term space as well as in the entity space (*bag-of-entities*)
- Several ranking models
 - Basic Ranking Models
 - Explicit Semantic Ranking (ESR)
 - Word-Entity Duet Framework

Basic Ranking Models

- Two basic ranking models based on bag-of-entities representations
- **Coordinate Match**

$$score_{CM}(d; q) = \sum_{e \in \mathcal{E}_q} \mathbb{1}(c(e; d) > 0)$$

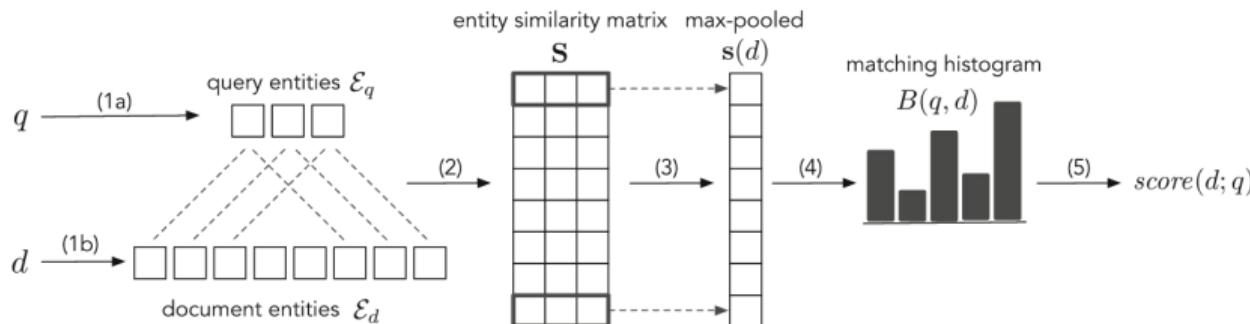
- **Entity Frequency**

$$score_{EF}(d; q) = \sum_{e \in \mathcal{E}_q} c(e; q) \log c(e; d)$$

- These ranking functions are used to re-rank the top-k documents retrieved by a standard term-based retrieval model

Explicit Semantic Ranking (ESR)

- Incorporates relationship information from a knowledge graph to enable "soft matching" in the entity space



- First, creates a query-document entity similarity matrix S .
 - Each element $S(e, e')$ in this matrix represents the similarity between a query entity $e \in \mathcal{E}_q$ and a document entity $e' \in \mathcal{E}_d$

$$S(e, e') = \cos(\mathbf{e}, \mathbf{e}')$$

- \mathbf{e} is the embedding vector of entity e

Explicit Semantic Ranking (cont'd)

- ESR performs two pooling steps
- First max-pooling along the query dimension

$$s(d) = \max_{e \in \mathcal{E}_q} S(e, \mathcal{E}_d)$$

- The second is bin-pooling to group and count the number of document entities according to the strength of their matches to the query

$$B_i(q, d) = \log \sum_j \mathbb{1}(st_i \leq s_j(d) < ed_i)$$

- $[st_i, ed_i)$ is the score range for the i th bin
- $B_i(q, d)$ is the number of entities that fall into that bin

Word-Entity Duet Framework

- Incorporates cross-space interactions between term-based and entity-based representations
 - Query terms to document terms ($match(\mathbf{q}_t, \mathbf{d}_t)$)
 - Query entities to document terms ($match(\mathbf{q}_e, \mathbf{d}_t)$)
 - Query terms to document entities ($match(\mathbf{q}_t, \mathbf{d}_e)$)
 - Query entities to document entities ($match(\mathbf{q}_e, \mathbf{d}_e)$)

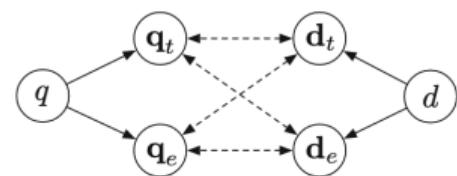


Figure: \mathbf{q}_t and \mathbf{d}_t denote the bag-of-words, \mathbf{q}_e and \mathbf{d}_e denote the bag-of-entities representations

Summary

- Understanding information needs
 - Identifying target entity types
 - Entity linking in queries
- Leveraging entities in document retrieval
 - Mapping queries to entities
 - Projection-based methods
 - Entity-based representations

Reading

- Entity-Oriented Search (Balog)
 - Chapter 7, Chapter 8
 - Section 7.2, until 7.2.4.1 (inclusive)
 - Section 7.3, until 7.3.2 (inclusive)
 - Introduction and Section 8.1
 - Sections 8.3–8.4.2.3 (inclusive)

Neural Information Retrieval

[DAT640] Information Retrieval and Text Mining

Weronika Lajewska

University of Stavanger

03.10.2022



CC BY 4.0

In this module

1. Text representations for ranking
2. Dense retrieval

Text representations for ranking

Text representations for ranking

- Recap on ranking and text representations
- Transition from sparse to dense embeddings
- Static word embeddings
- Contextualized word embeddings

Recap on ranking

- **Ranking** is a core of the search engine. It calculates scores for documents using a *ranking algorithm* based on a *retrieval model* and generates a ranked list of documents for the user's query.
- **Retrieval function** estimates the relevance of documents in the collections w.r.t. the input query q (so that the highest-scoring ones can be returned as retrieval results)
- In principle, the relevance score is computed for each document in the collection. In practice, we're only interested in the top-k results for each query.

Recap on text representation: Discrete (sparse) representation

- Sparse representation has the length of the vocabulary and each element of the vector corresponds to a term in the vocabulary
- The i -th component of the vector encodes a discrete value, e.g. the presence/absence of a word (0/1) in the represented text (query or document)
- Sparse representations are based on the bag of words (BOW) model and the position of the word in the text is ignored in the representation
- Sparse representations can be efficiently stored in an inverted index

ID	term
1	discrete
2	sparse
3	document
4	text
5	word
6	this
	...

Vocabulary

$$d = \text{"this is sparse text"}$$

$$d = [t_6, OOV, t_2, t_4]$$

$$\vec{d} = \langle 0, 1, 0, 1, 0, 1, \dots \rangle$$

Recap on text representation: Learning-to-Rank features

- Richer query and document representation can be built by exploiting additional relevance signals, such as PageRank, SPAM score, clicks, etc.
- Learning-to-Rank is implemented as a two-step retrieval with initial ranking (retrieving top-N candidate documents) and re-ranking (creating feature vectors and re-ranking top-N candidates)
- We distinguish three classes of features:
 - query features, e.g. query length, query type, number of named entities in the query
 - document features, e.g. URL length, number of inlinks, PageRank score
 - query-document features, e.g. retrieval score of a given document field

Recap on text representation: Learning-to-Rank features - cont.

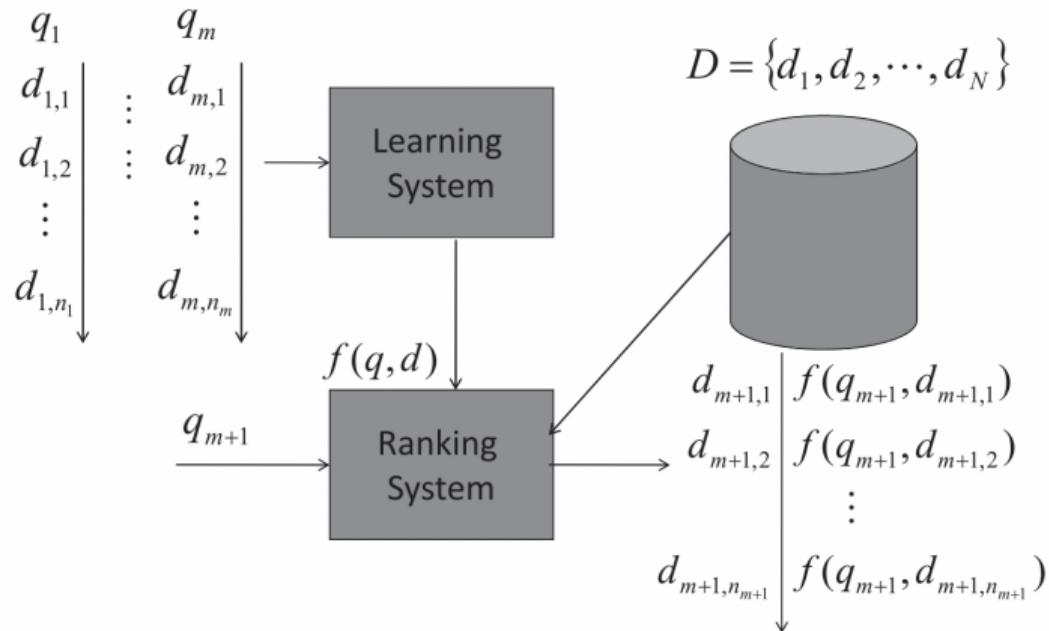


Figure: Learning to rank architecture for document retrieval¹

¹https://search.ieice.org/bin/pdf_link.php?category=D&lang=E&year=2011&fname=e94-d_10_1854&abst=

Text representations for ranking

- Recap on ranking and text representations
- Transition from sparse to dense embeddings
- Static word embeddings
- Contextualized word embeddings

Word-context matrix²

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

Word-context matrix is a co-occurrence matrix with W rows (words) and C columns (contexts), where f_{ij} gives the number of times word w_i occurs in context c_j . We distinguish different types of similarity:

- first-order co-occurrence - words appear nearby each other, e.g., *write* is a first-order associate of *book* or *poem*
- second-order co-occurrence - words have similar neighbors, e.g., *write* is a second-order associate of words like *say* or *remark*

²<https://web.stanford.edu/~jurafsky/slp3/>

Pointwise Mutual Information

- *Pointwise mutual information (PMI)* is a measure of how often two events x and y occur, compared with what we would expect if they were independent:

$$I(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$$

- Then the PMI between a target word w and a context word c (first-order co-occurrence) is defined as:

$$PMI(w, c) = \log_2 \frac{P(w, c)}{P(w)P(c)}$$

- The denominator tells us how often we would expect the two words to co-occur assuming they each occurred independently

PPMI matrix - sparse representation

PMI values range from negative to positive infinity. But negative PMI values (which imply things are co-occurring less often than we would expect by chance) tend to be unreliable unless our corpora are enormous. Thus, it is more common to use Positive PMI (PPMI) defined as $PPMI(w, c) = \max(PMI(w, c), 0)$.

	computer	data	result	pie	sugar
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

Figure: The PPMI matrix showing the association between words and context words³

³<https://web.stanford.edu/~jurafsky/slp3/>

Singular Value Decomposition (SVD)

- An $N \times M$ matrix X can be factorized into three matrices:

$$X = U\Sigma V^T$$

- where:
 - U (left singular vectors) is an $N \times N$ unitary matrix
 - Σ is an $M \times M$ diagonal matrix, where diagonal entries are eigenvalues (all positive, sorted from large to small values)
 - V^T (right singular vectors) is an $M \times M$ unitary matrix

PPMI + SVD

- By applying SVD to PPMI, keeping only top k eigenvalues in Σ , and setting the rest to 0 we can truncate the U and V^T , resulting in U_k and V_k^T
- U_k and V_k^T are new low-dimensional word vectors
- Both, U_k and V_k^T , can be treated as dense word embeddings

Text representations for ranking

- Recap on ranking and text representations
- Transition from sparse to dense embeddings
- Static word embeddings
- Contextualized word embeddings

Distributional semantics

You shall know a word by the company it keeps.

(J. R. Firth, A synopsis of linguistic theory (1957))

Distributional semantics

- Words that occur in the same context tend to have a similar meaning, e.g. book, volume, novel
- The meaning of words can be inferred by their usage together with other words in existing texts
- The *context* of a word is the set of words that appear within a fixed-size window. They are called *context-words*.

Distributional representation

- A word (token) can be represented as a vector of d dimensions
- d is much smaller than the size of the vocabulary
- This d -dimensional vector is called *distributional representation* or *word embedding*

Characteristics of word embeddings

- Components of word embeddings are rarely 0: they are real numbers, and can also have negative values
- Components of distributional representation are not mutually exclusive. More than one component can be "active" at the same time.

$$\vec{w} = \begin{bmatrix} 0.15 & 0.07 & 0.83 & 0.46 & \dots & 0.02 \end{bmatrix}$$

- Word embeddings are often referred to as *dense representations*

Word embeddings - visualization

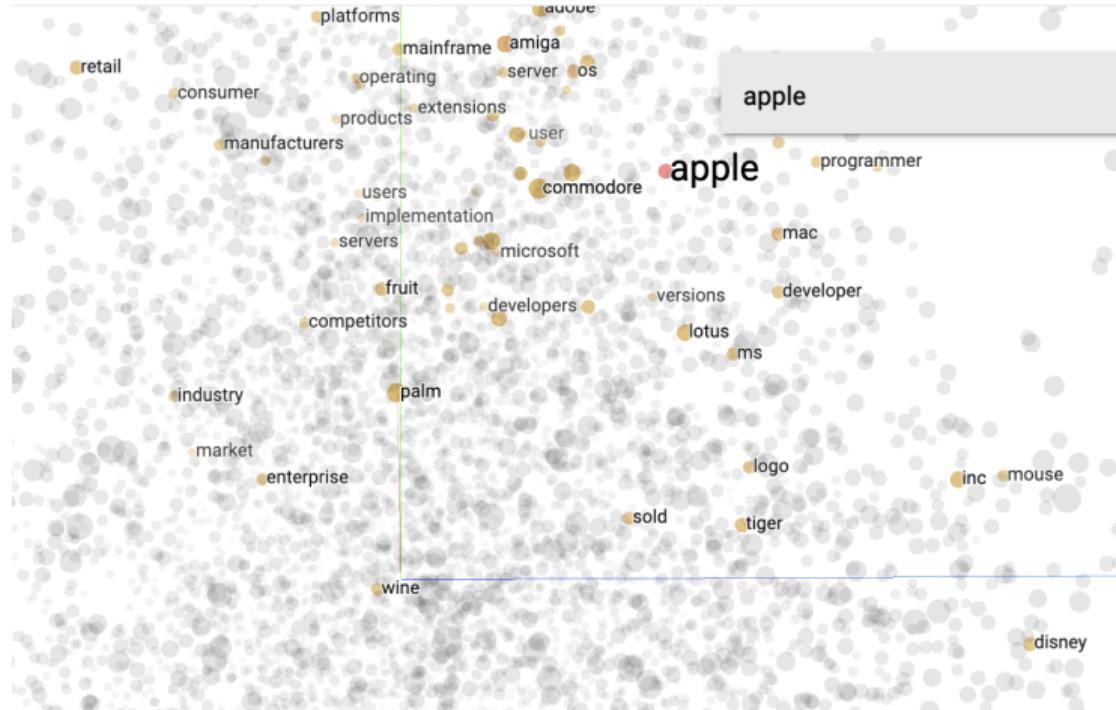
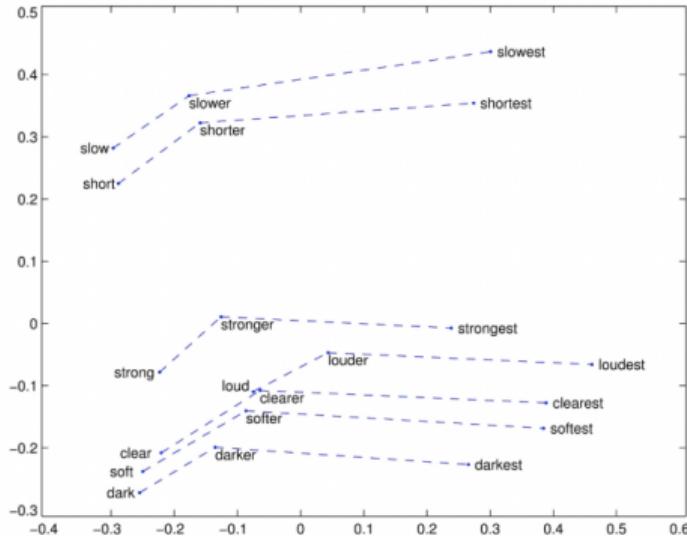
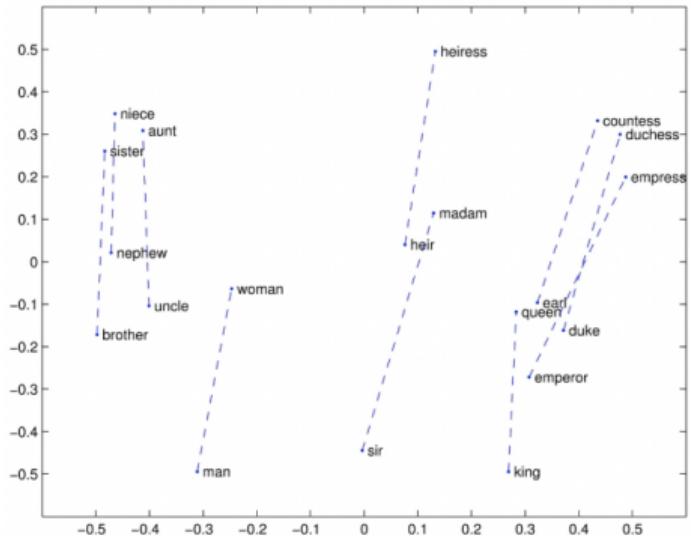


Figure: Tensorflow Embedding Projector ⁴

⁴<https://projector.tensorflow.org/>

Analogy problem⁵



Analogy/relational similarity: Embeddings are able to capture relational meanings.
Analogy problem can be generalized to the form X is to Y as X^* is to what?

⁵<https://web.stanford.edu/~jurafsky/slp3/>

Bias in word embeddings

- Embeddings reproduce the implicit biases and stereotypes that are latent in the text
- Bias visible in relational similarity:
 - *man:woman::king:queen*
 - *father:mother::doctor:nurse*
- Allocational harm - situation when system allocates resources unfairly to different groups
- Representational harm - situation when system demean or ignore some social groups
- Embeddings don't only reflect the statistics of their input, but they also amplify bias → biased terms become more biased in embedding space than they were in the input text statistics
- Debiasing is still an open problem

Recap: n-gram Language Model

- Language modeling is a task of predicting a word given a context:

$$P(\text{word}|\text{context})$$

- Formal definition: Given a sequence of words x_1, x_2, \dots, x_t , a language model calculates the probability distribution of next word x_{t+1} over all words in vocabulary

$$P(x_{t+1}|x_t, x_{t-1}, \dots, x_1)$$

- A *n-gram* is a chunk of n consecutive words
- A *n-gram language model* collects frequency statistics of different n -grams in a corpus, and uses these to calculate probabilities
- *Neural n-gram language model* predicts the co-occurrence probabilities
- Language models can be used to generate text by sampling from the probability distributions

word2vec: static word embeddings

- Static word embeddings are global representations of the words, i.e., there is a single fixed word embedding for each term in the vocabulary. Static word embeddings map terms with multiple senses into an average or most common sense representation based on the training data used to compute the vectors.
- Word embeddings are a by-product of neural language models
- We can use a neural language model to create a word embedding, by using the model to predict the probability of appearance of a context-word in a window around the given word

word2vec: skip-gram

- Intuition:
 - Treat the target word and a neighboring context word as positive examples
 - Randomly sample other words in the lexicon to get negative samples
 - Use logistic regression to train a classifier to distinguish those two cases
 - Use the learned weights as the embeddings
- Skip-gram trains a logistic regression classifier to compute the probability that two words are ‘likely to occur nearby in text’. This probability is computed from the dot product between the embeddings for the two words.

word2vec: skip-gram

- Maximize the probability of true context words $w_{t-m}, w_{t-m+1}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+m-1}, w_{t+m}$ for each target word w_t :

$$J'(\theta) = \prod_{t=1}^T \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j}|w_t; \theta)$$

- Negative Log Likelihood:

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j}|w_t; \theta)$$

word2vec: negative sampling

The skip-gram model tries to shift embeddings so that the target embeddings are closer to (have a higher dot product with) context embeddings for nearby words and further from (lower dot product with) context embeddings for noise words that don't occur nearby

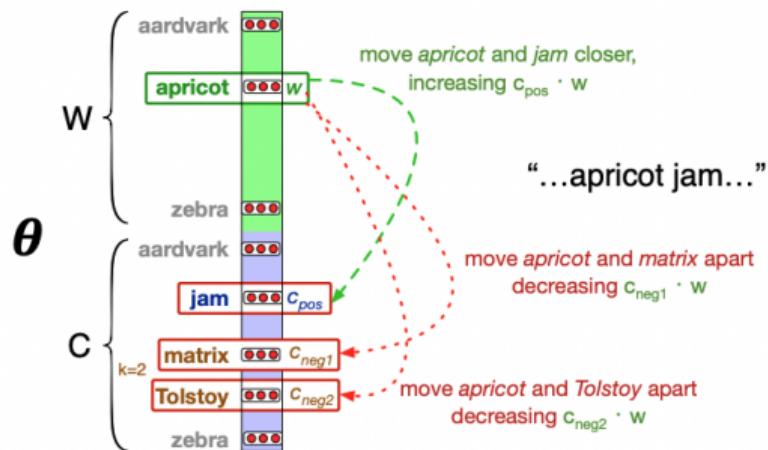


Figure: The skip-gram model ^a

^a<https://web.stanford.edu/~jurafsky/slp3/>

word2vec: architecture

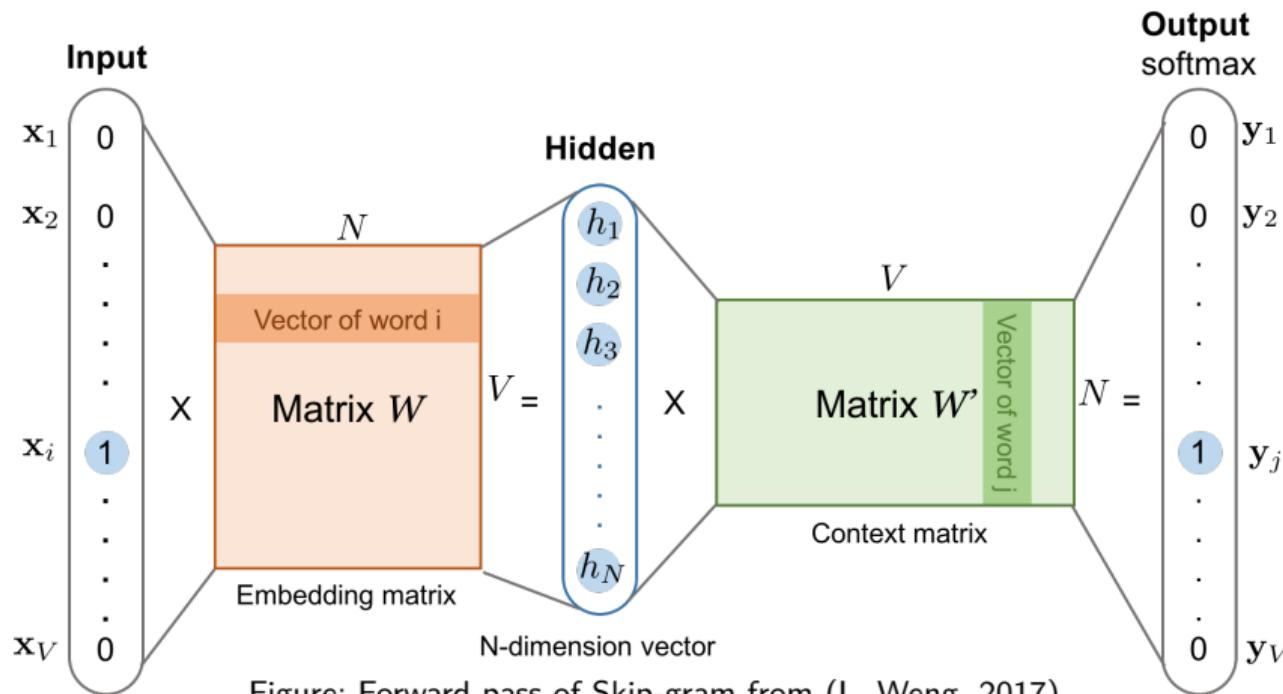


Figure: Forward pass of Skip-gram from (L. Weng, 2017).

Exercise

E9-1 Word embeddings with word2vec

Text representations for ranking

- Recap on ranking and text representations
- Transition from sparse to dense embeddings
- Static word embeddings
- Contextualized word embeddings

Contextualized word embeddings

- Contextualized word embeddings are local representations of the terms which benefit from the context. Each term in the vocabulary is associated with a different representation vector every time it appears in a document, depending on the surrounding tokens.
- The most popular contextualized word embeddings are learned with deep neural networks such as the Bidirectional Encoder Representations from Transformers (BERT), the Robustly Optimized BERT Approach (RoBERTa), and the Generative Pre-Trained model (GPT).

Dense retrieval

Dense retrieval

- Dense retrieval overview
- Negative sampling
- Embedding index
- Single representation systems
- Multiple representations systems

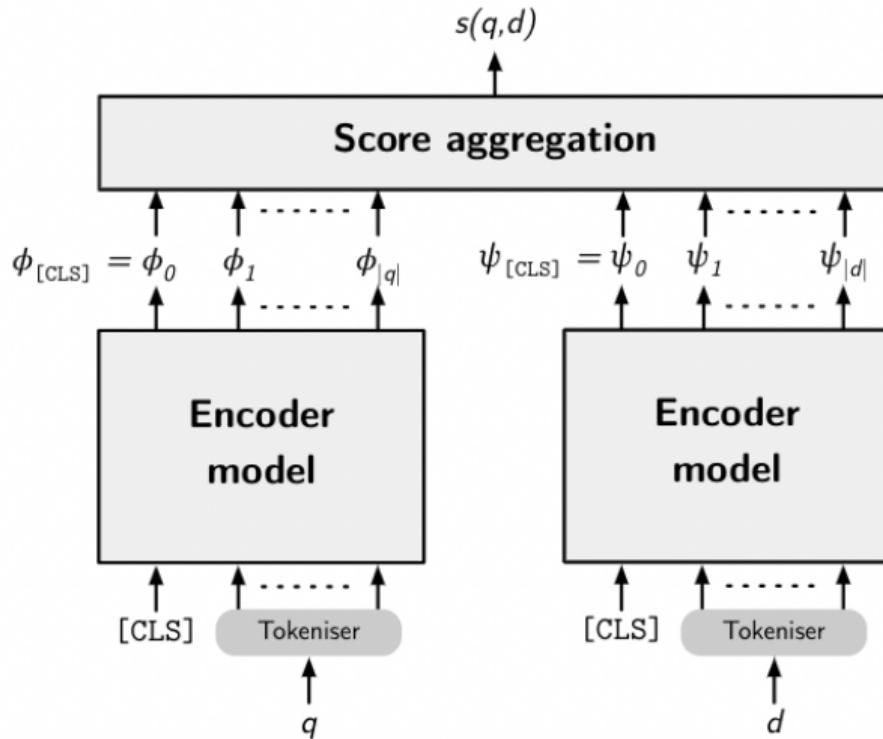
Recap on sparse retrieval models

- Advantages of BM25:
 - It is fast and effective
- Disadvantages of BM25:
 - Bag-of-words representation disregards word order
 - Context of words is not taken into consideration
 - It is an exact term-matching algorithm, meaning that it cannot leverage semantic similarity between words
 - Potential vocabulary mismatch

Dense Retrieval (Representation-focused Systems)

- Dense retrieval aims to match texts in a continuous representation space learned via deep neural networks
- Dense retrieval is characterized by fully learnable representation
- Dense retrieval calculates the relevance score using similarities in a learned embedding space
- Similarity function can be cosine or dot product
- The same representation model is used to encode a query and a document
- The effectiveness of dense retrieval resides in learning a good representation space that maps query and relevant documents together while separating irrelevant ones.

Representation-focused Systems⁶



⁶<https://arxiv.org/pdf/2207.13443.pdf>

Query and documents representations

- Query representation and document representations are learned independently
- Representations of all documents in the collection can be pre-computed and stored in advance
- During query processing, only the query representation is computed
- Representation-based systems are able to identify the relevant documents among all documents in the collection, not only the ones that are dependent on the query
- Dense retrieval takes into consideration the semantics of the terms in a query and is not limited to an exact match of terms

Dense Retrieval Ranking Architecture

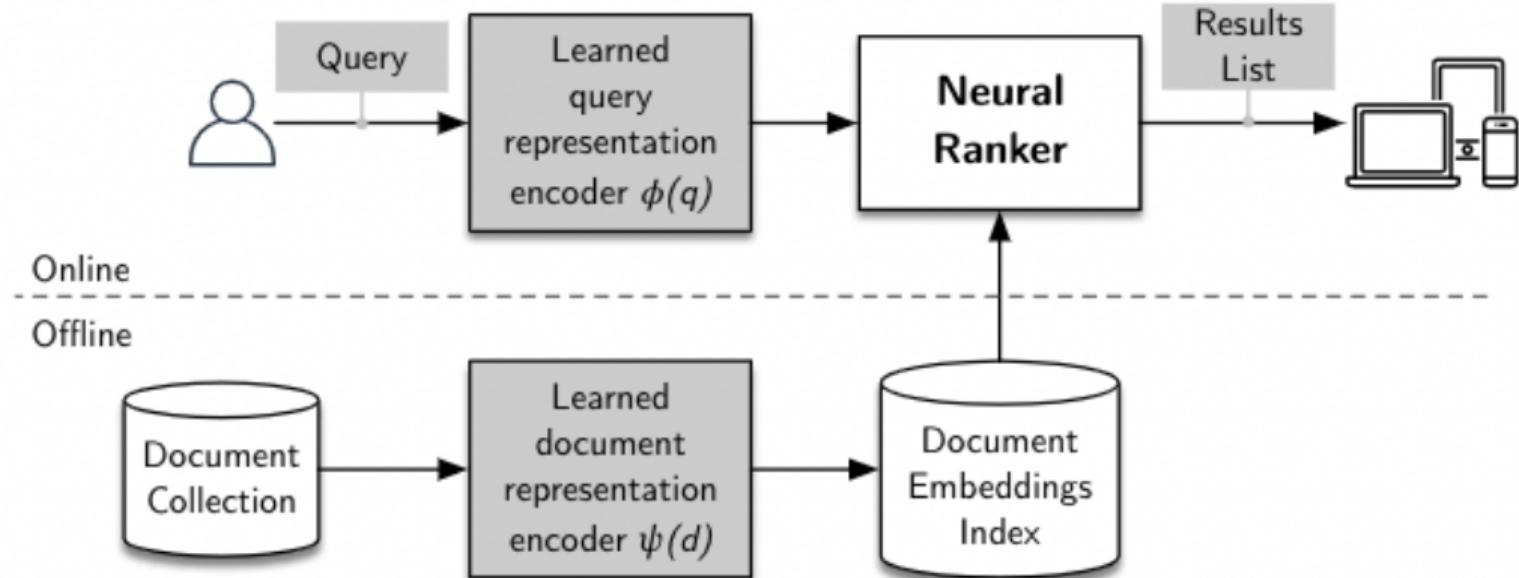


Figure: Dense retrieval architecture for representation-focused neural IR systems⁷

⁷<https://arxiv.org/pdf/2207.13443.pdf>

Main challenges

Question

How to train a model to distinguish between relevant documents and irrelevant documents?

Question

How to search through a massive index of document representations?

Main challenges

- How to train a model to distinguish between relevant documents and irrelevant documents? →**Negative sampling**
- How to search through a massive index of document representations?

Dense retrieval

- Dense retrieval overview
- **Negative sampling**
- Embedding index
- Single representation systems
- Multiple representations systems

Negative sampling

- Most likely we have several relevant documents and millions of irrelevant ones in the collection
- The model needs some negative examples to be able to differentiate between correct matches and incorrect ones
- Choosing meaningful irrelevant documents is a non-trivial task

Fine-tuning representation-focused systems with contrastive learning

- The fine-tuning of a bi-encoder corresponds to learning an appropriate inner-product function suitable for the ranking task, i.e., for relevance scoring
- The goal is to learn a latent vector space for query and document representations where a query and its relevant documents are closer than the query and its non-relevant documents

Negative sampling

- *Random negatives* - any document from the corpus is considered irrelevant with equal probability
- *In-batch local negatives* - during training, the queries are randomly aggregated into batches of size N . If passage P1 is a good match for query Q1, it is a negative example for the remaining queries in the batch Q2, Q3, ..., QN.
- The main problem with the random negatives and in-batch local negatives is that they are not really informative for the model. They are easily distinguishable which leads to a weaker signal sent to the model.
- We need negative examples that are hard to distinguish from the relevant documents

Negative sampling - hard negatives

- Negative documents can be generated exploiting a classical or trained retrieval system
- Each query is given as input to the retrieval system. The non-relevant documents included in the top-N retrieved documents are treated as negatives.
- We assume to know the relevant documents for the query
- The retrieval system used to mine the negative documents can be a standard BM25, the currently neural model under training or another fine-tuned neural model

Negative sampling - hard negatives

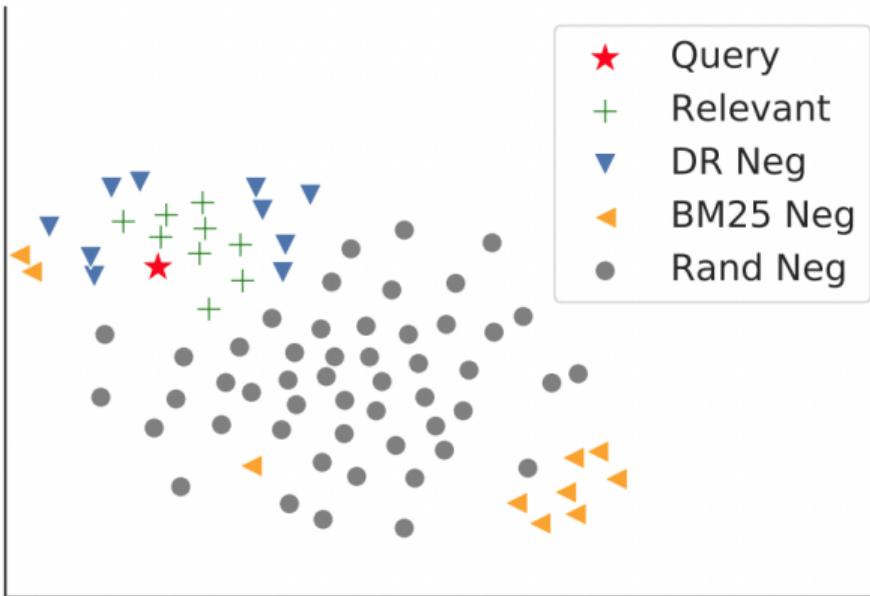


Figure: Representations of query, relevant documents, negative training instances from BM25 (BM25 Neg) or randomly sampled (Rand Neg), and testing negatives (DR Neg) in dense retrieval⁸

⁸<https://arxiv.org/abs/2007.00808>

Main challenges

- How to train a model to distinguish between relevant documents and irrelevant documents? → **Negative sampling**
- How to search through a massive index of document representations?
→ **Embedding index**

Dense retrieval

- Dense retrieval overview
- Negative sampling
- Embedding index
- Single representation systems
- Multiple representations systems

Embedding index

- *Embedding index* stores the pre-computed document embeddings and provides a search algorithm that, given a query embedding, efficiently finds the document embedding with the *maximum inner product* (MIP)
- There are different approaches for embedding index: flat index, nearest neighbor (NN) index, approximate nearest neighbor (ANN) index

Flat index

- Documents embeddings are stored explicitly in the flat index
- An exhaustive search is performed to find the document embedding with the maximum inner product
- The inner product search can be defined as:

$$\psi^* = \operatorname{argmax}_{\psi \in \Psi} \langle \psi, \phi \rangle$$

where $\phi \in \mathbb{R}^l$ denotes a query embedding, $\Psi = \psi_1, \dots, \psi_n$ denotes a set of n document embeddings, with $\psi_i \in \mathbb{R}^l$ for $i = 1, \dots, n$, and ψ^* denotes the maximum inner product

- It's particularly inefficient for large document collections

Nearest neighbor index

- A common approach to improve the space and time efficiency of the flat index is to convert the maximum inner product search into a nearest neighbor (NN) search
- The goal of NN search is to find the document embedding ψ^+ defined as:

$$\psi^+ = \operatorname{argmax}_{\psi \in \Psi} ||\phi - \psi||$$

- Many efficient index data structures exist for NN search. To leverage them with embedding indexes, MIP search between embeddings must be adapted to use the Euclidean distance and NN search.

Approximate nearest neighbor index

- The index data structures for exact NN search in low dimensional spaces have been very successful, but they are not efficient with high dimensional data, as in our case, due to the curse of dimensionality
- *Approximate nearest neighbor (ANN) search* is a natural compromise between search accuracy and search speed
- The ANN search approaches can be shared into three families: locality sensitive hashing, quantization and graph approaches

ANN search: Locality sensitive hashing approach

- Main idea: if two embeddings are close together, then after a "projection", using a hash function, these two embeddings will remain close together:
 - for any two embeddings that are close to each other, there is a high probability p_1 that they fall into the same hash bucket
 - for any two embeddings that are far apart, there is a low probability that they fall into the same hash bucket
- A set of random projections defines a family of hash function that can be used to build a data structure for ANN search
- The index is composed of hash tables, each containing concatenated random projections
- Main drawback: the index may require a large number of hash tables to cover most nearest neighbors

ANN search: Vector quantization approach

- Main idea: partitioning the input space Ψ (document embeddings) according to the data distribution:
 - Use k-means clustering algorithm to compute k centroids μ_1, \dots, μ_k
 - Use the centroids to partition the input space. The set of centroids is called a *codebook* $M = \mu_1, \dots, \mu_k$. A *vector quantiser* $q : \mathbb{R}^l \rightarrow \mathbb{R}^l$ maps a vector ψ to its closest centroid:

$$q(\psi) = \operatorname{argmax}_{\mu \in M} \|\psi - \mu\|$$

- An *Inverted File* (IVF) index is built over *codebook* M and input space Ψ by storing the set of document embeddings in k partitions or *inverted lists* L_1, \dots, L_k

$$L_i = \{\psi \in \Psi : q(\psi) = \mu_i\}$$

ANN search: Vector quantization approach - cont.

- At query processing time, we specify to search for the NN document embeddings in $p > 0$ partitions. If $p=k$, the search is exhaustive, but if $p < k$, the search is carried out in the partitions whose centroid is closer to the query embedding
- An IVF index does not improve the space consumption, since it still needs to store all document embeddings, but it can reduce the search time depending on the number of partitions processed for each query
- Main drawback: IVF indices can require a large number of centroids. This can be addressed with *product quantisation*.

ANN search: Graph approach

- Main idea: the distances between vectors in a dataset can be efficiently stored in a graph-based data structure called *kNN graph*
- In a kNN graph $G = (V, E)$, each input data $\psi \in \Psi$ is represented as a node $v \in V$, and, for its k nearest neighbours, a corresponding edge is added in E
- A search for an approximate nearest neighbor to a given element ψ in a kNN graph can be performed with *greedy heuristic search*
- In greedy heuristic search we start from a predefined entry node, we visit the graph one node at a time, and we keep on finding the closest node to ψ among the unvisited neighbour nodes. The search terminates when there is no improvement in the current NN candidate.

Dense retrieval

- Dense retrieval overview
- Negative sampling
- Embedding index
- Single representation systems
- Multiple representations systems

Different approaches for representation-focused systems

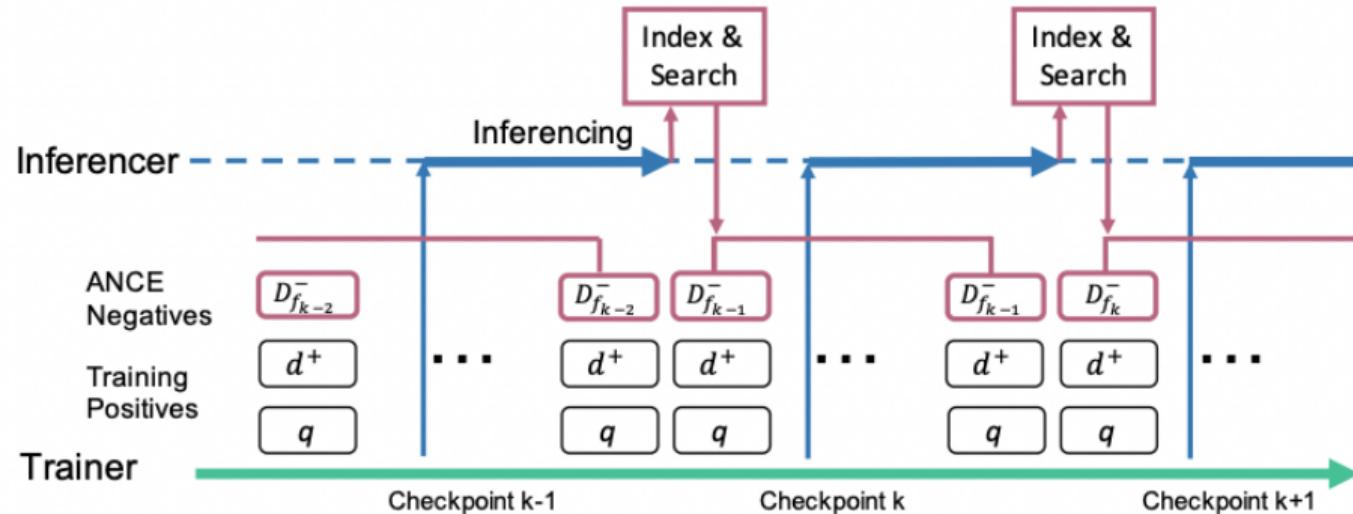
- Single representation systems - queries and documents are represented with a single embedding
- Multiple representations systems - queries and/or documents are represented with more than one embedding

Single representation

- Query representations and document representations are computed in the same latent vector space
- The representation functions are computed through fine-tuned encoder-only sequence-to-sequence models such as BERT
- Using these single representations, the relevance score between a query and a document is computed as a dot product of their output embeddings (corresponding to [CLS] tokens returned by the encoders)
- There are several different single-representation systems, e.g., DPR, ANCE, STAR

- Approximate nearest neighbor Negative Contrastive Learning (ANCE) is a contrastive learning mechanism for dense retrieval
- ANCE constructs global hard negatives using the being-optimized dense retrieval model retrieve from the entire corpus. It samples negatives from the top retrieved documents via the dense retriever model from the ANN index.
- ANCE uses an asynchronously updated ANN index of the collection representation. ANCE maintains an Inferencer that parallelly computes the document encodings with a recent checkpoint from the being optimized DR model, and refresh the ANN index used for negative sampling once it finishes, to keep up with the model training.
- ANCE can be used to train any dense retrieval model. The originally introduced system uses BERT Encoder, dot product similarity and negative log likelihood loss.

ANCE asynchronous training⁹



The Trainer learns the representation using negatives from the ANN index. The Inferencer uses a recent checkpoint to update the representation of documents in the corpus and once finished, refreshes the ANN index with most up-to-date encodings.

⁹<https://arxiv.org/abs/2007.00808>

Exercise

E9-2 Dense retrieval with ANCE

Dense retrieval

- Dense retrieval overview
- Negative sampling
- Embedding index
- Single representation systems
- Multiple representations systems

Dense Retrieval - multiple representations

- In contrast to single representation dense retrieval where a representation is assumed to incorporate the meaning of an entire text, multiple representations dense retrieval exploits more than a single embedding to represent a given text
- Multiple representations dense retrieval allows a richer semantic representation of the content
- Instead of using just the first output embedding (corresponding to the [CLS] token) to encode a document, this approach makes use of more output embeddings

Ranking pipeline architecture for multiple representation systems

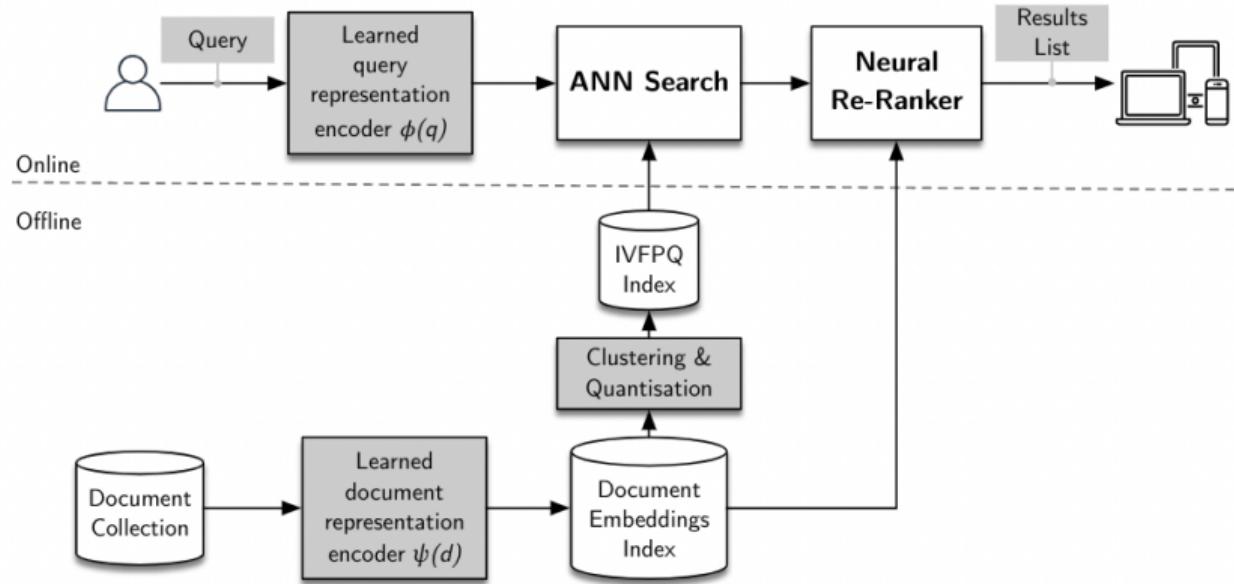


Figure: Ranking pipeline architecture for multiple representation systems¹⁰

¹⁰<https://arxiv.org/pdf/2207.13443.pdf>

CoBERT

- CoBERT uses all $1 + |d|$ output embeddings used to represent a document (of length d) - one output embedding per document token, including the [CLS] special token
- Similarly, query (of length q) is represented with $1 + |q|$ output embeddings
- CoBERT's late interaction scoring, also called *sum maxsim*, performs an all-to-all computation: each query embedding is dot-multiplied with every document embedding, and then the maximum computed dot products for each query embedding are summed up:

$$[\psi_0, \psi_1, \dots] = \text{Encoder}(q)$$

$$[\phi_0, \phi_1, \dots] = \text{Encoder}(d)$$

$$s(q, d) = \sum_{i=0}^{|q|} \max_{j=0, \dots, |d|} \psi_j \cdot \phi_j$$

ColBERT general architecture

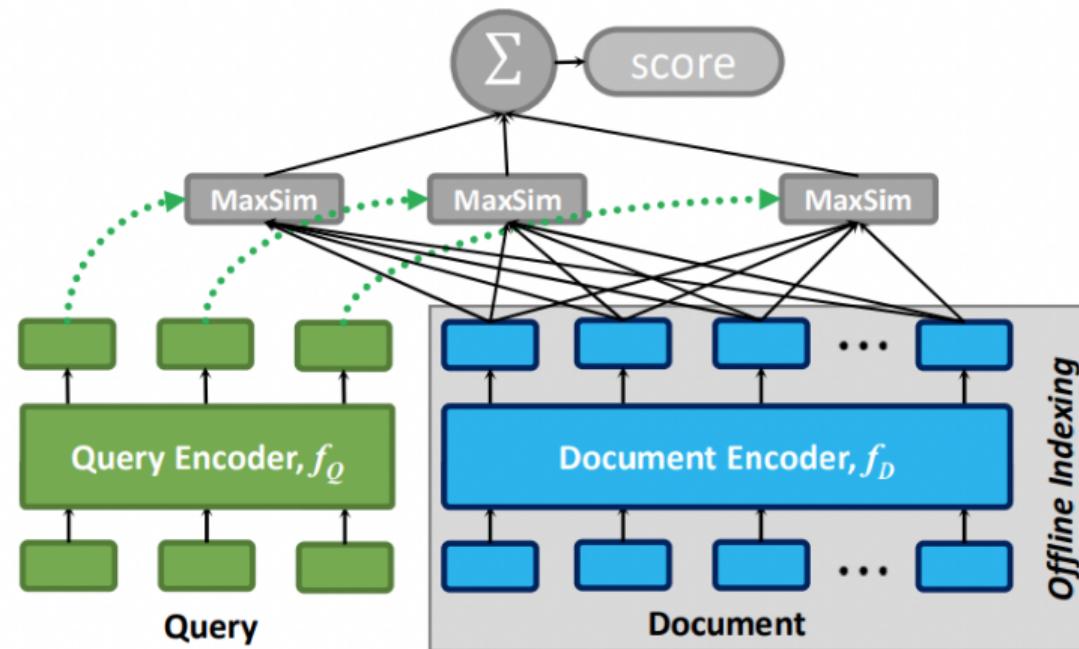


Figure: ColBERT architecture.¹¹

¹¹<https://dl.acm.org/doi/abs/10.1145/3397271.3401075>

Useful frameworks for dense retrieval

- FAISS framework for embedding indices (includes implementations of flat, LSH, IVF, PG indices)¹²
- Open-source search engines:
 - Lucene¹³
 - Vespa¹⁴

¹²<https://github.com/facebookresearch/faiss>

¹³<https://lucene.apache.org>

¹⁴<https://vespa.ai>

Summary

- Text representations for ranking
 - Recap on ranking and text representations
 - Transition from sparse to dense embeddings
 - Static word embeddings
 - Contextualized word embeddings
- Dense retrieval
 - Dense retrieval
 - Dense retrieval overview
 - Negative sampling
 - Embedding index
 - Single representation systems
 - Multiple representations systems

Reading

- *Lecture Notes on Neural Information Retrieval*, Nicola Tonello¹⁵
- *Vector Semantics and Embeddings*, Daniel Jurafsky and James H. Martin¹⁶

¹⁵<https://arxiv.org/pdf/2207.13443.pdf>

¹⁶<https://web.stanford.edu/~jurafsky/slp3/>

Neural Information Retrieval

[DAT640] Information Retrieval and Text Mining

Ivica Kostric & Weronika Lajewska
University of Stavanger

10.10.2022



CC BY 4.0

In this module

1. Interaction-focused ranking system
2. Learned sparse retrieval
3. Fusion strategies

Interaction-focused ranking system

Interaction-focused ranking system

- Main characteristics and high-level architecture
- Convolutional Neural Networks
- Transformer-based models

Interaction-focused Ranking System

- Interaction-focused ranking systems successfully improve the effectiveness of IR systems in the ad-hoc ranking task
- They use the word and n-gram relationships across a query and a document
- Both a query q and a document d are received as an input, and an output is a query-document representation $\mu(q, d)$

Re-ranking pipeline architecture

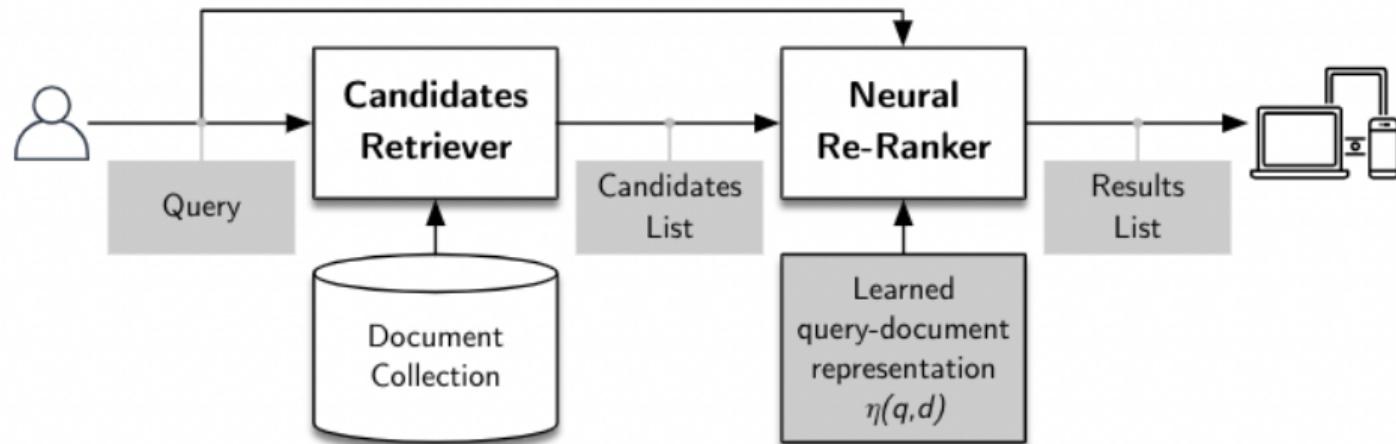


Figure: Re-ranking pipeline architecture for interaction-focused neural IR systems¹

¹<https://arxiv.org/pdf/2207.13443.pdf>

Retrieval architecture

Question

Why can't we just use them directly on the document collection to rank all documents matching a query?

Retrieval architecture - cont.

- Computationally very expensive, because the interaction has to be computed ad-hoc
- We deploy interaction-focused systems in the pipeline architecture only on the first top N candidates:
 - We conduct a preliminary ranking stage to retrieve a limited number of candidates (usually $N=1000$)
 - We re-rank the candidates using a more expensive neural re-ranking system

Interaction-focused Ranking System

- Generally done using deep neural networks
- Two neural network architectures have been investigated to build a representation of these relationships:
 - convolutional neural networks (CNNs)
 - transformers

Interaction-focused ranking system

- Main characteristics and high-level architecture
- Convolutional Neural Networks
- Transformer-based models

Convolutional Neural Networks (CNNs)

- A convolutional neural network is a family of neural networks designed to capture local patterns in structured inputs, such as images and texts
- The core component of a convolutional neural network is the convolution layer, used in conjunction with feed forward and pooling layers
- A convolutional layer can be seen as a small linear filter, sliding over the input and looking for *proximity patterns* (such as unigrams, bigrams, and so on)

CNN: architecture

- Several neural models employ convolutional neural networks over the interactions between queries and documents to produce relevance scores
- Typically, in these models, the word embeddings of the query and document tokens are aggregated into an *interaction matrix*
- On top of *interaction matrix* the convolutional neural networks are used to learn hierarchical *proximity patterns*
- The final top-level *proximity patterns* are fed into a feed forward neural network to produce the relevance score $s(q,d)$ between the query q and the document d

Interaction-focused model based on CNN

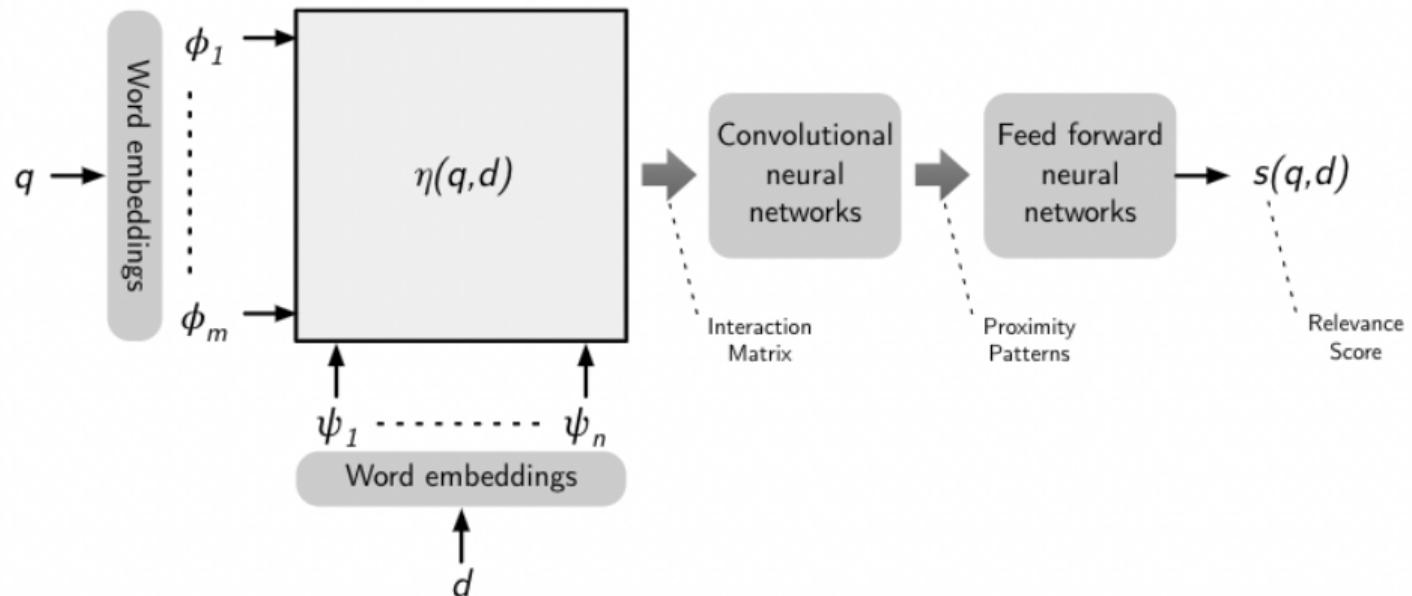


Figure: Scheme of an interaction-focused model based on convolutional neural networks.²

²<https://arxiv.org/pdf/2207.13443.pdf>

CNN: interaction matrix

- To construct the *interaction matrix* we need to:
 - tokenize the query q and the document d into m and n tokens respectively
 - map each token to a corresponding static word embedding
 - compute cosine similarities between a query token embedding and a document token embedding
- Each entry in the *interaction matrix* is the cosine similarity corresponding to the given query and document

Interaction-focused ranking system

- Main characteristics and high-level architecture
- Convolutional Neural Networks
- Transformer-based models

Recap on static and contextualized word embeddings

- Static word embeddings map words with multiple senses into an average or most common-sense representation based on the training data used to compute the vectors
- The static word embedding of a word does not change with the other words used in a sentence around it
- In contextualized word embeddings, the representation of each token is conditioned on the context in which it is considered.

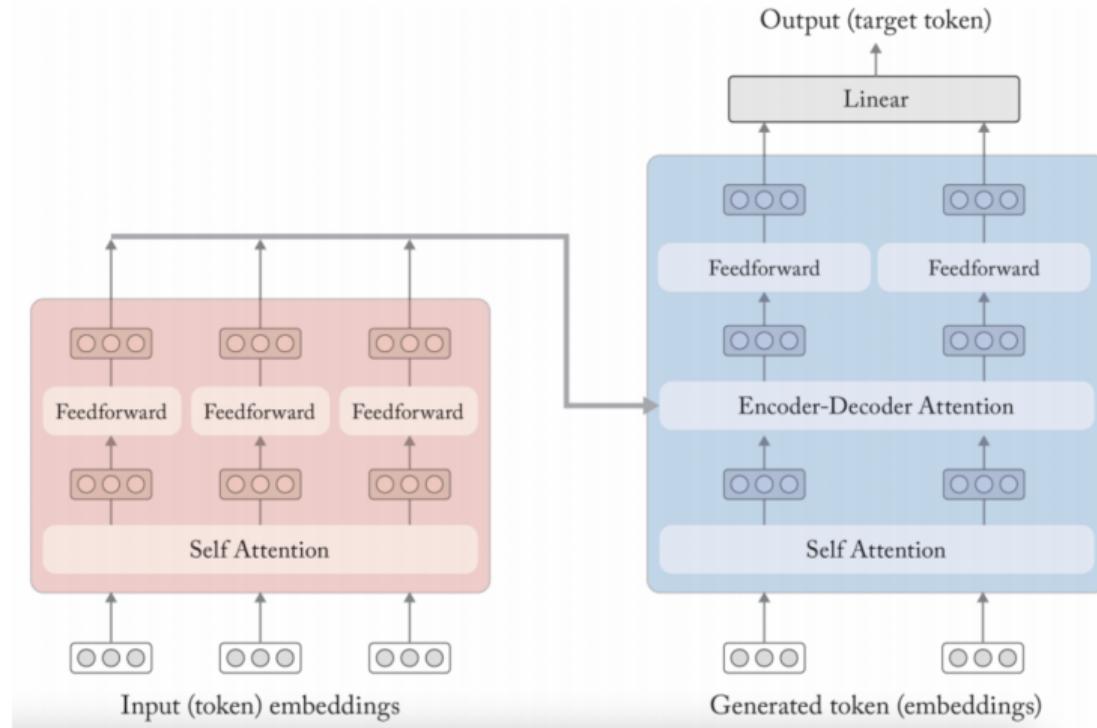
Transformers

- The transformer is a neural network designed to explicitly take into account the context of arbitrary long sequences of text
- Transformers are an example of *sequence-to-sequence* models that transforms an input vectors (x_1, \dots, x_n) to some output vectors (y_1, \dots, y_n) of the same length
- Transformers are made up of stacks of transformer blocks, which are multilayer networks made by combining simple linear layers, feedforward networks, and **attention layers**
- Attention is a key innovation of the transformers which allows to directly extract and use information from arbitrarily long contexts

Transformers

- Originally, the *sequence-to-sequence* neural network is composed of two parts: an *encoder* model, which receives an input sequence and builds a contextualised representation of each input element, and a *decoder* model, which uses these contextualised representations to generate a task-specific output sequence
- Three variants made for different purposes
 - Encoder-decoder architecture (e.g., machine translation)
 - Encoder only architecture (e.g., generating contextualized embeddings)
 - Decoder only architecture (e.g., language models)

High-level architecture of Transformer model ³



³<https://www.morganclaypool.com/doi/abs/10.2200/S01057ED1V01Y202009HLT047>

Transformers: self-attention

At the core of an attention-based approach is the ability to compare an item of interest to a collection of other items in a way that reveals their relevance in the current context. A self-attention layer allows the network to take into account the relationships among different elements in the same input.

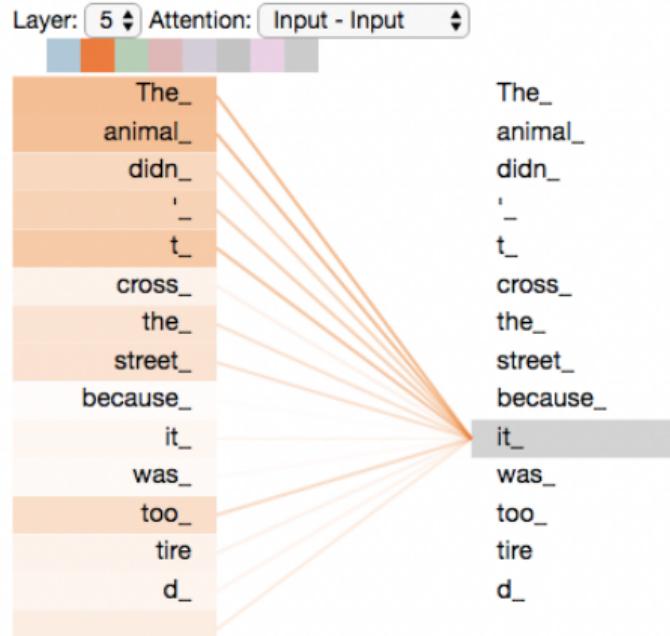


Figure: Self-attention at high level ^a

Transformers: self-attention - cont.

- A self-attention layer maps input sequences (x_1, \dots, x_n) to output sequences of the same length (y_1, \dots, y_n)
- Self attention differs in encoder and decoder modules
- The transformers in the *encoder* employ bidirectional self-attention layers on the input sequence or the output sequence of the previous transformer
- The transformers in the decoder employ causal self-attention on the previous decoder transformer's output and bidirectional cross-attention of the output of the final encoder transformer's output
 - In encoder module, the model has access to all of the inputs
 - In decoder module, the model has access to all of the inputs up to and including the one under consideration, but no access to information about inputs beyond the current one
- In addition, the computation performed for each item is independent of all the other computations → we can easily parallelize both forward inference and training of such models

Transformers: self-attention - cont.

- The simplest form of comparison between elements in a self-attention layer is a dot product
- Let's refer to the result of this comparison as a score: $score(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
- We can compute these scores for all pairs i, j (or $j \leq i$ in decoder self-attention) and normalize them with a *softmax* to create a vector of weights α_{ij} , that indicates the proportional relevance of each input to the input element i that is the current focus of attention:

$$\alpha_{ij} = \forall_j softmax(score(\mathbf{x}_i, \mathbf{x}_j)) = \forall_j \frac{exp(score(\mathbf{x}_i, \mathbf{x}_j))}{\sum_k exp(score(\mathbf{x}_i, \mathbf{x}_k))}$$

- Given the proportional scores in α , we then generate an output value y_i by taking the sum of the inputs seen so far, weighted by their respective α value

$$\mathbf{y}_i = \sum_j \alpha_{ij} \cdot \mathbf{x}_j$$

Transformers: self-attention - cont.

- Transformers use a bit more sophisticated way of representing how words can contribute to the representation of longer input
- Input embedding can play three different roles during the course of the attention process:
 - As the current focus of attention when being compared to all of the other query preceding inputs →**query**
 - As a preceding input being compared to the current focus of attention →**key**
 - As a value used to compute the output for the current focus of attention →**value**
- To capture these three different roles, transformers introduce weight matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$. These weights will be used to project each input vector \mathbf{x}_i into a representation of its role as a key, query, or value

$$\mathbf{q}_i = \mathbf{W}^Q \cdot \mathbf{x}_i; \mathbf{k}_i = \mathbf{W}^K \cdot \mathbf{x}_i; \mathbf{v}_i = \mathbf{W}^V \cdot \mathbf{x}_i$$

Transformers: self-attention - cont.⁴

Given these projections, the score between the current focus of attention, x_i and an element in the context, x_j consists of a dot product between its query vector q_i and the context element's key vectors k_j :

$$\text{score}(x_i, x_j) = q_i \cdot k_j$$

The ensuing softmax calculation resulting in α_{ij} remains the same, but the output calculation for y_i is now based on a weighted sum over the value vectors v :

$$y_i = \sum_{j \leq i} \alpha_{ji} \cdot v_j$$

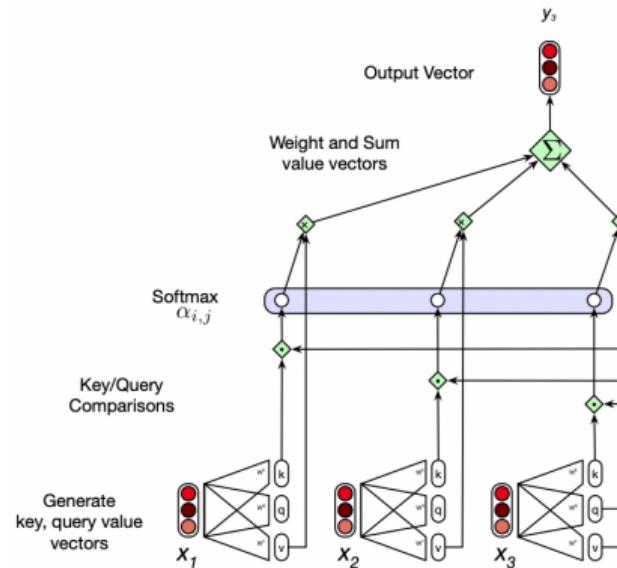


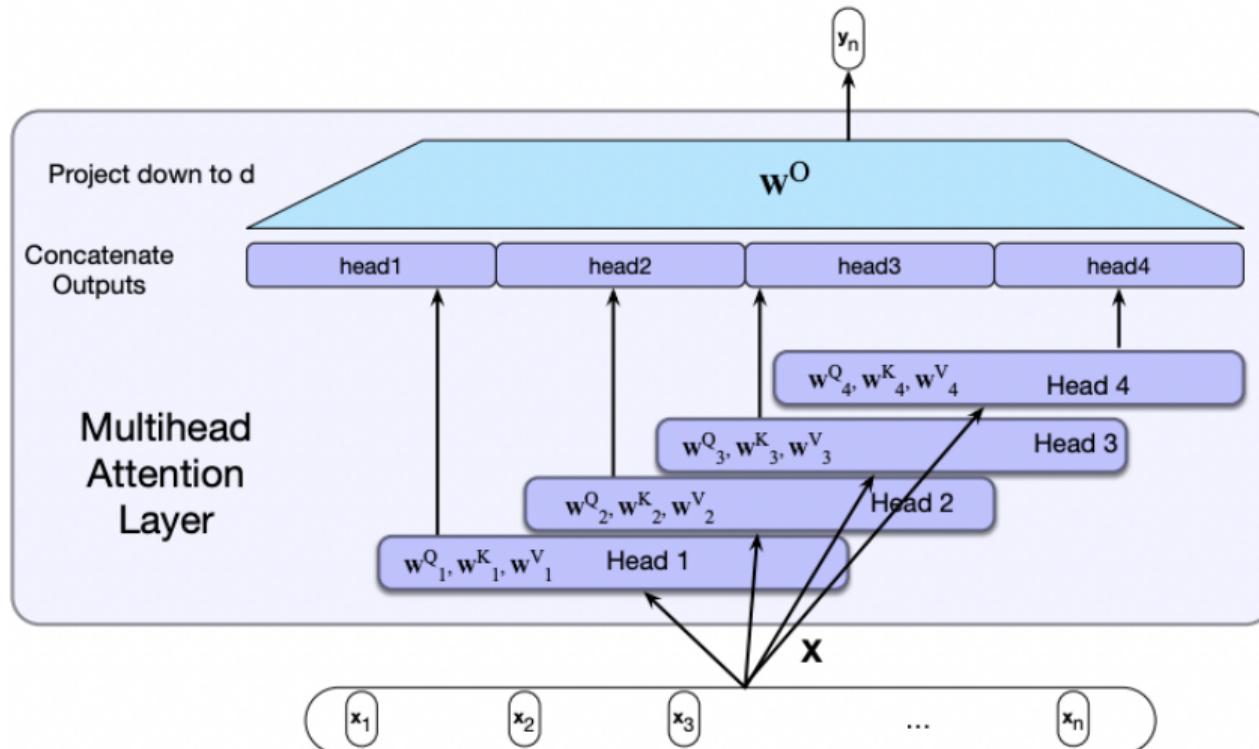
Figure: Self-attention when considering the input x_3

⁴<https://arxiv.org/pdf/2207.13443.pdf>

Transformers: multihead self-attention

- The different words in a sentence can relate to each other in many different ways simultaneously
- It would be difficult for a single transformer block to learn to capture all of the different kinds of parallel relations among its inputs
- Transformers address this issue with *multihead self-attention layers*
- *Multihead self-attention layers* are sets of self-attention layers, called heads, that reside in parallel layers at the same depth in a model, each with its own set of parameters
- Each of the multihead self-attention layers is provided with its own set of key, query and value weight matrices
- The outputs from each of the layers are concatenated and then projected down to the same size as the input so layers can be stacked

Transformers: multihead self-attention⁵

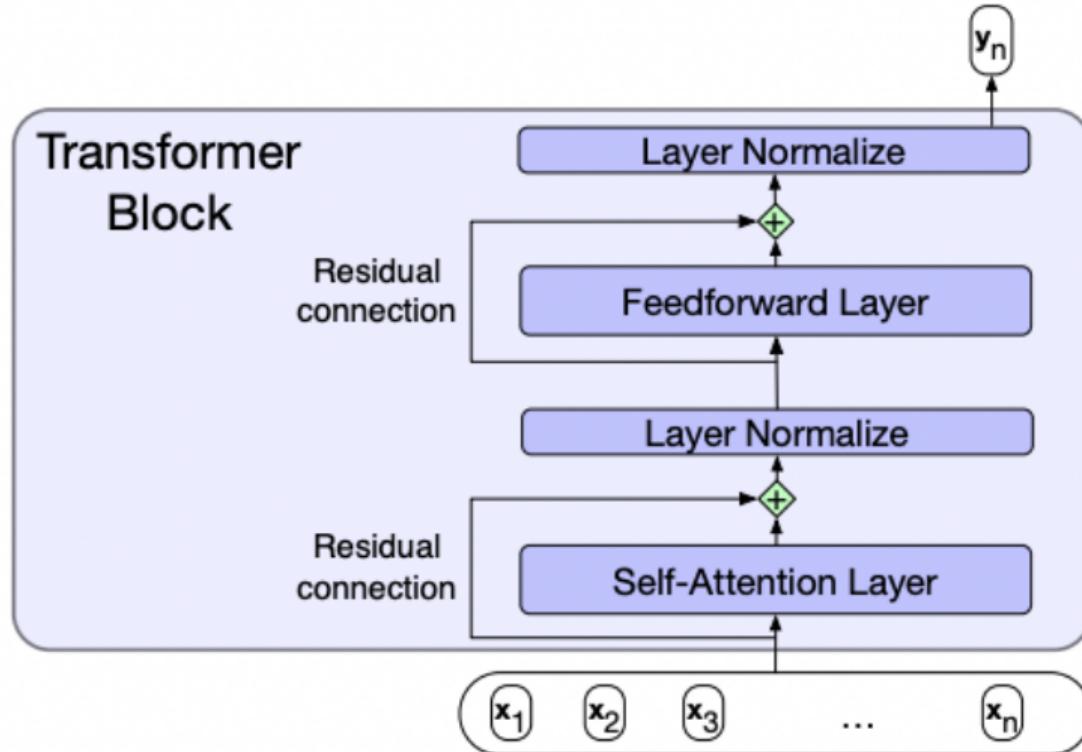


⁵<https://web.stanford.edu/~jurafsky/slp3/>

Transformers: transformer block

- The self-attention calculation lies at the core of what's called a *transformer block*, which, in addition to the self-attention layer, includes additional feedforward layers, residual connections, and normalizing layers
- A standard *transformer block* consists of a single attention layer followed by a fully-connected feedforward layer with residual connections and layer normalizations following each

Transformers: transformer block⁶



⁶<https://web.stanford.edu/~jurafsky/slp3>

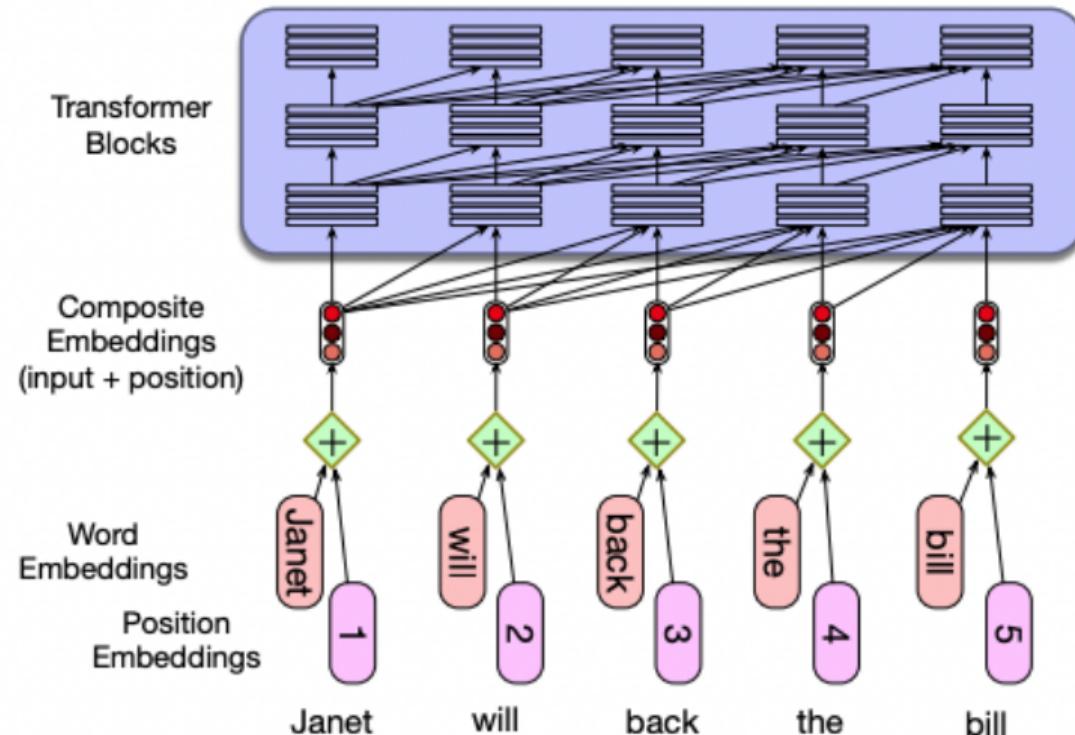
Transformers: transformer block - cont.

- In deep networks, residual connections are connections that pass information from a lower layer to a higher layer without going through the intermediate layer
- Allowing information from the activation going forward and the gradient going backwards to skip a layer improves learning and gives higher level layers direct access to information from lower layers
- Residual connections in transformers are implemented by adding a layer's input vector to its output vector before passing it forward
- *Layer normalization* is used to improve training performance in deep neural networks by keeping the values of a hidden layer in a range that facilitates gradient-based training

Transformers: positional embeddings

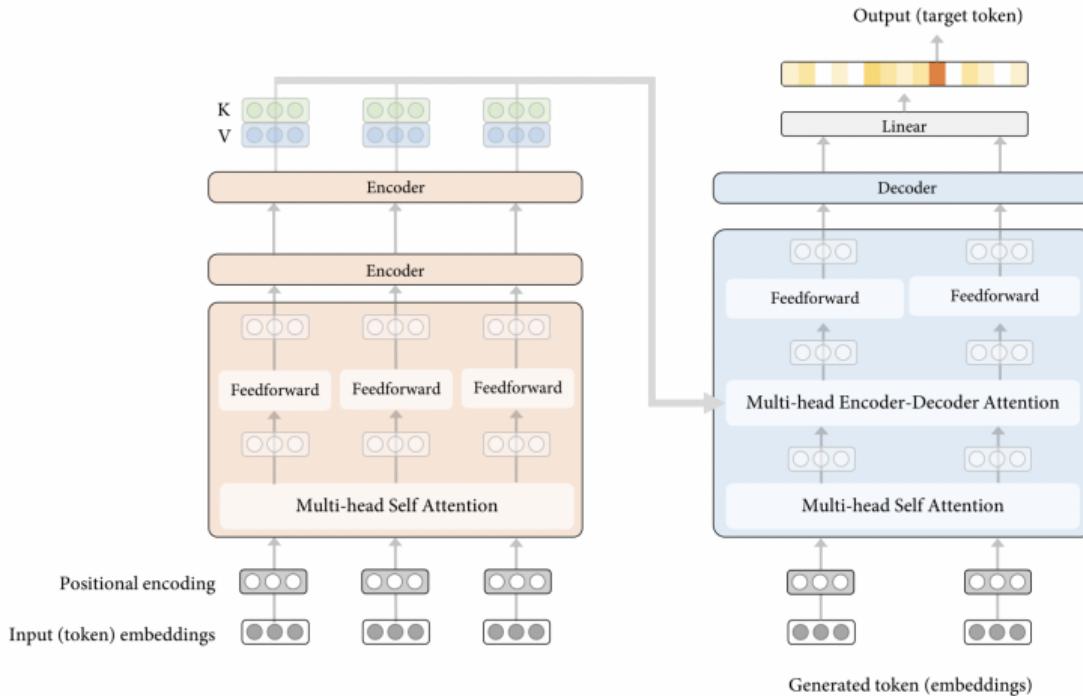
- Transformers modify the input embeddings by combining them with *positional embeddings* specific to each position in an input sequence
- The simplest way to model the position is to:
 - start with randomly initialized embeddings corresponding to each possible input position up to some maximum length
 - and learn the positional embeddings along with other parameters during training
- To produce an input embedding that captures positional information, we just add the word embedding for each input to its corresponding positional embedding

Transformers: positional embeddings⁷



⁷<https://web.stanford.edu/~jurafsky/slp3>

High-level architecture of Transformer-based encoder-decoder model⁸



⁸<https://aclanthology.org/2020.coling-tutorials.2/>

Pre-trained Language Models

- Transformer-based models are usually trained using massive text data to obtain *pre-trained language models*
- It allows the model to learn general-purpose knowledge about a language that can be adapted afterwards to a more specific *downstream task*
- This approach is called *transfer learning*
- In *transfer learning*, a pre-trained language model is used as initial model to *fine-tune* it on a domain-specific, smaller training dataset for the downstream target task
- *Fine-tuning* is the procedure to update the parameters of a pre-trained language model for the domain data and target task

Transfer learning⁹



⁹<https://arxiv.org/pdf/2207.13443.pdf>

Demo

HuggingFace Transformers¹⁰

¹⁰<https://transformer.huggingface.co>

Fine-tuning interaction-focused systems

- The pre-trained language models adopted in IR require a fine-tuning on a specific downstream task
- Given an input query-document pair (q, d) a neural IR model $M(\theta)$, parametrised by θ , computes $s_\theta(q, d) \in \mathbb{R}$, the score of the document d w.r.t. the query q
- We want to predict $y \in \{+, -\}$ from $(q, d) \in Q \times D$, where “-” stands for non-relevant and “+” for relevant
- We perform the classification by assuming a joint distribution p over $\{+, -\} \times Q \times D$ from which we can sample correct pairs $(+, q, d) \equiv (q, d^+)$ and $(-, q, d) \equiv (q, d^-)$
- Training objective: *the score function must assign a high score to a relevant document and a low score to a non-relevant document*

Fine-tuning interaction-focused systems - cont.

- We need to find θ^* that minimises the (binary) cross entropy I_{CE} between the conditional probability $p(y|q, d)$ and the model probability $p_\theta(y|q, d)$:

$$\theta^* = \operatorname{argmin}_\theta \mathbb{E}[I_{CE}(y, q, d)]$$

where the expectation is computed over $(y, q, d) \sim p$, and the cross entropy is computed as:

$$I_{CE}(y, (q, d)) = \begin{cases} -\log(s_\theta(q, d)) & \text{if } y=+ \\ -\log(1 - s_\theta(q, d)) & \text{if } y=- \end{cases} \quad (1)$$

Fine-tuning interaction-focused systems - cont.

- Typically, a dataset T available for fine-tuning pre-trained language models for relevance scoring is composed of a list of triples (q, d^+, d^-) , where q is a query, d^+ is a relevant document for the query, and d^- is a non-relevant document for the query
- In this case, the expected cross entropy is approximated by the sum of the cross entropies computed for each triple:

$$\mathbb{E}[I_{CE}(y, (q, d))] \approx \frac{1}{2|T|} \sum_{(q, d^+, d) \in T} (-\log(s_\theta(q, d^+)) - \log(1 - s_\theta(q, d^-)))$$

Transformers: sequence-to-sequence models

- In neural IR, two specific instances of the sequence-to-sequence models have been studied:
 - encoder-only models
 - encoder-decoder models

Transformers: encoder-only models

- *Encoder-only* models receive as input all the tokens of a given input sentence, and they compute an output contextualised word embedding for each token in the input sentence
- Representatives of this family of models include BERT, RoBERTa, and DistilBERT
- Encoder-only models are trained as *masked language models (MLM)*
- MLM training focuses on learning to predict missing tokens in a sequence given the surrounding tokens

BERT model

- BERT input text is tokenised using the WordPiece sub-word tokeniser
- The vocabulary V of this tokeniser is composed of 30,522 terms, where the uncommon/rare words, e.g., *goldfish*, are splitted up in sub-words, e.g., *gold##* and *##fish*
- The first input token of BERT is always the special $[CLS]$ token, that stands for “classification”
- BERT accepts as input other special tokens, such as $[SEP]$, that denotes the end of a text provided as input or to separate two different texts provided as a single input
- BERT accepts as input at most 512 tokens, and produces an output embedding in \mathbb{R}^d for each input token
- The most commonly adopted BERT version is *BERT-base*, which stacks 12 transformer layers, and whose output representation space has $d = 768$ dimensions

BERT model - cont.

- Given a query-document pair, both texts are tokenised into token sequences q_1, \dots, q_m and d_1, \dots, d_n
- Then, the tokens are concatenated with BERT special tokens to form the following input configuration that will be used as BERT input:

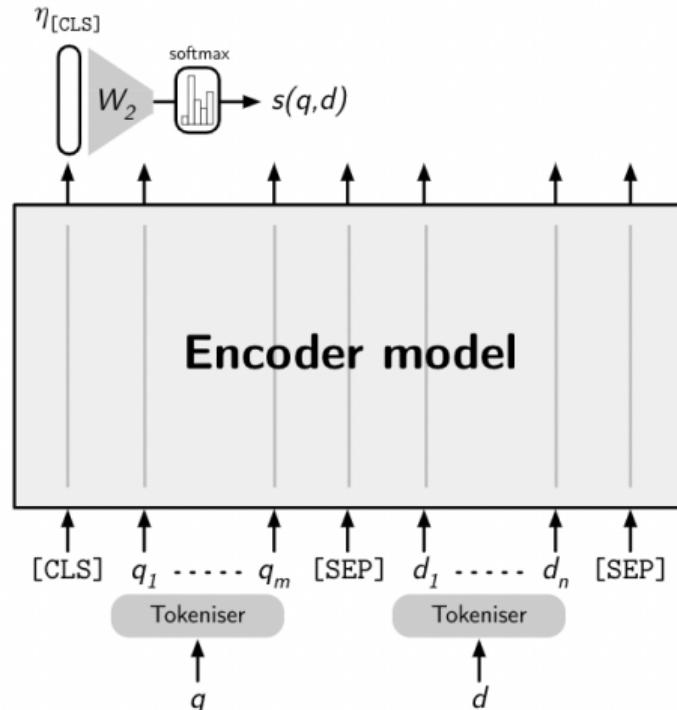
$$[CLS]q_1, \dots, q_m[SEP]d_1, \dots, d_n[SEP]$$

- The self-attention layers in the BERT encoders are able to take into account the semantic interactions among the query tokens and the document tokens
 - The attention mechanism in encoder models is bi-directional
- The output embedding $\mu_{[CLS]} \in \mathbb{R}^I$, corresponding to the input $[CLS]$ token, serves as a contextual representation of the query-document pair as a whole

Fine-tuning BERT model for computing relevance score

- The query and the document are processed by BERT to generate the output embedding $\mu_{[CLS]} \in \mathbb{R}^I$
- $\mu_{[CLS]}$ is multiplied by a learned set of classification weights $W_2 \in \mathbb{R}^{2 \times I}$ to produce two real scores z_0 and z_1
- The scores z_0 and z_1 are transformed into a probability distribution p_0 and p_1 over the non-relevant and relevant classes
- The probability corresponding to the relevant class, conventionally assigned to label 1, i.e., p_1 , is the final relevance score

BERT classification model for ad-hoc ranking¹¹



¹¹<https://arxiv.org/pdf/2207.13443.pdf>

Exercise

E10-1 - WordPiece Tokenizer

Transformers: encoder-decoder models

- In the *encoder-decoder* model the encoder model receives as input all the tokens of a given sequence and builds a contextualised representation, and the decoder model sequentially accesses these embeddings to generate new output tokens, one token at a time
- Representatives of this family of models include BART and T5
- Encoder-decoder models are trained as *casual language models (CLM)*
- CLM training focuses on predicting the next token in an output sequence given the preceding tokens in the input sequence
- Encoder-decoder models use *prompt learning* which converts the relevance score computation task into a cloze test, i.e., a fill-in-the-blank problem

Encoder-decoder models - prompt learning

- In *prompt learning* the input texts are reshaped as a natural language template, and the downstream task is reshaped as a cloze-like task
- For example, in topic classification, assuming we need to classify the sentence *text* into two classes c_0 and c_1 , the input template can be: *Input : text Class : [OUT]*
- Among the vocabulary terms, two label terms w_0 and w_1 are selected to correspond to the classes c_0 and c_1 , respectively
- The probability to assign the input text to a class can be transferred into the probability that the input token *[OUT]* is assigned to the corresponding label token:

$$p(c_0 | \text{text}) = p([\text{OUT}] = w_0 | \text{Input : text Class : [OUT]})$$

$$p(c_1 | \text{text}) = p([\text{OUT}] = w_1 | \text{Input : text Class : [OUT]})$$

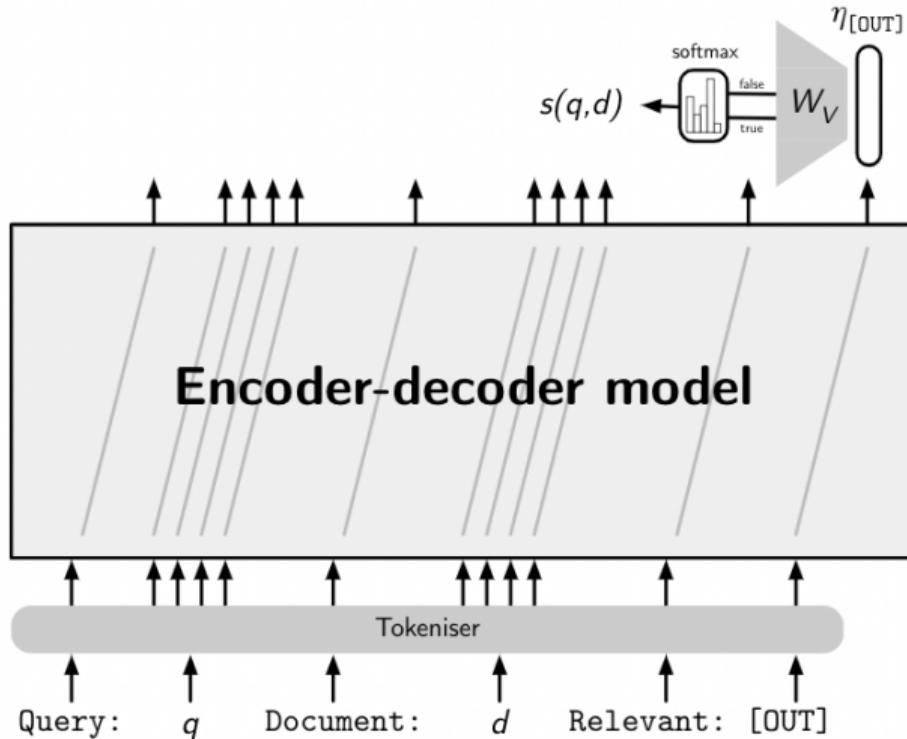
Fine-tuning T5 model for computing relevance score

- In prompt learning approach for relevance ranking using a T5 model, the query and the document texts q and d are concatenated to form the following input template:

Query : q Document : d Relevant : [OUT]

- An encoder-decoder model is fine-tuned with a downstream task taking as input this input configuration, and generating an output sequence whose last token is equal to *True* or *False*, depending on whether the document d is relevant or non-relevant to the query q
- The query-document relevance score is computed by normalising only the *False* and *True* output probabilities, computed over the whole vocabulary, with a *softmax* operation

T5 model for ad-hoc ranking¹²

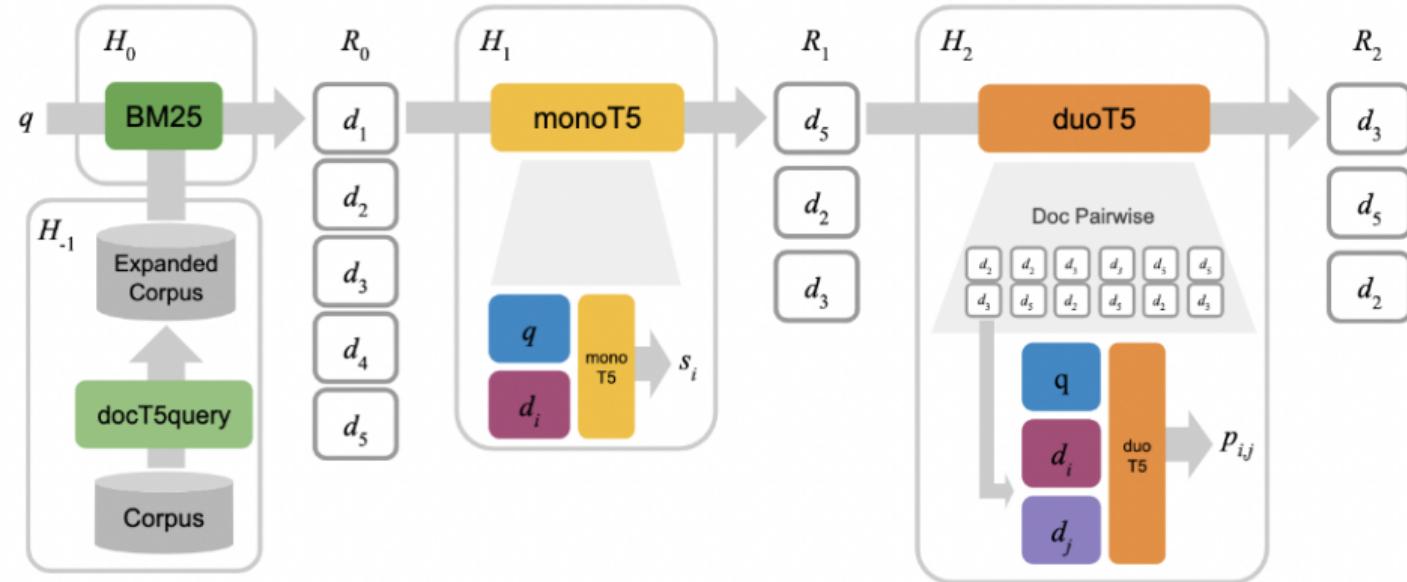


¹²<https://arxiv.org/pdf/2207.13443.pdf>

Pointwise and pairwise re-ranking

- Re-rankers play a crucial role in pushing the most relevant passages to the top of the ranking
- We distinguish two classes of the reranking methods:
 - *pointwise reranking*, e.g. monoT5, where the relevance of each passage in the ranking with respect to the query is computed independently of other passages
 $P(\text{Relevant} = 1 | d_i, q)$
 - *pairwise reranking*, e.g. duoT5, where the relevance is considered in terms of pairs of passages $P(d_i > d_j = 1 | d_i, d_j, q)$
- Pointwise reranking with monoT5 is highly effective with different first-stage retrieval results
- Pairwise reranking with duoT5 further improves the effectiveness of monoT5 output, particular in early-precision metrics

Re-ranking with mono/duoT5¹³



¹³<https://www.semanticscholar.org/reader/e08eed9608382beea1febca49119c665fbabd031>

Learned sparse retrieval

Learned Sparse Retrieval

- In industry-scale web search, BM25 is a widely adopted baseline due to its trade-off between effectiveness and efficiency
- On the other side, neural IR systems are based on dense representations of queries and documents, that have shown impressive benefits in search effectiveness, but at the cost of query processing times
- In recent years, there have been some proposals to incorporate the effectiveness improvements of neural networks into inverted indexes, with their efficient query processing algorithms, through *learned sparse retrieval* approaches

Learned Sparse Retrieval

- In learned sparse retrieval the transformer architectures are used in different scenarios:
 - document expansion learning
 - impact score learning
 - sparse representation learning

Learned sparse retrieval

- Document Expansion Learning
- Impact Score Learning
- Sparse Representation Learning

Document Expansion Learning

- *Document expansion techniques* address the vocabulary mismatch problem:
queries can use terms semantically similar but lexically different from those used in the relevant documents
- Traditionally, this problem has been addressed using query expansion techniques, such as relevance feedback and pseudo-relevance feedback
- In *document expansion learning*, sequence-to-sequence models are used to modify the actual content of documents, boosting the statistics of the important terms and generating new terms to be included in a document

Document Expansion Learning: Doc2Query

- Transformer architectures can be used to expand the documents' content to include new terms
- Doc2Query generate new queries for which a specific document will be relevant
- Doc2Query fine-tunes a sequence-to-sequence transformer model by taking as input the relevant document and generating the corresponding query
- Then, the fine-tuned model is used to predict new queries using top k random sampling to enrich the document by appending these queries before indexing

Document Expansion Learning: Doc2Query

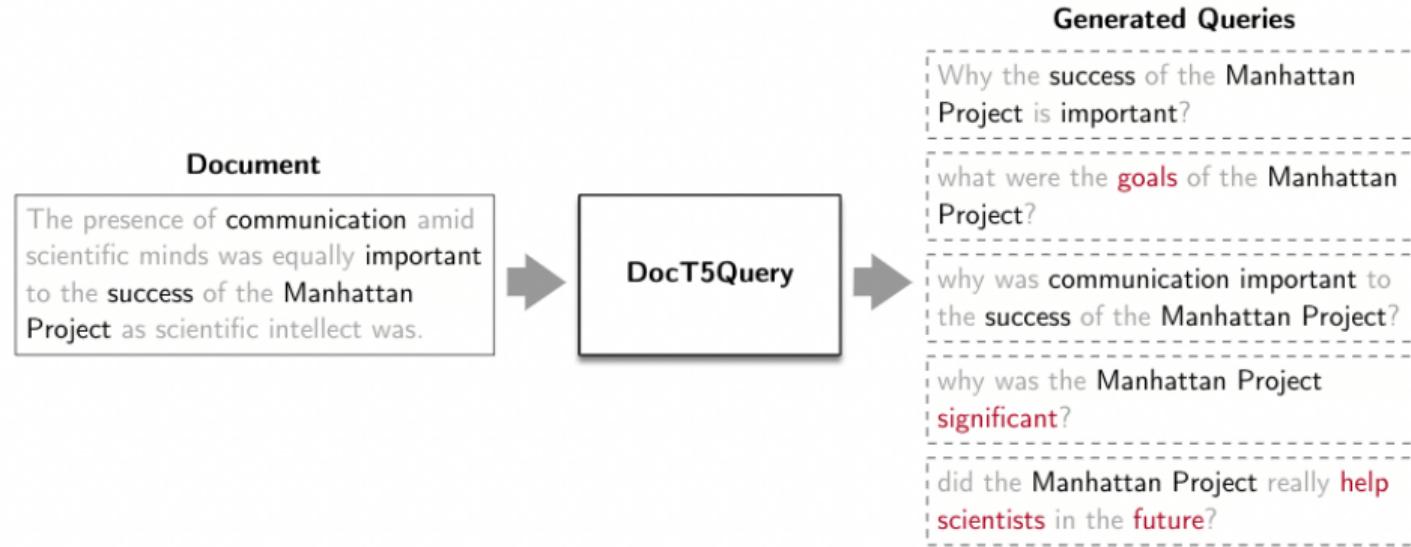


Figure: Example of generated queries using DocT5Query. Black terms denote boosted important terms, red terms denote new important terms not present in the original document¹⁴

¹⁴<https://arxiv.org/pdf/2207.13443.pdf>

Exercise

E10-2 - Document expansion with Doc2Query

Learned sparse retrieval

- Document Expansion Learning
- Impact Score Learning
- Sparse Representation Learning

Impact Score Learning: Recap on inverted index

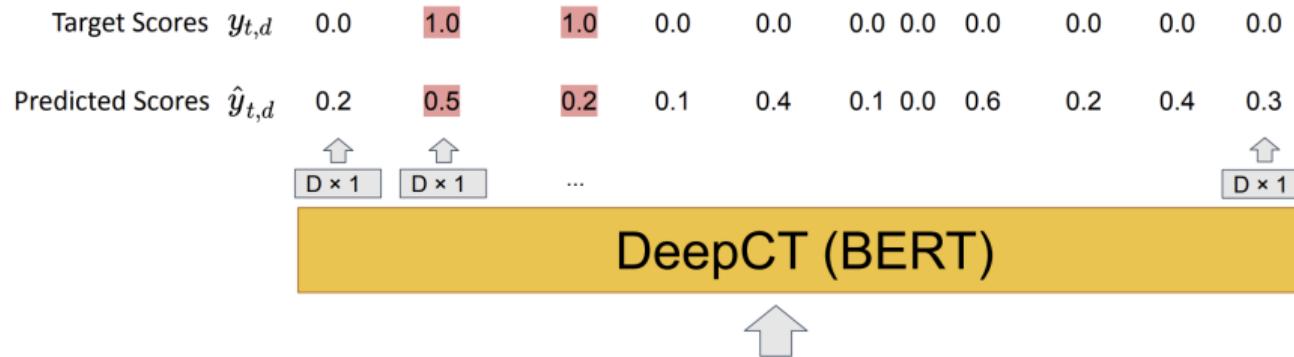
- Classical inverted indexes store statistical information on term occurrences in documents in posting lists, one per term in the collection
- Every posting list stores a posting for each document in which the corresponding term appears in, and the posting contains a document identifier and the in-document term frequency, i.e., a positive integer counting the number of occurrences of the term in the document
- When a new query arrives, the posting lists of the terms in the query are processed to compute the top scoring documents

Impact Score Learning

- The goal of *impact score learning* is to leverage the document embeddings generated by an encoder-only model to compute a single integer value to be stored in postings, and to be used as a proxy of the relevance of the term in the corresponding posting, i.e., its *term importance*
- The simplest way to compute term importance in a document is to project the document embeddings of each term with a neural network into a single-value representation, filtering out negative values with *ReLU* functions and discarding zeros
- The collection vocabulary that is used by impact score learning can be constructed in two different ways: by using the terms produced by the encoder-specific sub-word tokeniser, e.g., by BERT-like tokenisers, or by using the terms produced by a word tokeniser
- These two alternatives have an impact on the final inverted index: in the former case, we have fewer terms, but longer and denser posting lists, while in the latter case, we have more terms, with shorter posting lists and with smaller query processing times

Impact Score Learning: DeepCT¹⁵

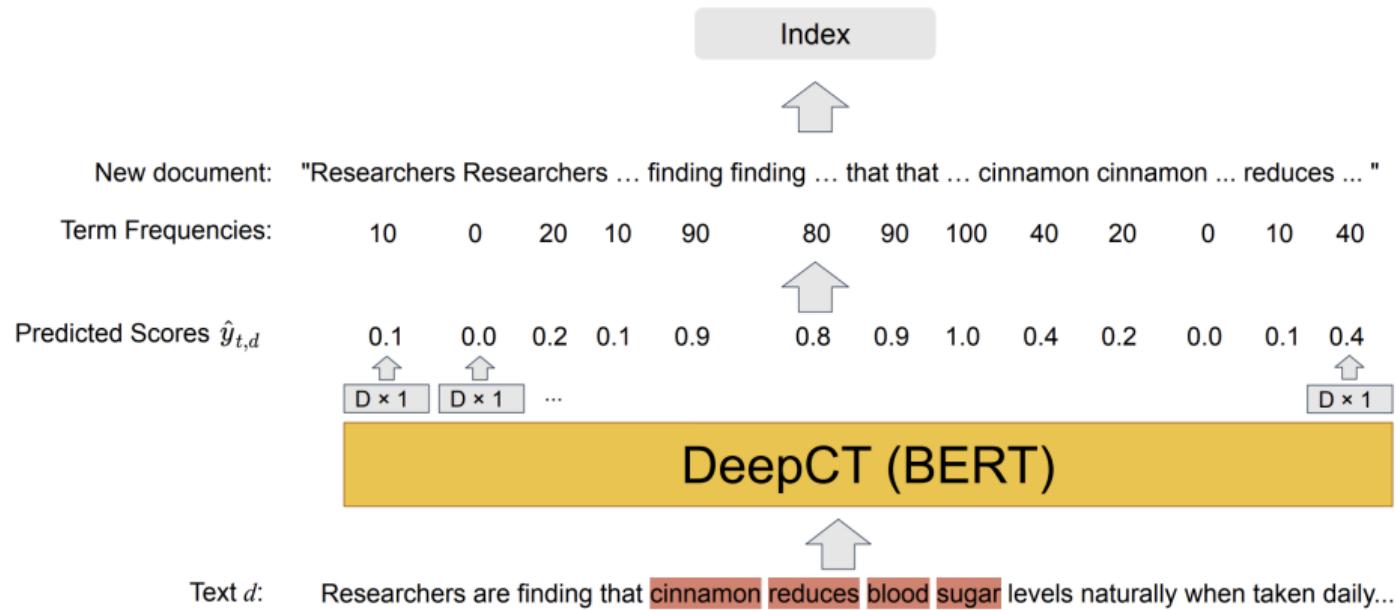
$$\text{loss} = \sum_t (\hat{y}_{t,d} - y_{t,d})^2$$



Relevant query q : "who proposed the geocentric theory"

¹⁵<http://essir2022.org/slides/andrew-yates.pdf>

Impact Score Learning: Indexing with DeepCT¹⁶



¹⁶<http://essir2022.org/slides/andrew-yates.pdf>

Impact Score Learning

- To sum up: in *impact score learning*, the output embeddings of documents provided as input to encoder-only models are further transformed with neural networks to generate a single real value, used to estimate the average relevance contribution of the term in the document

Learned sparse retrieval

- Document Expansion Learning
- Impact Score Learning
- Sparse Representation Learning

Sparse Representation Learning

- Instead of independently learning to expand the documents and then learning the impact score of the terms in the expanded documents, sparse representation learning aims at learning both at the same time
- In general, *sparse representation learning* projects the output embeddings of an encoder-only model into the input vocabulary and computes, for each input term in the document, a language model, i.e., a probability distribution over the whole vocabulary
- These term-based language models capture the semantic correlations between the input term and all other terms in the collection, and they can be used to:
 - expand the input text with highly correlated terms
 - compress the input text by removing terms with low probabilities w.r.t. the other terms

Sparse Representation Learning: SPLADE¹⁷

SPLADE: Sparse Lexical and Expansion Model

Key improvement: leverage the **MLM head** for weighting and expansion

- Recall that MLM head predicts what term should fill in a [MASK]

Input: a query Q or document D

Output: a sparse vector of dimension $|V|$, to be indexed

1. For each term in the input, use MLM head to predict scores for $|V|$ terms
2. For each term in the vocabulary V, take maximum score as the term's weight
3. Use weights from #2 to represent the input (Q or D) as a $|V|$ vector; index

¹⁷<http://essir2022.org/slides/andrew-yates.pdf>

Sparse Representation Learning

- To sum up: in *sparse representation learning*, the output embeddings of documents provided as input to encoder-only models are projected with a learned matrix on the collection vocabulary, in order to estimate the relevant terms in a document
- These relevant terms can be part of the documents or not, hence representing another form of document enrichment

Fusion strategies

Fusion strategies

- Both, sparse retrieval methods and dense retrieval methods have their advantage
- To exploit all of the advantages, we can perform a hybrid retrieval with a sparse method and a dense method
- The final retrieval result will be the “combination” of the two

Fusion strategies: Reciprocal Rank Fusion

- Reciprocal Rank Fusion (RRF) is a simple method for combining the document rankings from multiple IR systems
- Given a set of documents D to be ranked and a set of rankings R , each a permutation on $1, \dots, |D|$, RRF sorts the documents according to the scoring formula:

$$RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)}$$

- r is the rank of a document d in one ranking
- k is a smoothing factor that mitigates the impact of high rankings by outlier systems ($k = 60$ in the original paper)
- Intuition: while highly ranked documents are more important, the importance of lower-ranked documents does not vanish (as it would if we use for example an exponential function)

Summary

- Interaction-focused ranking systems
 - Main characteristics and high-level architecture
 - Convolutional Neural Networks
 - Transformer-based models
- Learned sparse retrieval
 - Document Expansion Learning
 - Impact Score Learning
 - Sparse Representation Learning
- Fusion strategies

Reading

- *Lecture Notes on Neural Information Retrieval*, Nicola Tonello¹⁸
- *Vector Semantics and Embeddings*, Daniel Jurafsky and James H. Martin ¹⁹

¹⁸<https://arxiv.org/pdf/2207.13443.pdf>

¹⁹<https://web.stanford.edu/~jurafsky/slp3/>

Conversational Information Access

[DAT640] Information Retrieval and Text Mining

Ivica Kostric & Krisztian Balog
University of Stavanger

17.10.2022



CC BY 4.0

In this module

1. Conversational AI
2. Architecture
3. Examples

Conversational AI

Recap

- So far the focus was on single turn IR techniques
 - Represent documents (items) and queries
 - Rank documents (items) with respect to queries
 - Evaluate the goodness of rankings
- Many information needs are not answered by a single query

Question

What is conversational AI?

Conversational AI

- The branch of computer science concerned with making machines interact with people in a human-like way
- Can use speech, text or mixed interfaces
- Design and abilities vary
 - Task-oriented dialogue agents (e.g., Siri, Alexa, Google Now/Home, Cortana)
 - Chatbots (e.g., ELIZA, PARRY, Xiaolce, BlenderBot)

Conversational AI - Traditional distinction¹

Task-oriented (goal-driven)

- Aim to assist users to **solve a specific task** (as efficiently as possible)
- Dialogues follow a **clearly designed structure** (flow) that is developed for a particular task in a **closed domain**
- Well-defined measure of performance that is explicitly related to **task completion**

Non-task-oriented (non-goal-driven)

- Aim to **carry on an extended conversation** ("chit-chat") with the goal of mimicking human-human interactions
- Developed for **unstructured**, open domain conversations
- Objective is to be **human-like**, i.e., able to talk about different topics (breadth and depth) in an engaging and coherent manner

¹Chen et al. A Survey on Dialogue Systems: Recent Advances and New Frontiers. SIGKDD Explor. Newslett. 19(2), 2017.

Conversational AI - Contemporary distinction²

Task-oriented

- Aim to assist users to **solve a specific task** (as efficiently as possible)
- Dialogues follow a **clearly designed structure** (flow) that is developed for a particular task in a **closed domain**
- Well-defined measure of performance that is explicitly related to **task completion**

Social chat

- Aim to **carry on an extended conversation** ("chit-chat") with the goal of mimicking human-human interactions
- Developed for **unstructured, open domain** conversations
- Objective is to be **human-like**, i.e., able to talk about different topics (breadth and depth) in an engaging and coherent manner

Interactive QA

- Aim to **provide concise, direct answers** to user queries
- Dialogues are **unstructured**, but commonly follow a question-answer pattern; mostly **open domain** (dictated by the underlying data)
- Evaluated with respect to the **correctness of answers** (on the turn level)

²Gao et al. Neural Approaches to Conversational AI. Found. Trends Inf. Retr. 13(2-3), 2019.

Question

What is conversational information access?

Question

Where does conversational information access fit within conversational AI?

Conversational information access

Task-oriented

- Aim to assist users to **solve a specific task** (as efficiently as possible)
- Dialogues follow a **clearly designed structure** (flow) that is developed for a particular task in a **closed domain**
- Well-defined measure of performance that is explicitly related to **task completion**



Social chat

- Aim to **carry on an extended conversation** ("chit-chat") with the goal of mimicking human-human interactions
- Developed for **unstructured, open domain** conversations
- Objective is to be **human-like**, i.e., able to talk about different topics (breadth and depth) in an engaging and coherent manner

Interactive QA

- Aim to **provide concise, direct answers** to user queries
- Dialogues are **unstructured**, but commonly follow a question-answer pattern; mostly **open domain** (dictated by the underlying data)
- Evaluated with respect to the **correctness of answers** (on the turn level)



Conversational information access

- A subset of conversational AI systems that specifically aim at a task-oriented sequence of exchanges
- Supports multiple user goals, including search, recommendation and exploratory information gathering
- Requires multi-step interactions over possibly multiple modalities
- Combine elements from both task-oriented and interactive QA systems
- Consider both long-term and short-term information about the user when solving information seeking tasks

Question

Hasn't search always been conversational?

Search as a conversation

Where will ECIR take place in 2022?



G

The ECIR conference is the premium European forum for presentation of new research results in the area of Information Retrieval (IR). The 2022 conference will take place in **Stavanger**. Dec 21, 2020

<https://www.uis.no/research/uis-will-host-the-ecir-conf...> ::

[UiS will host the ECIR conference 2022 | University of Stavanger](#)

About featured snippets · Feedback

<https://ecir2022.org> ::

[ECIR 2022 – 44th European Conference on Information ...](#)

The 44th edition of ECIR is planned to be held as a physical conference (with support for ...
Abstracts are due on September 30, papers on October 7 (AoE).

[Important Dates](#) · [Calls](#) · [Call for Full Papers](#) · [Call for Demonstrations](#)

Search as a conversation

places to visit in Stavanger

G

Top sights in Stavanger

Most visited Kid-friendly Outdoors Museums

Pulpit Rock
4.9 ★★★★ (1,000)
Landmark outcropping with fjord views

Norwegian Petroleum Museum
4.4 ★★★★ (2,454)
Petroleum museum in landmark building

Svend i fjell
4.3 ★★★★ (3,655)
Monument with 3 vast seascapes by a fjord

More things to do

Stavanger

City in Norway

Stavanger is a city in southwestern Norway. In the center of town, Stavanger Cathedral dates back to the city's 12th-century founding. Stavanger Museum chronicles the city's history and displays preserved wildlife. The Norwegian Petroleum Museum illuminates the oil industry with submersibles, a large drill bit and an escape chute. The shopping street Øvre Holmegate is known for its colorful houses. — Google

Area: 71.35 km²

Metropolitan area: 2,500 km²

Urban area: 77.36 km²

Weather: -1°C, Wind E at 2 m/s, 70% Humidity

Population: 130,754 (2014) 1000 House

Mayor: Karl Helge Nordin

Local time: Friday 17:02

Plan a trip

Things to do

3-star hotel averaging NOK 1,147

Upcoming Events

Colleges and Universities: University of Stavanger, VID Specialized University

People also search for

View 15+ more

[www.planetware.com - stavanger-norgl-stav](#)

15 Top-Rated Tourist Attractions in Stavanger | PlanetWare

Nov 17, 2020 — Pulpit Rock (Preikestolen); Pulpit Rock (Preikestolen); Norwegian Petroleum Museum; Norwegian Petroleum Museum; Norwegian Canning Museum; Norwegian Canning Museum | Chris Welsh / photo modified; Stavanger Cathedral; The Vårtberg Tower (Vårtbergstårnet); Stavanger Museum; Gamle Stavanger (Old Stavanger) Lysefjord.

[falsuitcase.com - stavanger-best-things-to-do](#)

Best Things To Do in Stavanger Norway (These 8 Are a Must)

Nov 13, 2020 — This is not a complete list of Stavanger attractions. This beautiful region has so much more to offer than that, but if you want to see the main...

People also ask

Is Stavanger worth visiting?

What is Stavanger famous for?

What is there to do in Stavanger today?

How far is Oslo from Stavanger?

Feedback

[www.tripadvisor.com - Stavanger](#)

Things to Do in Stavanger, Norway - Stavanger Attractions

Top Attractions in Stavanger | TripAdvisor 1,107 reviews 100 Experiences 2 Old Stavanger 2,345 reviews 3 For og Fjære 747 reviews 4 Norway ...

14 / 69

Search as a conversation

places to visit in Stavanger 

G

Top sights in Stavanger

More visited | Eat & drink | Outdoors | Museums

Publix Rock  Landmark overlooking with Stolt sea
4.5 stars (4,000 reviews)

Norwegian Petroleum Museum  Petroleum museum in Søgne
4.5 stars (3,400 reviews)

Sverd i fjell  Monument with 2 cast swords by a fjord
4.5 stars (5,000 reviews)

More things to do

[www.planetware.com - stavanger-in-norway/](#) *
15 Top-Rated Tourist Attractions in Stavanger | PlanetWare
Nov 17, 2020 — Publix Rock (Prostekjet) | Publix Rock (Prostekjet); Norwegian Petroleum Museum; Norwegian Petroleum Museum; Norwegian Canning Museum | Chris Heaton / photo modified; Stavanger Cathedral; The Tower (Torget) (Stavanger); Stavanger Museum; Giske Stavanger (Stavanger) Lysefjord

[www.visitnorway.com - visitnorway.visitnorway.com/](#) *
Best Things To Do in Stavanger Norway (These 8 Are a Must!)
Nov 13, 2020 — This is not a complete list of Stavanger attractions. This beautiful region has so much more to offer than that, but if you want to see the main

People also ask

Is Stavanger worth visiting?
What is Stavanger famous for?
What is there to do in Stavanger today?
How far is Oslo from Stavanger?

Feedback

[www.tripadvisor.com - Stavanger *](#)
Things to Do in Stavanger, Norway - Stavanger Attractions
Top Attractions in Stavanger: 1. Lysefjord (1,035 reviews) See 10 Experiences · 2. Old Stavanger (2,045 reviews) · 3. Flørløg Fjær (748 reviews) · 4. Norskegata ...
What are the top attractions to visit in Stavanger?
What are the best outdoor activities in Stavanger?
What are the most popular things to do in Stavanger with kids?

View 15+ more

Local time: Friday 17:02

Plan a trip

3-star hotel averaging NOK 1,147
Upcoming Events

Colleges and Universities: University of Stavanger, VSU
Sørlandet University

People also search for

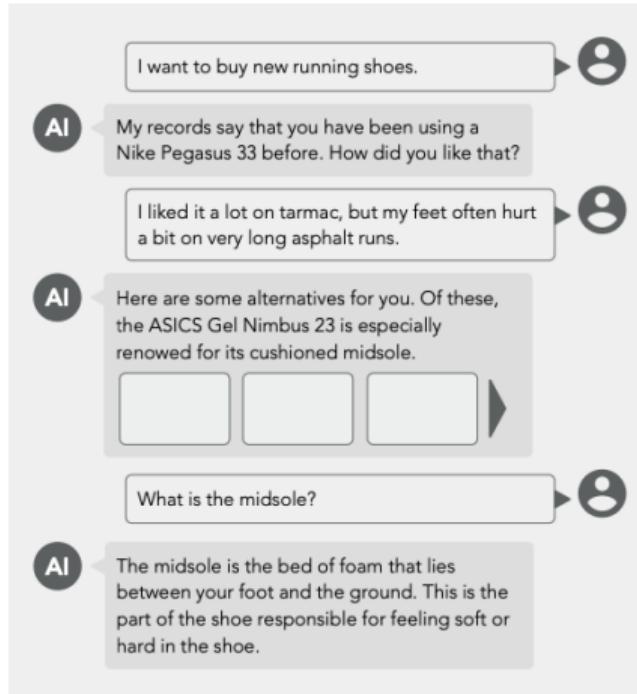
By Stavanger you meant the city in Norway, right?

Do you want to know more about Stavanger?

Would you be interested in other cities in Norway as well?

Conversational, in the “modern” sense

- Degree of personalization and long-term user state
- Support for complex (multi-step) tasks
- Answer generation vs. answer retrieval
- Dialogue setting where a screen or keyboard may not be present
- Mixed initiative



Architecture

Architecture

- Chatbots
- Task-oriented Dialogue Systems

Chatbots

- Goal is to mimic the unstructured conversations characteristic of informal human-human interaction
- The simplest kinds of dialogue systems
- Often have an entertainment value
- Can carry on extended conversation
- Several types
 - Rule-based (e.g., ELIZA, PARRY)
 - Corpus-based (e.g., Xiaoice)
 - Hybrid

Rule-based chatbot - ELIZA

- Most important chatbot dialogue system in the history of the field
- Shows we do not need to know anything about the real world to have a conversation
- Designed to simulate a Rogerian psychologist
 - Methods involve drawing the patient out by reflecting patient's statements back at them
- Most chatbots trying to pass the Turing test choose a domain with similar properties

Rule-based chatbot - ELIZA

Men are all alike.

IN WHAT WAY

They're always bugging us about something or other.

CAN YOU THINK OF A SPECIFIC EXAMPLE

Well my boyfriend made me come here.

YOUR BOYFRIEND MADE YOU COME HERE

He says I'm depressed much of the time.

I AM SORRY TO HEAR YOU ARE DEPRESSED

..

WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU?

My father

YOUR FATHER

You are like my father in some ways.

WHAT RESEMBLANCE DO YOU SEE

You are not very aggressive but I think you don't want me to notice that.

WHAT MAKES YOU THINK I AM NOT AGGRESSIVE

You don't argue with me.

WHY DO YOU THINK I DON'T ARGUE WITH YOU

You are afraid of me.

DOES IT PLEASE YOU TO BELIEVE I'M AFRAID OF YOU

My father is afraid of everybody.

WHAT ELSE COMES TO MIND WHEN YOU THINK OF YOUR FATHER

Bullies.

DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR
BOYFRIEND MADE YOU COME HERE

Corpus-based chatbots

- Mine conversations of human-human conversations
- Data-intensive, requiring hundreds of millions or even billions of words for training (e.g., excerpts of movie dialogues, use of crowdworkers to have conversations)
- Due to lack of large conversational corpora, many systems pre-train first on pseudo-conversations generated from Twitter, Reddit and other social media
- After deployment, chatbots can collect usage data
- Response can be done by
 - **retrieval** - using IR techniques to find a response in a corpus
 - **generation** - using a language model or encoder-decoder to generate the response

Response by retrieval

- The goal is to find a response $r \in C$ such that r is most similar to the query q (user's turn)
- Compute tf-idf models for C and q , choosing the r that has the highest tf-idf cosine with q

$$\text{response}(q, C) = \operatorname{argmax}_{r \in C} \frac{q \cdot r}{|q||r|}$$

- If the collection contains conversational turns, we find the most similar turn to the query and return the turn following it

Response by retrieval - cont'd

- Alternatively, we can use neural IR (e.g., Transformer-based encoder model)
- In the spirit of dense retrieval we train two separate encoders, one for the query and one for the responses and use the dot product as similarity

$$h_q = BERT_Q(q)[CLS]$$

$$h_r = BERT_R(r)[CLS]$$

$$\text{response}(q, C) = \operatorname{argmax}_{r \in C} h_q \cdot h_r$$

Response by generation

- It was proposed early on to think of response generation as a kind of translation
 - "Translate" an input query q into an response r
- Use of encoder-decoder model was natural
- The task is to generate each token r_t of the response by conditioning on the encoding of the entire query q and the response so far $r_1 \dots r_{t-1}$

$$r_t = \operatorname{argmax}_{w \in V} P(w | q, r_1 \dots r_{t-1})$$

- It is common practice to include a longer context, forming the query not just from the user's turn but from the entire conversation so far

Response by generation - cont'd

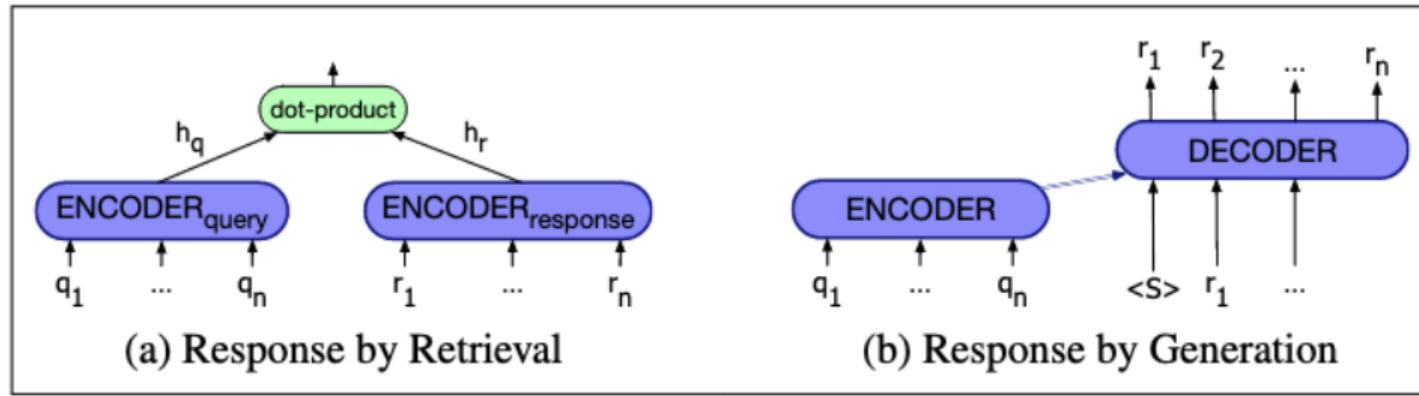


Figure: Two architectures for generating responses for a neural chatbot. In response by retrieval (a) we choose a response by finding the turn in the corpus whose encoding has the highest dot-product with the user's turn. In response by generation (b) we use an encoder-decoder to generate the response.

Response by retrieving and refining knowledge

- Responses are a combination of retrieval and generation
 - First use IR to retrieve potentially useful passages from Wikipedia (retrieval)
 - Create multiple candidates by concatenating each retrieved Wikipedia sentence to the dialogue context with a separator token
 - Each candidate can be given as the encoder context to the encoder-decoder model (generation)

Architecture

- Chatbots
- Task-oriented Dialogue Systems

Task-oriented Dialogue Systems

- Task-oriented dialogue agents use conversation with users to help complete task
- They can answer questions, give directions, control appliances, find restaurants, or make calls
- Two most common architectures are
 - Simple Frame-based Dialogue Systems
 - The Dialogue-State Architecture

Simple Frame-based Dialogue Systems

- Representative of the architecture is a very simple GUS system for travel planning introduced in 1977
- Even though its old, it underlies most modern commercial digital assistants
- A frame is a kind of knowledge frame structure representing the kinds of intentions the system can extract from user sentences
- A frame consists of a collection of **slots**, each of which can take a set of possible **values**
 - The set of slots in a frame specifies what the system needs to know
 - The filler of each slot is constrained to values of a particular semantic type

Simple Frame-based Dialogue Systems

GUS: Hello. My name is GUS. I can help you plan a simple trip by air. Where do you want to go ?

Client: I want to go to San Diego on May 28

GUS: What time do you want to leave?

Client: I must be in San Diego before 10 am

GUS: Would you like P.S.A. flight 102 that arrives at 9.15 am ?

Client: What is the next flight ?

GUS: Air California flight 310 that leaves at 8.30am and arrives at 10.00am

Client: I'll take the first one

GUS: In what name should I make a reservation, please?

Client: Dan Bobrow

GUS: I have confirmed the following flight: P.S.A. flight 102 on Wednesday May 28 from San Jose to San Diego that leaves at 7.30 am and arrives at 9.15 am.

GUS: What date do you want to return on ?

Client: On Friday in the evening.

GUS: Would you like the flight that leaves at 7.45 pm ?

Client: That's fine.

GUS: I have confirmed the following flight: P.S.A. flight 307 on Friday May 30 from San Diego to San Jose that leaves at 7.45 pm and arrives at 9.30 pm Thank you for calling. Goodbye

Figure: The travel domain: A transcript of an actual dialogue with the GUS system.

Simple Frame-based Dialogue Systems

- Types can have hierarchical structure (e.g., *date* in GUS is a frame with slots like *month*, *year*, *day*, *weekday*)

Slot	Type	Question Template
ORIGIN CITY	city	“From what city are you leaving?”
DESTINATION CITY	city	“Where are you going?”
DEPARTURE TIME	time	“When would you like to leave?”
DEPARTURE DATE	date	“What day would you like to leave?”
ARRIVAL TIME	time	“When do you want to arrive?”
ARRIVAL DATE	date	“What day would you like to arrive?”

Figure: A frame in a frame-based dialogue system, showing the type of each slot and a question used to fill the slot.

Determining Domain, Intent, and Slot fillers

- In each turn of the conversation the system needs to extract 3 things from a user utterance
- The first task is domain classification
 - Unnecessary for single-domain systems
- The second is user intent determination
 - What general task or goal is the user trying to accomplish?
- The third is slot filling
 - Extract the particular slots and fillers that the user intends the system to understand from their utterance with respect to their intent
 - Method used in the original GUS system, and still quite common in industrial applications, is to use handwritten rules

Control structure for frame-based dialogue

- The system's goal is twofold
 - Fill the slots in the frame with the fillers the user intends
 - Perform the relevant action for the user (e.g., answering a question, or booking a flight)
- First goal is achieved with the system asking questions of the user (using pre-specified question templates)
- Once the system has enough information it performs the necessary action (e.g., querying a database of flights)

Other components of frame-based dialogue

- The ASR (automatic speech recognition) component takes audio input outputs a transcribed string of words
- The NLU (natural language generation) module of any dialogue system produces the utterances that the system says to the user
 - Generally, template-based generation (can be fixed or include variables)

The Dialogue-State Architecture

- More sophisticated version of the frame-based architecture
- The four main components include
 - Natural language understanding (NLU)
 - Responsible for determining domain, intent, and slot fillers
 - Generally using machine learning rather than rules
 - Dialogue State Tracking
 - Maintains the current state of the dialogue including the most recent **dialogue act** and the entire set of slot-filler constraints the user has expressed so far
 - Dialogue Policy
 - decides what the system should do or say next
 - More advanced than in GUS where the policy was to ask questions until the frame is full and report back the results
 - Natural language generation (NLG)
 - Rather than using templates, condition on the exact context to produce turns that seem much more natural

The Dialogue-State Architecture

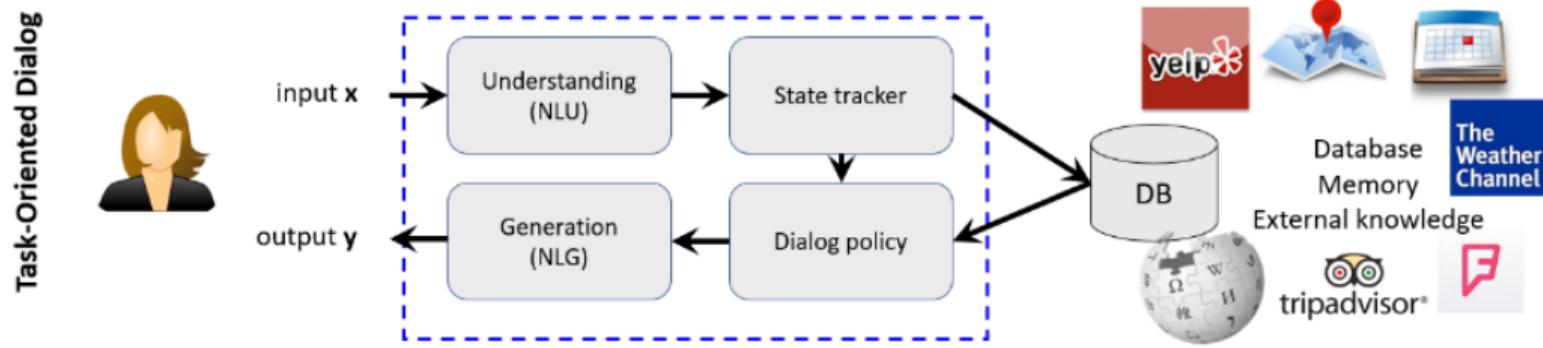


Figure: Architecture of a dialogue-state system for task-oriented dialogue³

³J. Gao, M. Galley, and L. Li, "Neural Approaches to Conversational AI." arXiv, Sep. 10, 2019.

Dialogue Acts

- Dialogue acts represent the interactive function of the turn or sentence
- Different types of dialogue systems require labeling different kinds of acts

Tag	Sys	User	Description
HELLO($a = x, b = y, \dots$)	✓	✓	Open a dialogue and give info $a = x, b = y, \dots$
INFORM($a = x, b = y, \dots$)	✓	✓	Give info $a = x, b = y, \dots$
REQUEST($a, b = x, \dots$)	✓	✓	Request value for a given $b = x, \dots$
REQALTS($a = x, \dots$)	✗	✓	Request alternative with $a = x, \dots$
CONFIRM($a = x, b = y, \dots$)	✓	✓	Explicitly confirm $a = x, b = y, \dots$
CONFREQ($a = x, \dots, d$)	✓	✗	Implicitly confirm $a = x, \dots$ and request value of d
SELECT($a = x, a = y$)	✓	✗	Implicitly confirm $a = x, \dots$ and request value of d
AFFIRM($a = x, b = y, \dots$)	✓	✓	Affirm and give further info $a = x, b = y, \dots$
NEGATE($a = x$)	✗	✓	Negate and give corrected value $a = x$
DENY($a = x$)	✗	✓	Deny that $a = x$
BYE()	✓	✓	Close a dialogue

Figure: Dialogue acts used by the HIS restaurant recommendation system.^a

^aYoung, S. J. et al, 2010. The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management. Computer Speech & Language

Slot Filling

- Use neural model to jointly associate each sentence with the correct set of slots, domain, and intent
- Classify each token to a predefined BIO label (beginning (B), inside (I), and outside (O))
- There are $2n + 1$ tags, where n is the number of slots

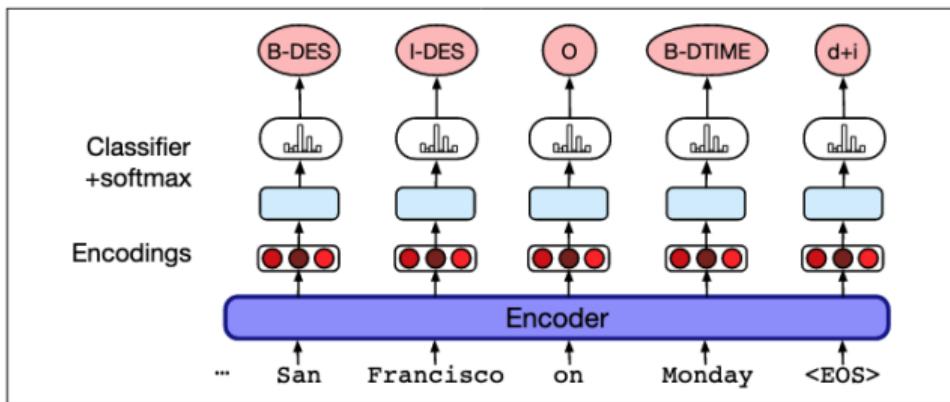


Figure: A simple architecture for slot filling

Dialogue State Tracking

- Job is to determine both the current state of the frame and the user's most recent dialogue act
- Contains the value of the frame since the beginning of the conversation
- Simple solution
 - take the output of a slot-filling sequence-model after each sentence (NLU)
- More complex model
 - Train a classifier for each slot to decide whether its value is being changed in the current sentence or should be carried over from the previous sentence

Dialogue Policy

- Decide what action the system should take next, i.e., what dialogue act to generate
- At turn i in the conversation predict which action A_i to take, based on the entire dialogue state. Given the sequence of dialogue acts from the system (A) and from the user (U) we want to compute

$$\hat{A}_i = \operatorname{argmax}_{A_i \in A} P(A_i | (A_1, U_1, \dots, A_{i-1}, U_{i-1}))$$

- Or alternatively, collapsing across the many different conversational paths that could lead to the same set of filled slots

$$\hat{A}_i = \operatorname{argmax}_{A_i \in A} P(A_i | \text{Frame}_{i-1}, A_{i-1}, U_{i-1})$$

- Estimate the probabilities using neural models

Policy Example: Confirmation and Rejection

- Modern dialogue systems often make mistakes
- We can use confirmation and rejection dialogue acts to make sure we have understood the user correctly
- Confirmation is used to confirm the understanding with the user
 - can be **explicit** or **implicit**
 - Explicit and implicit confirmation have complementary strengths
- Rejection is when a system expresses the lack of understanding (e.g., *I'm sorry, I didn't understand that*)

Explicit Understanding

- A system asks the user a direct question to confirm the system's understanding
- Makes it easier for users to correct the system's misrecognitions
- It is awkward and increases the length of the conversation

S: Which city do you want to leave from?

U: Baltimore.

S: **Do you want to leave from Baltimore?**

U: Yes.

U: I'd like to fly from Denver Colorado to New York City on September twenty first in the morning on United Airlines

S: **Let's see then. I have you going from Denver Colorado to New York on September twenty first. Is that correct?**

U: Yes

Implicit Understanding

- System can demonstrate its understanding as a grounding strategy repeating back the system's understanding as part of asking the next question
- It is much more conversationally natural

U: I want to travel to Berlin

S: **When do you want to travel to Berlin?**

U2: Hi I'd like to fly to Seattle Tuesday Morning

A3: **Traveling to Seattle on Tuesday, August eleventh in the morning.**
Your full name?

Natural language generation (NLG)

- Often modeled in two stages
 - content planning (what to say)
 - sentence realization (how to say it)
- Assume that content planning was done by dialogue policy (By selecting a dialogue act and a set of slots and values)
- Training data is hard to come by, increase generality by **delexicalization**

```
recommend(restaurant name= Au Midi, neighborhood = midtown,  
cuisine = french)
```

- 1 Au Midi is in Midtown and serves French food.
- 2 There is a French restaurant in Midtown called Au Midi.

```
recommend(restaurant name= Loch Fyne, neighborhood = city  
centre, cuisine = seafood)
```

- 3 Loch Fyne is in the City Center and serves seafood food.
- 4 There is a seafood restaurant in the City Centre called Loch Fyne.

Figure: Two examples of inputs to the sentence realization phase of NLG

Natural language generation (NLG) - Delexicalization

- The process of replacing specific words in the training set that represent slot values with a generic placeholder token representing the slot
- Mapping from frames to delexicalized sentences is generally done by encoder decoder model
- The encoder reads all the input slot/value representations, and the decoder outputs the following delexicalized English sentence
- Use the input frame from the content planner to relexicalize

```
recommend(restaurant name= Au Midi, neighborhood = midtown,  
cuisine = french
```

- 1 `restaurant.name` is in `neighborhood` and serves `cuisine` food.
- 2 There is a `cuisine` restaurant in `neighborhood` called `restaurant.name`.

Natural language generation (NLG) - Generating Clarification Questions

- Example of NLG algorithm specific to a particular dialogue act
- Instead of using dialogue act *reject*("Please repeat") it is better to use targeted clarification questions

User: What do you have going to UNKNOWN_WORD on the 5th?

System: Going where on the 5th?

- Targeted clarification questions can be created by rules or by building classifiers to guess which slots might have been misrecognized in the sentence

Examples

Conversational Recommender Systems

Definition

A recommendation system that can elicit the dynamic preferences of users and take actions based on their current needs through real-time multturn interactions.

Main challenges in CRS

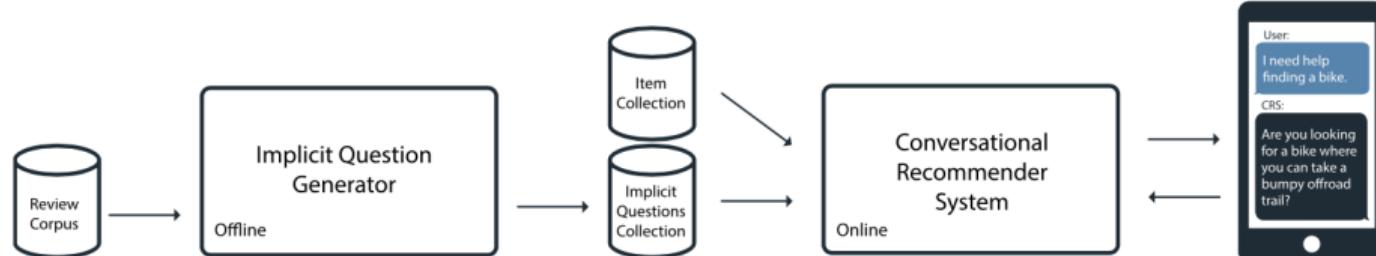
- Question-based user preference elicitation
 - What to ask?
 - How to adjust the recommendations based on user response?
- Multi-turn Conversational Recommendation Strategies
 - When to ask questions and when to make recommendations?
- Natural Language Understanding and Generation
 - Communicating like a human continues to be one of the hardest challenges
- Trade-offs between Exploration and Exploitation
 - Users only have limited time and energy to interact with the system, a failed exploration will waste time and lose the opportunity to make accurate recommendations.
- Evaluation and User Simulation
 - Emphasis on the user experience during dynamic interactions makes online evaluation preferred but very expensive

Examples

- Soliciting User Preferences in Conversational Recommender Systems via Usage-related Questions
- IAI MovieBot
- DAGFiNN

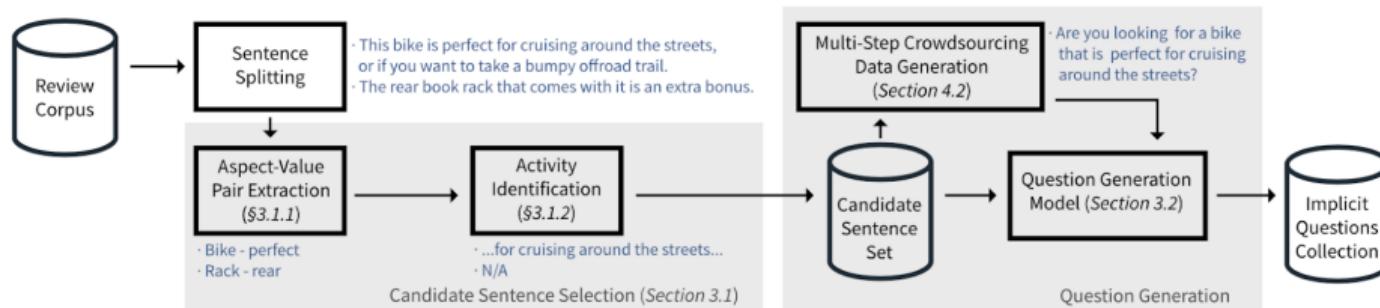
Usage-related Questions - Motivation

- Examples of traditional preference elicitation:
 - How thick should the tires be?
 - How many gears should the bike have?
- Usage based preference elicitation:
 - Will you be using this bike mostly for commuting?



Usage-related Questions - Approach

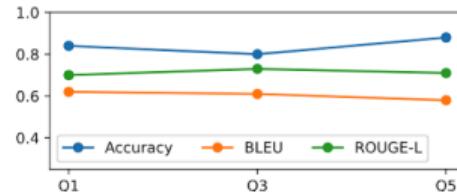
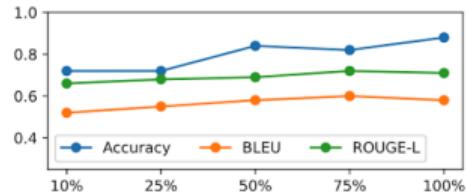
- Extract usage related sentences from a review corpus.
- Use crowdsourcing to generate labeled data.
- Train a text-to-text model (T5) to generate questions.



Usage-related Questions - Results

Model	#Parameters	Accuracy	BLEU-4	ROUGE-L
T5-small	60.5 M	0.76	0.56	0.69
T5-base	222 M	0.84	0.53	0.66
T5-large	737 M	0.88	0.58	0.71

Fig. 3. Model performance (T5-large) with sentence-based (Left) or question-based (Right) training data reduction.



Examples

- Soliciting User Preferences in Conversational Recommender Systems via Usage-related Questions
- IAI MovieBot
- DAGFiNN

- Open-source conversational movie recommender system
- Models users' preferences dynamically
- Supports multi-turn recommendations
- Typical dialogue-state architecture
- Multi-modal chat interface
- Student project

Discussion

Question

If you had to design a movie recommender system, what intents would you choose?

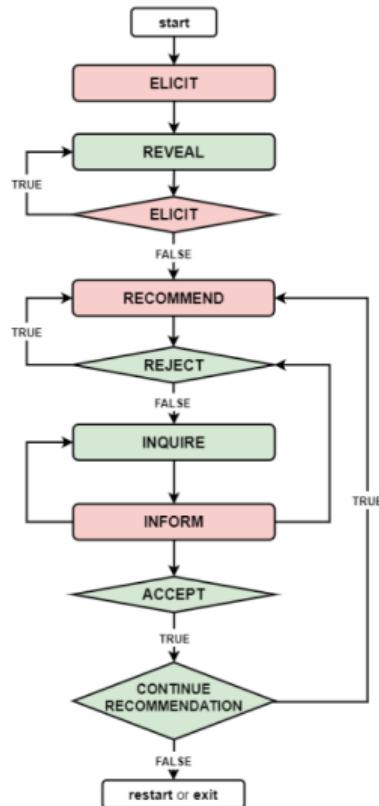
IAI MovieBot - Overview of intents

	Intent	Description
User Revealment [7] / Query Formulation [9] / Query Reformulation [9]		
User	Reveal	The user wants to reveal a preference. <i>“Do you have any sports movies?”</i>
	Remove preference	The user wants to remove any previously stated preference. <i>“I don’t want to see any sports movies anymore.”</i>
Agent	Elicit	Ask the user to describe their preferences. <i>“Which genres do you prefer?”</i>
System Revealment [7] / Result Exploration [9]		
User	Inquire	Once the agent has recommended an item, the user can ask further details about it. <i>“Please tell me more about this movie.”</i>
	Accept/Reject	The user can decide if they like the recommendation or not.
		The user accepts (likes) the recommendation. This will determine the success of the system as being able to find a recommendation the user liked. <i>“I like this recommendation.”</i>
	Reject	The user either has already seen/consumed the recommended item or does not like it. <i>“I have already seen this one.”</i>
	Continue recommendation	If the user likes a recommendation, they can either restart, quit or continue the process to get a similar recommendation. <i>“I would like a similar recommendation.”</i>

IAI MovieBot - Overview of intents

Agent	Reveal Too many results	Reveal the results or the number of matching results to the user. The number of items matching the user's preferences is larger than a maximum limit. This will be followed by an <i>elicit</i> intent. <i>"There are almost 1100 action movies."</i>
	Recommend	Based on the user's preferences, make a recommendation. <i>"I would like to recommend a fairy tale film, named Shrek."</i>
	No Results	The database does not contain any items matching the user's preferences. <i>"Sorry. I couldn't find any romantic Korean movies."</i>
	Inform	If the user <i>inquires</i> about the recommended item, the agent provides the relevant information. <i>"The director of this movie is XYZ."</i>
	Miscellaneous Intents	
User	Hi	When the user initiates the conversation, they start with a formal <i>hi/hello</i> or <i>reveal</i> preferences.
	Acknowledge	Acknowledge the agent's question where required.
	Deny	Negate the agent's question where required.
	Bye	End the conversation by sending a bye message or an exit command.
Agent	Welcome	Start the conversation by giving a short introduction.
	Acknowledge	Acknowledge the user's query where required.
	Can't Help	The agent does not understand the user's query or is not able to respond properly based on the current dialogue state.
	Bye	End the conversation.

IAI MovieBot - Dialog Flow



Examples

- Soliciting User Preferences in Conversational Recommender Systems via Usage-related Questions
- IAI MovieBot
- DAGFiNN

DAGFiNN - Motivation

- Create an open-source, multi-domain, multi-modal conversational recommender system
- Provide a personalized and engaging user experience
- Allow users to ask a broad range of questions that naturally arise before and during the conference.
 - There are currently only a few open-source CRS prototypes available
 - The vast majority of research uses text as the sole modality and targets a single domain

DAGFiNN - Conversational Conference Assistant



Figure: Furhat robot.

The image shows a screenshot of the DAGFiNN conversational interface. On the left, there is a white Furhat robot head. To its right is a rectangular screen displaying a conversation and a map. The conversation starts with the DAGFiNN logo and the text "I'm looking for somewhere to eat". The user responds with "What kind of food do you like?". The DAGFiNN replies with "I like burgers but I want one without meat". Finally, it suggests a restaurant: "I found this top-rated restaurant which offers burgers and vegetarian food." To the right of the conversation is a map of Stavanger, Norway, showing the location of McDonald's (\$). The address is listed as "McDonald's (\$)" and "Saregata 7, 4006 Stavanger". A red line on the map indicates a route from the user's current location to the restaurant. Below the map is a QR code labeled "Learn more:".

Figure: External screen with a rich response example.

DAGFiNN - Conversational Conference Assistant

- Provide recommendations across multiple domains, have basic conversational capabilities, and personalization integrated into a single system.
 - Currently, recommenders are built for point-of-interest (POI) and conference content
- Support multiple user interfaces (UI)
 - A Web chat and a Furhat robot with an optional external screen
 - Use of rich responses customized to each user interface to effectively convey information.
- Utilize multiple input modalities
 - Furhat is a social robot with a microphone, speaker, and camera, which makes it possible to recognize users that interacted with the robot previously.

DAGFiNN - Architecture

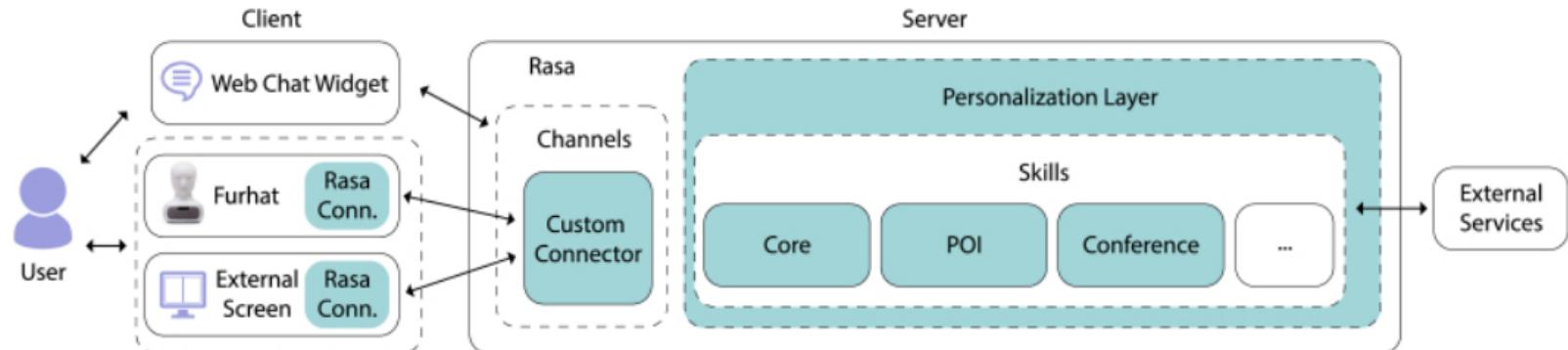


Figure: DAGFiNN architecture. Left: Clients with different interfaces; Web chat and Furhat robot with external screen. Right: Server.

DAGFiNN - Example questions

- What is there to see in Stavanger?
- Can you recommend me some bars?
- Do you know of any good Italian restaurants?
- Is there somewhere cheaper?
- Where can I go running?
- Where can I go hiking?
- How do I get there?
- How do I get to the airport?

Exercise

DAGFiNN Demo

Summary

- Conversational AI
 - Conversational information access
- Architecture
 - Chatbots
 - Rule-based chatbot
 - Corpus-based chatbots
 - Task-oriented Dialogue Systems
 - Simple Frame-based Dialogue Systems
 - The Dialogue-State Architecture
 - Evaluating Dialogue Systems
 - Evaluating Chatbots
 - Evaluating Task-Based Dialogue
 - Dialogue System Design
- Examples
 - Soliciting User Preferences in Conversational Recommender Systems via Usage-related Questions
 - IAI MovieBot
 - DAGFiNN

Reading

- *Chatbots and Dialogue Systems*, Daniel Jurafsky and James H. Martin ⁴

⁴<https://web.stanford.edu/~jurafsky/slp3/>

Advanced Evaluation

[DAT640] Information Retrieval and Text Mining

Krisztian Balog & Nolwenn Bernard

University of Stavanger

October 24, 2022



CC BY 4.0

In this module

1. Evaluating Conversational Search and Recommender Systems
2. Data collection exercise

Recap

q001	doc00320	0
q001	doc01321	4
q001	doc17754	2
q001	doc44510	3
q001	doc90974	0
...	...	
q211	doc55204	2
q211	doc09077	0
q211	doc13201	1
q211	doc89210	1
q211	doc67828	4
...	...	

f2015.qrels

Question

What are the assumptions made in offline test collections for ranking evaluation?

Assumptions in offline retrieval evaluation

- Queries are a representative sample of actual user needs
 - Head vs. torso vs. tail queries
- User judgments are representative of an average user of the actual user population
- There is a finite set of items to choose from
 - Items can only be retrieved, not generated
- All relevant answers have been identified
 - This is achieved by pooling

From relevance assessments to evaluation measures

- Relevance assessments are for (query, item) pairs
- Evaluation measures express how well the system ranks items for a given query w.r.t. the ground truth
- Evaluation measures have an (implicit or explicit) underlying user model
 - The user model describes how users interact with the ranked list

Question

Assume that your boss tells you that the search engine of the site you're in charge of should be replaced. They show you a few queries where the search doesn't work as expected and suggest to switch to a different search service provider who handle those queries much better. How would you react to this?

Evaluating Conversational Search and Recommender Systems

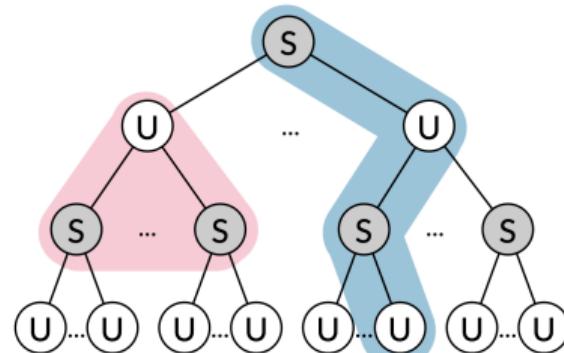
Types of evaluation

- Component-level vs. end-to-end
- Automatic vs. human vs. live evaluation

Challenges

Turn-based (offline) evaluation

- ✓ Possible to create a reusable test collection for a specific subtask
- ✗ Limited to a single turn, does not measure overall user satisfaction



End-to-end (human) evaluation

- ✓ Possible to annotate entire conversations
- ✗ Expensive, time-consuming, does not scale, does not yield a reusable test collection

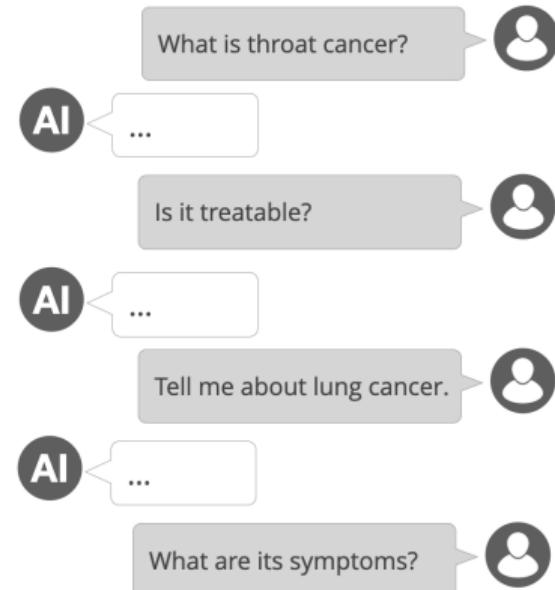
Evaluating Conversational Search and Recommender Systems

- Offline Evaluation
- Evaluation Using User Simulation

Conversational search

Task: given a natural language query, return a passage from a collection as an answer.

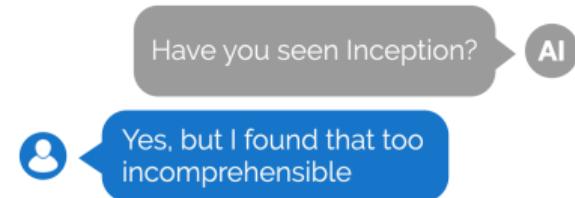
- TREC Conversational Assistance track
- Evaluation is performed with respect to answer relevance



Critiquing in conversational recommendations

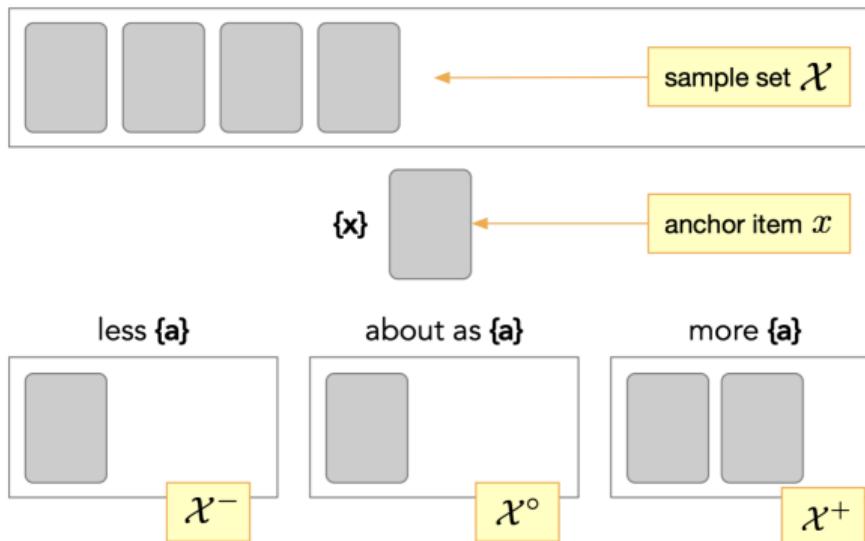
Task: given an item recommendation, incorporate natural language feedback (critiques) on items, which comes in the form of *soft attributes*

- A *soft attribute* is a property of an item that is not a verifiable fact that can be universally agreed upon, and where it is meaningful to compare two items and say that one item has more of the attribute than another
- Example soft attributes for movies: artsy, light-hearted, intense, predictable, violent ...



Purpose-built annotation interface

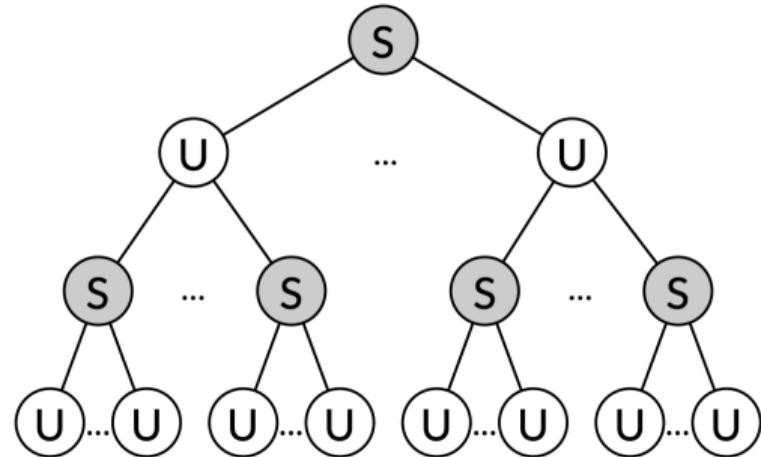
Drag and drop these movies into three categories based on how $\{a\}$ they are compared to $\{x\}$.



Evaluating Conversational Search and Recommender Systems

- Offline Evaluation
- Evaluation Using User Simulation

Challenges



Objectives

The user simulator should produce responses that a real user would give in a certain dialog situation.

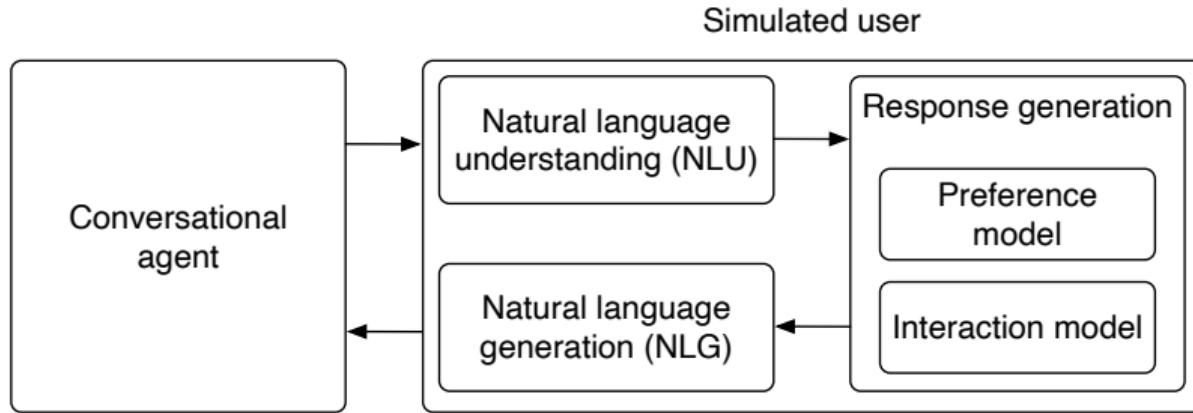
- Enable automatic assessment of conversational agents
- Make no assumptions about the inner workings of conversational agents

Formally

- For a given system S and user population U , the goal of user simulation U^* is to predict the performance of S when used by U , denoted as $M(S, U)$
- For two systems S_1 and S_2 , U^* should be such that

$$M(S_1, U) < M(S_2, U) \Rightarrow M(S_1, U^*) < M(S_2, U^*)$$

User simulator architecture

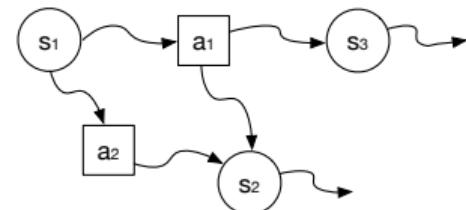


- **NLU:** Translate an agent utterance into a structured format
- **Response generation:** Determine the next user action based on the understanding of the agent's utterance
- **NLG:** Turn a structured response representation into natural language

Modeling simulated users

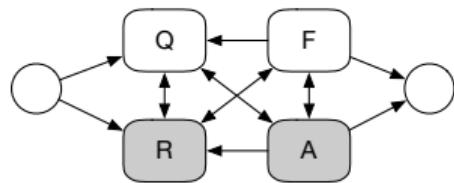
- Model dialogue as a Markov Decision Process (MDP)
- Every MDP is formally described by a finite state space \mathcal{S} , a finite action set \mathcal{A} , and transition probabilities P
- At each time step (dialogue turn) t , the dialogue manager is in a particular state $s_t \in \mathcal{S}$
- By executing action $a_t \in \mathcal{A}$, it transitions into the next state s_{t+1} according to the transition probability $P(s_{t+1}|s_t, a_t)$
- The Markov property ensures that the state at time $t + 1$ depends only on the state and action at time t :

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) = P(s_{t+1}|s_t, a_t) .$$

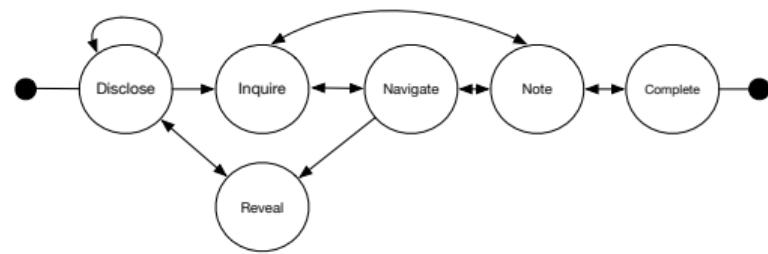


Interaction model

- Defines the space of agent and user actions and the possible transitions between them
- Examples



Simplest model with two user actions (query, feedback) and agent actions (request, answer)



User actions specific to the task of conversational item recommendation

Instantiating simulators

- Specifying the space of user and agent actions
- Training the NLU for the particular agents
- Setting up NLG (template-based/machine-learned) for the particular scenario
- It has to be grounded in actual user data!

Data collection exercise

Motivations

- Conversations mix *conversational goals*: question answering, search, and recommendation
- Lack of datasets with conversations mixing goals
- Provide resources to support development of new conversational agent handling multiple goals

Virtual shopping assistant

- Objective: collect human-human conversations mixing conversational goals in the retail domain
- Roles
 - Client: you are looking to buy an item and need the help of an assistant
 - Retail assistant: provide help to a client in order to satisfy their needs

Today's session

- Two groups: (1) clients and (2) retail assistants
- Topics of discussion available
 - Books
 - Sports and outdoors
 - CDs and vinyl
 - Grocery and gourmet food
 - Office products
 - Cell phones and accessories
 - Musical instruments
- [Link](#) to the chat platform
- Aim: complete 2 conversations

Data collection session

- We plan to perform a large scale data collection session
- [Link](#) to the registration form

Summary

- Thinking carefully about assumptions behind evaluation
- Evaluating conversational search and recommender systems
 - Types of evaluation (turn-based vs. end-to-end, automatic vs. human)
 - Advantages and limitations of offline evaluation
 - Examples of automatic turn-based evaluation
 - User simulation for automatic end-to-end evaluation

Reading

- Zhang and Balog. **Evaluating Conversational Recommender Systems via User Simulation.** In: *26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '20)*
 - <https://arxiv.org/pdf/2006.08732>
- Balog et al. **On Interpretation and Measurement of Soft Attributes for Recommendation.** In: *44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21)*
 - <https://arxiv.org/pdf/2105.09179>

Conversational Search Systems

[DAT640] Information Retrieval and Text Mining

Weronika Lajewska
University of Stavanger

31.10.2022



CC BY 4.0

In this module

1. Conversational Search Systems
2. Mixed-initiative

Conversational Search Systems

Recap - conversational information access

- A subset of conversational AI systems that specifically aim at a task-oriented sequence of exchanges
- Supports multiple user goals, including search, recommendation and exploratory information gathering
- Requires multi-step interactions over possibly multiple modalities
- Combine elements from both task-oriented and interactive QA systems
- Consider both long-term and short-term information about the user when solving information seeking tasks

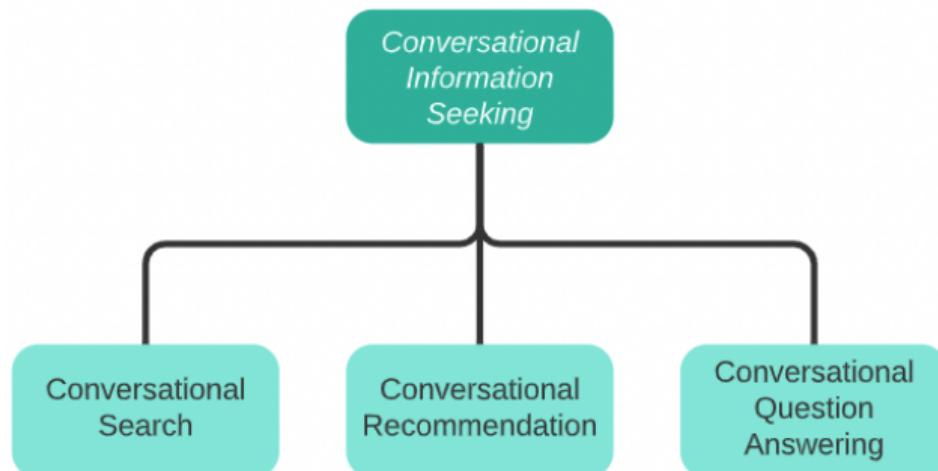
Recap - conversational information access

- Chatbot - system that mimics the unstructured conversations characteristic of informal human-human interaction
- Task-oriented dialogue system - uses conversation with users to help complete task. It can answer questions, give directions, control appliances, find restaurants, or make calls.
 - Conversational recommender system - system that can elicit the dynamic preferences of users and take actions based on their current needs through real-time multturn interactions.

Conversational information seeking¹

Definition

A *Conversational Information Seeking (CIS)* system is a system that satisfies the information needs of one or more users by engaging in information seeking conversations



¹Conversational information seeking <https://arxiv.org/pdf/2201.08808.pdf>

Conversational Search System

Definition

A *conversational search system (CSS)* is a system for retrieving information that permits a mixed-initiative back and forth between a user and agent, where the agent's actions are chosen in response to a model of current user needs within the current conversation, using both short- and long-term knowledge of the user. The responses are expected to be concise, fluent, stateful, mixed-initiative, context-aware, and personalized.

CSS - properties

- User revealment - the system helps the user express (potentially discover) their true information need, and possibly also long-term preferences
- System revealment - the system reveals to the user its capabilities and corpus, building the user's expectations of what it can and cannot do
- Mixed initiative - the system and user both can take initiative as appropriate
- Memory - the user can reference past statements, which implicitly also remain true unless contradicted
- Set retrieval - the system can work with, manipulate and explain the sets of options/objects which are retrieved given the conversational context

CSS - additional challenges

- Filtering out superfluous, e.g., fillers, pauses, false starts
- Answer aggregation by presenting a summary of the retrieved list of results
- General knowledge about external world
- Recovering from communication breakdowns
- Understanding and reasoning about user limitations, e.g., cognitive abilities and styles, domain knowledge

Question

When is the conversational aspect of the search system needed?

Complex scenarios that require CSS

- Faceted elicitation - searching for an item with rich attributes that can be individually specified, but are much simpler to provide piecewise. As part of the search, the user is identifying aspects that can be used to describe a relevant item.
- Multi-item elicitation - searching for a single item supported by a set of nearby items, which requires estimating the relevance of the whole set of items
- Multi-item faceted elicitation - searching for a set of items directly. Not only must the system estimate the utility of each single item, it must combine the utilities of multiple items to reach an assessment of an entire set.
- Bounding choices - providing the user with precise choices rather than expecting them to come up with particular terms. It simplifies the problem of need elicitation.

Conversational Search Systems

- TREC CAsT
- Query Rewriting
- State of the art CSS

TREC CAsT - main goal

- Conversational Assistance Track (CAsT) is a part of the Text REtrieval Conference (TREC) since 2019
- It aims to advance Conversational Information Seeking research and to create a large-scale reusable test collection for CSSs
- The track addresses conversational search that is about to satisfy a user's information need expressed or formalized through multiple turns in a conversation
- The desired response is not a list of relevant documents, but rather a brief passage of a maximum of 3 sentences in length
- The topics released every year are inspired by sessions in web search and triggered to be more challenging for CSSs

TREC CAsT - differences between editions

- In TREC CAsT'19, user utterances may only refer to the information mentioned in previous user utterances
- Since 2020, utterances may refer to previous responses given by the system as well, which significantly extends the scope of contextual information that the system needs to use to understand a request
- TREC CAsT'21 is characterized by the increased dependence on previous system responses, as well as simple forms of user revealment, reformulation, topic shifts, sequence reference, and explicit feedback introduced in users' utterances
- In year 1 and year 2, the systems were using passage collections, while in year 3 the retrieval corpus from which the system response is chosen is over documents, with passages returned, which is a more realistic setting

TREC CAsT - topic²

Title: Steroid use in US sports

Description: The history of steroid use in US sports.

Turn Conversation Utterances

- 1 What's the history of steroid use in sports in the US?
 - 2 What were Ziegler's improvements?
 - 3 Why are they banned?
 - 4 Are there visible signs?
 - 5 That sounds easy to spot. How do they get away with it?
 - 6 What is the NFL policy?
 - 7 Isn't that speed?
 - 8 What is the difference between the two policies?
 - 9 I heard it even affects card players. Didn't bridge also have a problem?
 - 10 I know what bridge is. I heard there was a drug scandal recently.
 - 11 Does the article have more about it?
-

²TREC CAsT 2021: The Conversational Assistance Track Overview

TREC CAsT - evaluation

- The evaluation of TREC CAsT tasks takes into consideration two dimensions of the ranking evaluation:
 - the ranking depth focused on the earlier positions (3,5) for the conversational scenario
 - the turn depth focused on deeper rounds to capture the ability of the system to understand the context of the whole conversation
- The main evaluation metric is the mean NDCG@3 with all conversation rounds averaged using uniform weights
- Additionally, the Recall@1000, MAP and MRR are calculated to evaluate each system

TREC CAsT - pooling

- The assessment pools are formed using the top ten passages from up to four runs per group
- The systems taken into consideration in the pooling need to be intrinsically different in order to achieve higher diversity
- Exhaustive pools let us assume that whatever is not pooled is not relevant
- Participants are asked to prioritize the submitted runs because not all the systems that appear in the competition are pooled

CAsT systems architecture

- There is an established two-step passage ranking architecture
- The first-pass passage retrieval is usually performed using standard unsupervised IR techniques, which is followed by re-ranking using a neural model
- Additionally, most of the systems are using a query rewriting component, where the original query is de-contextualized using a neural model to be independent of the previous turns

TREC CAsT - system architecture - cont.

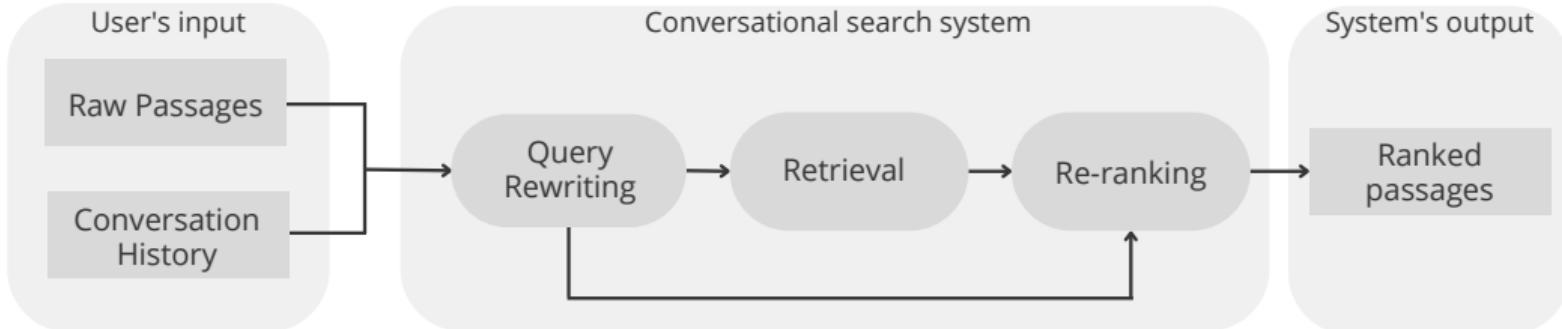


Figure: Cascade two-stage pipeline architecture with query rewriting module.

Conversational Search Systems

- TREC CAsT
- Query Rewriting
- State of the art CSS

Query Rewriting

Definition

The goal of *Conversational Query Rewriting (CQR)* is to produce an informative stand-alone, de-contextualized query for each raw query. Specifically, given a conversational history $H = [q_1, r_1, q_2, r_2, \dots, q_{i-1}, r_{i-1}]$, where r_k is a response provided by the system for k th query q_k and q_i is a current raw query, the task of CQR consists of the filtering out unnecessary information in H and generating the de-contextualized query \hat{q}_i .

Query Rewriting - example

Raw query	Rewrite
I remember there was a global coffee shortage some time ago. Can you tell me more about that?	Can you tell me more about the global coffee shortage?
What caused the drought?	What caused the drought in Brazil last month?
I also heard that other Latin American countries had coffee production issues. Was the disruption widespread?	Was the disruption of coffee production in other Latin American countries widespread?

Table: Examples of query rewrites generated for utterances from TREC CAsT 2021 dataset.

Query Rewriting - feature-based methods

- Unsupervised feature-based methods - expanding an original query by words chosen from conversation history or additional metadata provided by the organizers (e.g., the title of the session)
- Supervised feature-based methods use linguistic features based on dependency parsing, coreference resolution, named entity resolution, or part-of-speech tagging

Query Rewriting - feature-based method example³

- Historical query expansion - term importance estimation through frequency-based signals
- Intuition:
 - Observation #1: A session is centered around a main topic, and the turns in the session dive deeper into several subtopics, each of which only lasts a few turns
 - Observation #2: The degree of ambiguity divides utterances into three categories: The first category includes utterances with clear intents, which can be directly treated as ad-hoc queries. The second category contains those starting a subtopic, and the last category is composed of ambiguous utterances that continue a subtopic

³Multi-Stage Conversational Passage Retrieval:An Approach to Fusing Term Importance Estimation and Neural Query Rewriting

Query Rewriting - feature-based method example⁴ - cont.

Main steps:

- extracting the main topic and subtopic keywords from the utterance
- measuring the ambiguity of the utterance
- expanding queries for the ambiguous utterances with the main topic and subtopic keywords extracted from previous turns

Title: Career choice for Nursing and Physician's Assistant	
Turn (<i>i</i>)	Conversation utterances (<i>u_i</i>)
1	What is a physician's assistant?
2	What are the educational requirements required to become one?
3	What does it cost?
4	What's the average starting salary in the UK?
5	What about in the US?
6	What school subjects are needed to become a registered nurse?
7	What is the PA average salary vs an RN?
8	What the difference between a PA and a nurse practitioner?
9	Do NPs or PAs make more?
10	Is a PA above a NP?
11	What is the fastest way to become a NP?
12	How much longer does it take to become a doctor after being an NP?

⁴Multi-Stage Conversational Passage Retrieval:An Approach to Fusing Term Importance Estimation and Neural Query Rewriting

Query Rewriting - supervised methods

- Supervised neural query rewriting approaches are characterized by utilizing large pre-trained language models
- In particular, generative models such as GPT-2 model, or the T5 model are used
- They are mostly fine-tuned on the CANARD dataset or QReCC
- In some cases, generated query reformulation is further expanded with terms chosen from conversation history, with its own paraphrase, or the related topic sentences from the semantically-associated documents

Conversational Search Systems

- TREC CAsT
- Query Rewriting
- State of the art CSS

State of the art CSS - main components

- Inverted index and ANN index
- Query rewriting:
 - T5 model fine-tuned on the QReCC dataset
- First-pass retrieval:
 - Sparse retrieval with BM25 on queries extended with pseudo-relevance feedback
 - Dense retrieval with ANCE dense retriever
 - Final ranking is a reciprocal rank fusion of rankings returned by sparse and dense retrievers
- Re-ranking:
 - pointwise re-ranking with monoT5
 - pairwise re-ranking with duoT5

State of the art CSS - schema of architecture

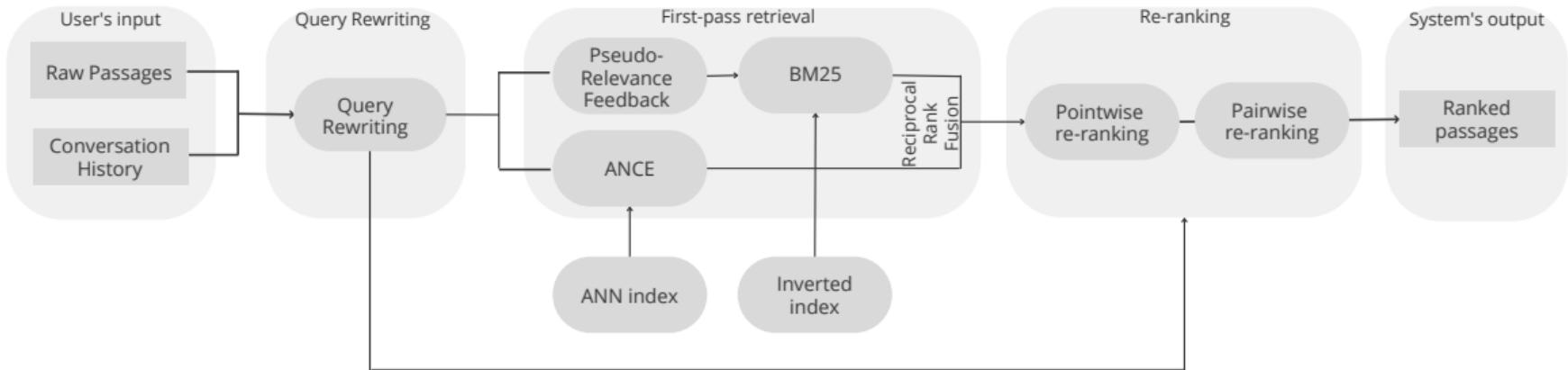


Figure: State of the art system from TREC CAsT'21

State of the art CSS - reproducibility⁵

- The reproducibility study aims to check whether the measurement can be obtained with:
 - stated precision by a different team using the same measurement procedure
 - the same measuring system
 - under the same operating conditions
 - in the same or a different location on multiple trials
- For computational experiments, this means that an independent group can obtain the same result using the author's own artifacts

⁵<https://www.acm.org/publications/policies/artifact-review-badging>

State of the art CSS - key challenges in reproducibility

- TREC systems are commonly regarded as reference points for effectiveness comparison
- TREC CAsT is a competition, so the top-performing teams are not willing to share their code with the community
- Top performing systems are very complex, they use many parameters and usually several stages
- The end-to-end performance of the system may not explicitly indicate which component is causing a drop in effectiveness
- Intermediate files, such as file rewrites and first-pass retrieval rankings are usually not shared, which makes component-based analysis impossible

Mixed-initiative

Mixed-initiative

- Mixed-initiative interaction - a flexible interaction strategy in which each agent (human or computer) contributes what it is best suited at the most appropriate time⁶
- Mixed-initiative systems can take control of the communication either at the dialogue level (e.g., by asking for clarification or requesting elaboration) or at the task level (e.g., by suggesting alternative courses of action)
- Mixed-initiative is more human-like and flexible because both actors can independently contribute to the conversation
- Main goal: improving the passage ranking effectiveness by allowing the systems to take the initiative at any point in a conversation

⁶Principles of Mixed-Initiative User Interfaces,
<https://dl.acm.org/doi/pdf/10.1145/302979.303030>

Mixed-initiative at TREC CAsT

- In the mixed-initiative sub-task at TREC CAsT, the system can pose questions to the user to gain additional context for a turn
- The supported question types for the task are: (1) elicit the task, (2) ask for feedback, or (3) clarify ambiguity

Example

User: What are some cool things to do in California?

MI-System: California is very large, what area would you like to visit?

MI-User: I'd like to explore Northern California.

System revealment

- System revealment - the system allowing the user to learn about the system's abilities, building the user's expectations of what it can and cannot do
- System revealment in terms of its confidence in the provided response:
 - the information need is not clear and the system is not able to find the answer → asking clarifying questions
 - the information need is clear to the system but the question is unanswerable in the collection used → handling an answerable question
- Main goals: CSS's explainability and transparency

Clarifying questions

- Search queries are often short, and the underlying user intent may be ambiguous
- This makes it challenging for search engines to predict possible intents, only one of which may pertain to the current user
- To address this issue, search engines often diversify the result list and present documents relevant to multiple intents of the query
- Clarifying questions can be used by the system to resolve ambiguity, to prevent potential errors, and in general to clarify user's requests and response
- Main challenge: trade-off between the efficiency of the conversation and the accuracy of the information need as the system has to decide between how important it is to clarify and how risky it is to infer or impute the underspecified or missing details

Demo

E13-1 - Clarifying questions

Unanswerable questions

- Open challenges in the conversational search:
 - Detecting questions for which no good answer exists in a corpus and informing the user that the answer has not been found
 - Detecting questions for which the answer is partially present and identifying the missing part
- Datasets containing unanswerable questions:
 - SQuAD 2.0 dataset created for extractive reading comprehension tasks
 - QuAC dataset containing information-seeking dialogs over sections from Wikipedia articles with unanswerable and open-ended questions

Unanswerable questions - master thesis project

- Project is based on a conversational search system (used in the previous editions of TREC CAsT) built with three components: query rewriting, first-pass retrieval, and re-ranking
- The goal is to build a classifier for the answerability of the given query that uses the final ranking provided by the system
- The student is expected to build a baseline classifier and one advanced method
- The experiments should focus on the performance of the classifier and on different ways of handling its output in the conversational search system
- The student is expected to propose two different methods of handling the unanswerable questions in the system's response
- An alternative task can be creating a new dataset for handling unanswerable questions in conversational search systems

Summary

- Conversational Search Systems
 - Recap on conversational information access
 - TREC CAsT
 - Query rewriting
 - State of the art
- Mixed initiative
 - Clarifying questions
 - System revealment

Reading

- A *Theoretical Framework for Conversational Search*, Filip Radlinski, Nick Craswell
<https://dl.acm.org/doi/pdf/10.1145/302979.303030>
- *Conversational search (Dagstuhl Seminar 19461)*, Avishek Anand, Lawrence Cavedon, Hideo Joho, Mark Sanderson, and Benno Stein
https://drops.dagstuhl.de/opus/volltexte/2020/11983/pdf/dagrep_v009_i011_p034_19461.pdf
- *Conversational information seeking, Chapter 2*, Hamed Zamani, Johanne R. Trippas, Jeff Dalton, Filip Radlinski, <https://arxiv.org/abs/2201.08808>

Fairness and Transparency in IR

[DAT640] Information Retrieval and Text Mining

Nolwenn Bernard
University of Stavanger

October 31, 2022



CC BY 4.0

In this module

1. Sociotechnical system
2. Fairness in IR
3. Transparency in IR

Sociotechnical system

Applications of IR

- Search engines
- Product and items recommendation, e.g., music, movies, books, news, etc.
- Candidate ranking, e.g., hiring platform, college admission, etc.
- Etc.



Position of the people

- Where do the people intervene in IR applications?
 - Algorithms and systems design, implementation, and evaluation (e.g., developers, annotators)
 - Consumers
 - Producers

Position of the people

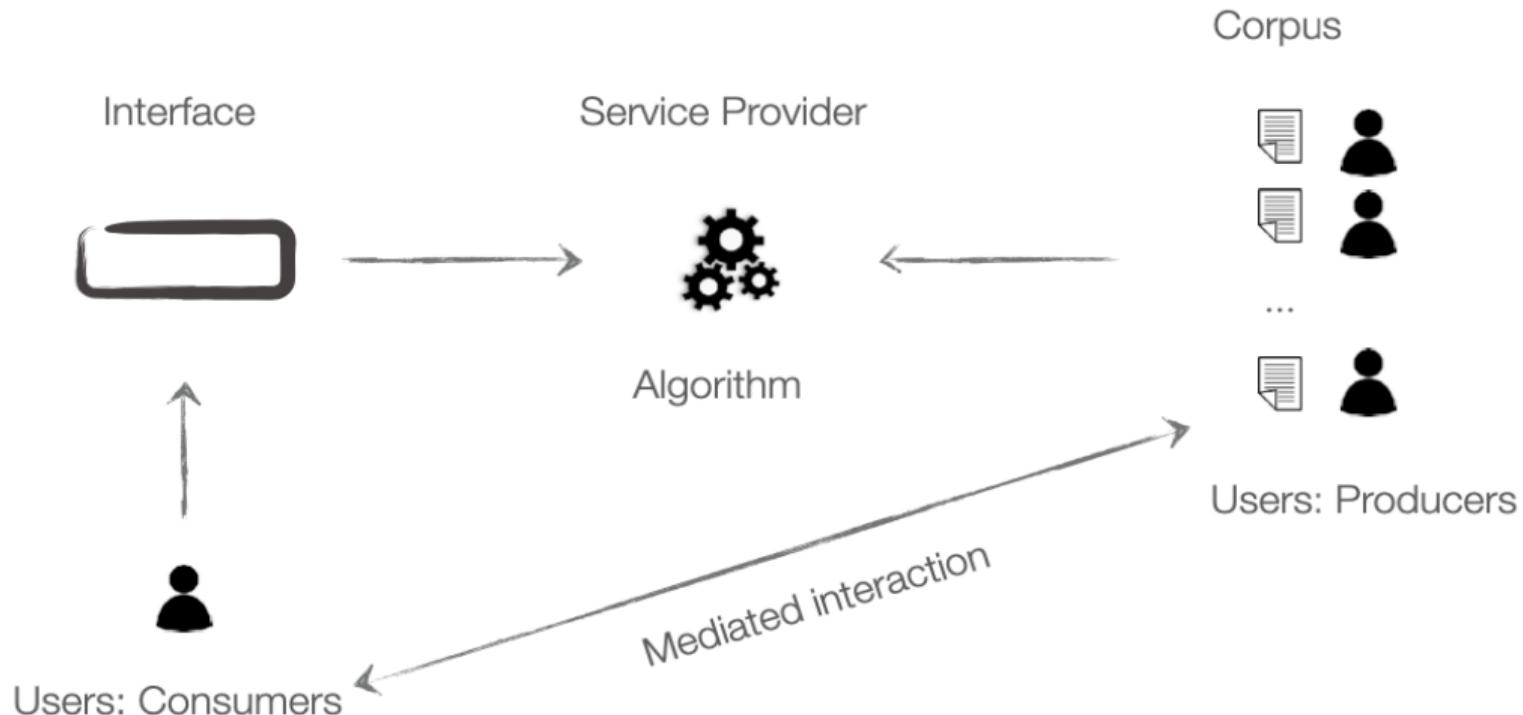


Figure: Information Retrieval system. Figure taken from Biega talk at ESSIR '22.

Sociotechnical system

- A IR system is a **sociotechnical system** due to the *omnipresence of people* in its ecosystem
 - It is used by producers and consumers
 - It learns from users behavior and feedback
 - It is designed and implemented by people
 - It can have an impact on the society, e.g., consumption habits, political outcomes
 - Etc.

Negative impacts

- **Distributional harms:** unfair distribution of a resource to an individual or group of consumers, e.g., unfair exposure of producers
- **Representation harms:** underrepresentation of a category of items from the corpus, e.g., independent movies, women
- **Privacy loss:** collection of consumers personal data, disclosure of producers information

Fairness in IR

Question

What does it mean to be fair?

Common definition

Fairness: “the quality of treating people equally or in a way that is reasonable.”¹

- *Broad* definition which can be *interpreted differently* depending on the people
- It raises several questions such as:
 - What is the goal? → Definition
 - Is the goal completed? → Evaluation
 - How can we complete the goal? → Method, implementation
- Taxonomies have been proposed to formalize fairness

¹<https://www.oxfordlearnersdictionaries.com>

Fairness taxonomy

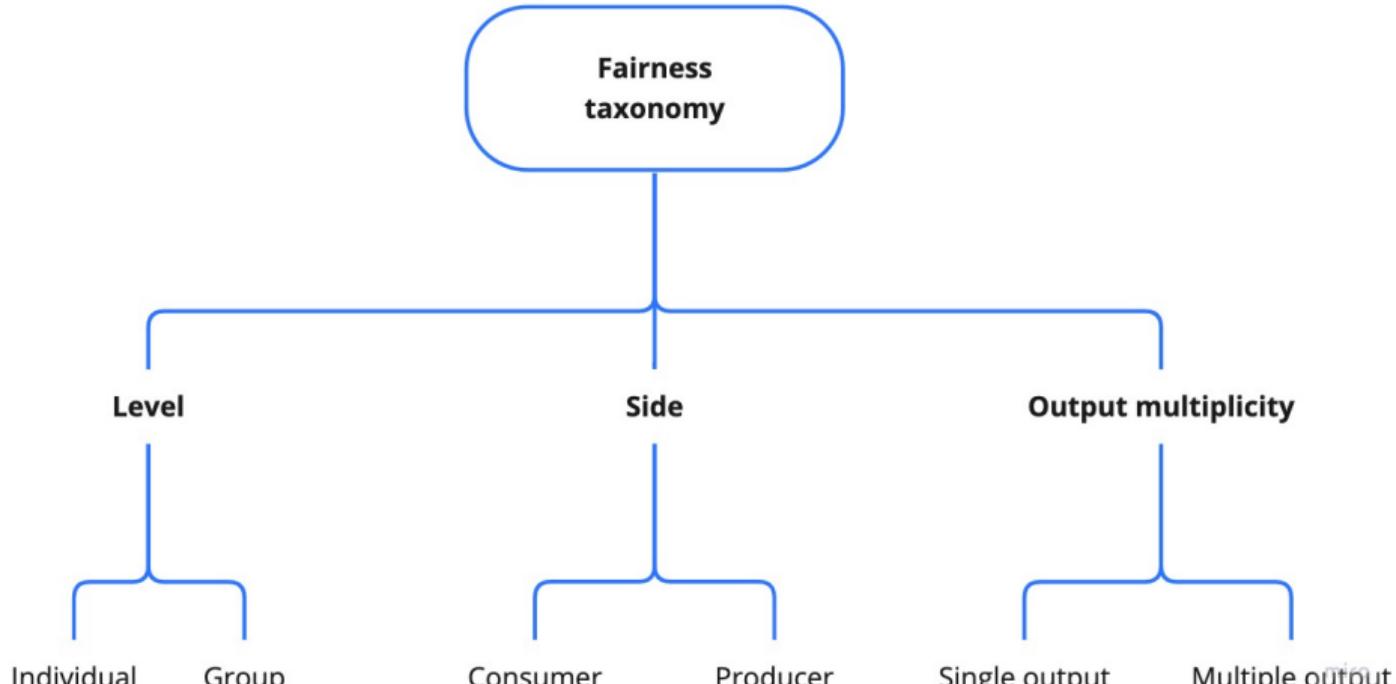


Figure: Fairness definition taxonomy (adapted from Pitoura et al., 2021)

Fairness dimensions

Level

- **Individual:** ensure consistent treatment of similar entities
- **Group:** ensure consistent treatment of entities belonging to a group

Example

Search Query	Work Experience	Education Experience	Candidate	Xing Ranking
Brand Strategist	146	57	male	1
Brand Strategist	327	0	female	2
Brand Strategist	502	74	male	3
Brand Strategist	444	56	female	4
Brand Strategist	139	25	male	5
Brand Strategist	110	65	female	6
Brand Strategist	12	73	male	7
Brand Strategist	99	41	male	8
Brand Strategist	42	51	female	9
Brand Strategist	220	102	female	10
...				
Brand Strategist	3	107	female	20
Brand Strategist	123	56	female	30
Brand Strategist	3	3	male	40

TABLE I: Top k results on www.xing.com (Jan 2017) for an employer's job search query "Brand Strategist".

Figure: Example taken from Lahoti et al., 2019

- Group fairness is achieve in top-10
- Unfair to individuals with similar qualifications

Fairness dimensions

Side

- **Consumers:** ensure that similar users or group of users receive similar output
 - Example: democrats should have similar results when looking for information on gun regulations
- **Producers:** ensure that similar items or group of items are ranked/recommended in a similar way
 - Example: in candidate ranking, gender should not influence the final ranking

Fairness dimensions

Output multiplicity

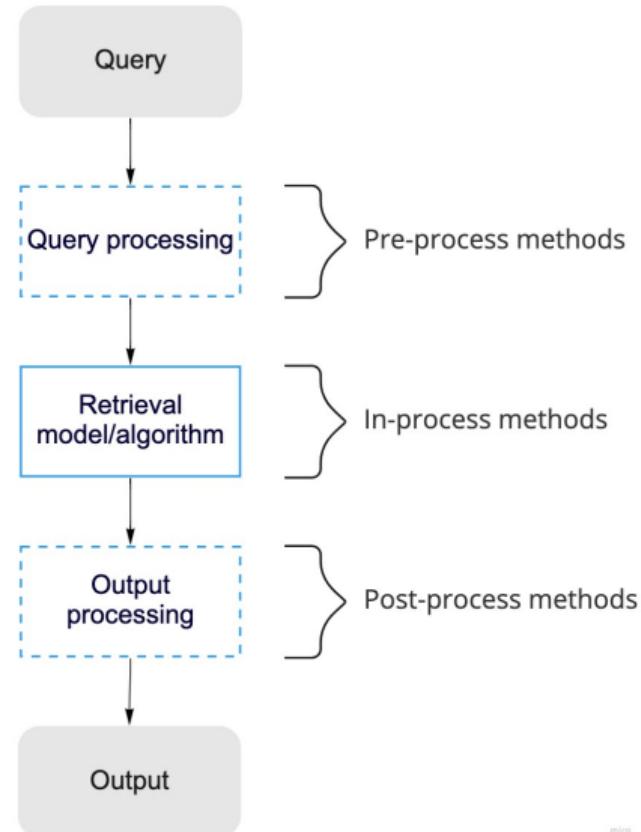
- **Single:** fairness is studied on only one output
- **Multiple:** fairness is studied on a sequence of outputs as a whole
 - Some rankings can be considered unfair in a fair system

Approach type

- Different types of solution exist to tackle the problem of fairness
 - **Model specific:** the approach does depend on the model used for information retrieval
 - **Model agnostic:** the approach does not depend on the model used for information retrieval

Approach position

- **Pre-process:** the approach is used before the retrieval process
- **In-process:** the approach modifies the retrieval process
- **Post-process:** the approach modifies the output of the retrieval process



Current trends

- Most of the literature focus on:
 - Group fairness
 - Single output
 - Item-side fairness
- Difficulty to establish state-of-art approach due to the number definitions
- A common approach is *post-process* and *model agnostic*

Exercise

E14-1 Measuring fairness in ranking

Transparency in IR

Common definition

Transparency: “the quality of something, such as a situation or an argument, that makes it easy to understand.”²

- *Broad* definition which opens the door for many approaches
- Operationalization raises similar questions as for fairness

²<https://www.oxfordlearnersdictionaries.com>

Question

How can we make a system transparent?

Modality

Several modalities are possible to communicate with the users:

- **Article:** communication information in an article, or a blog post
 - Example: PubMed Best Match algorithm (Fiorini et al. 2018)
- **Open source:** build the system with open source resources
- **User Interface (UI):** communicate with the user through the UI (e.g., explanations, icons)

Example

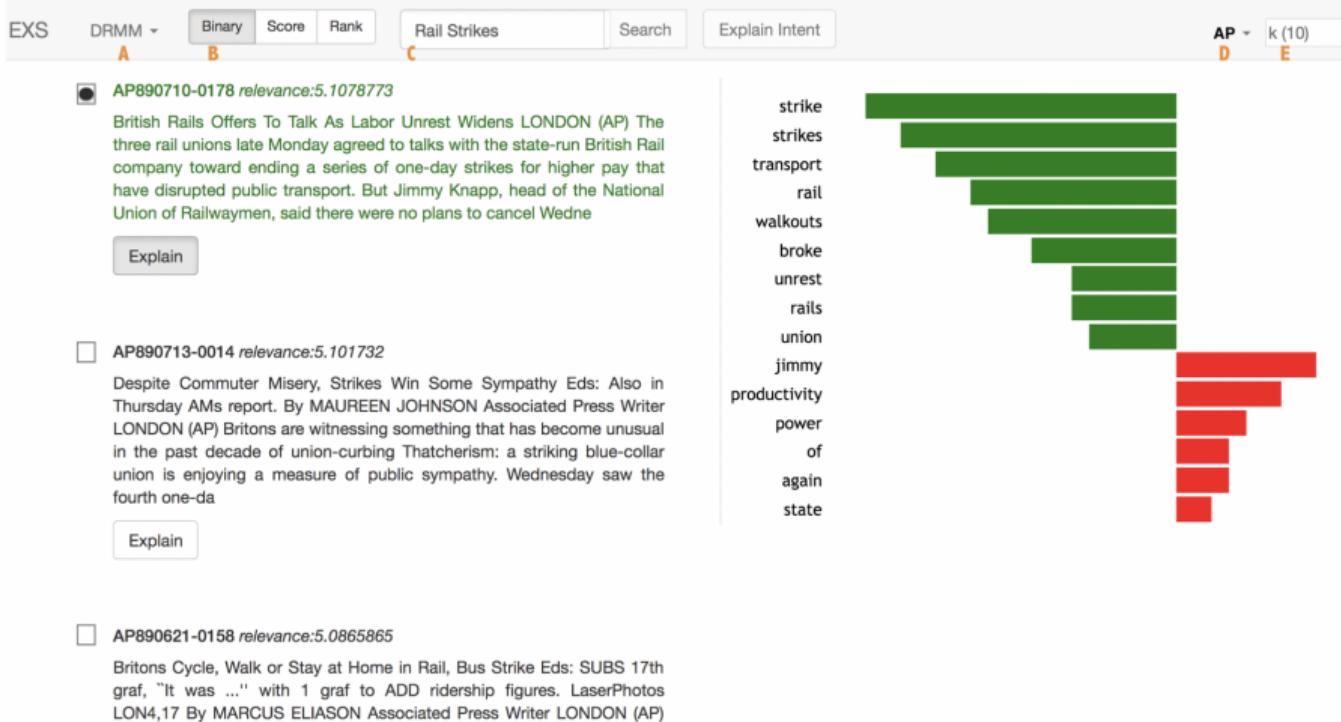


Figure: Example of the EXS user interface (Singh and Anand, 2019)

Challenges

There are many challenges related to transparency of the system, such as:

- How to be as transparent as possible without revealing business secrets to competitors?
- How to be transparent and prevent subsequent malicious behavior?
 - Example: search engine optimization, *i.e.*, website optimization to get a better ranking from the search algorithm
- Transparency of blackbox models
 - Example: explain a ranking produced by a transformer model

Transparency taxonomy

- Based on previous work and existing challenges, we identify two other dimensions for transparency:
 - Degree
 - Level

Degree of transparency

Degree

- **Full:** provide all the information necessary to make the entire system transparent
 - Example: description of the indexing and BM25 retrieval model for a search engine
- **Partial:** provide all the information necessary to make a part of system transparent
 - Example: description of the indexing, BM25 retrieval model, and transformer-based re-ranker for a search engine
 - The transformer-based re-ranker is not transparent yet

Level of transparency

Level

- **Global:** describe the inner-workings of the system
- **Local:** describe the relationship between a specific input and output
- **Causality:** describe the relationship between the inner-workings of a system (*i.e.*, the cause) and a specific output (*i.e.*, the effect)

Current trends

- A common approach is *post-process* and *model agnostic*
- Work in the literature tend to favor:
 - Local explanations
 - Communicate explanations through the UI

Exercise

E14-2 Ranking explanation

Summary

- IR systems are sociotechnical systems
 - Position of the people
 - Negative harms (incl. distributional and representational harms, privacy loss)
- Fairness in IR
 - Fairness taxonomy (incl. level, side, output multiplicity)
 - Fairness approach categorization
 - Current trends
- Transparency in IR
 - Transparency dimensions (incl. modality, degree, level)
 - Challenges
 - Current trends

Reading

- Pitoura et al. **Fairness in rankings and recommendations: an overview**. In: *The VLDB Journal* (2021)
 - <https://link.springer.com/article/10.1007/s00778-021-00697-y>
- Talk by Asia J. Biega at ESSIR '22. **Responsible Design of Information Access Systems**
 - Video: <https://www.youtube.com/watch?v=BnoiNTsUNZU>
 - Slides: <http://essir2022.org/slides/asia-biega.pdf>