Lecture 9: Information Retrieval

Information retrieval

- Information retrieval (IR) denotes methods for finding documents, or parts of documents, from large collections (specific gathered collections online and offline, and the web)
- In an **ad-hoc retrieval problem**, the user enters a query and the system returns one or more documents suggested to be relevant, usually in ranked order
- The typical IR setting is retrieval of text documents based on text queries, but there are several other cases:
  - Retrieval of images, audio, video, based on text queries
  - Retrieval based on something other than a text query: retrieval based on an image (e.g. google reverse image search, product search by image on various shopping sites), retrieval based on audio (e.g. finding songs by humming)
  - Interactive retrieval where the user provides **relevance feedback** (which results are relevant) beyond the original query
  - Personalized retrieval: the user's profile influences what is shown

Information retrieval

- For retrieving text documents, a simple approach is **exact match**: given a query term or phrase, return only those documents that exactly contain that term/phrase
- Multiple exact-match requirements can be combined as **Boolean queries** by logical operators AND (∧), OR (∨), NOT (¬): e.g. (("sports news" AND "goal") AND (NOT "football")) OR "playoffs"

$$\left(\left(\text{"sports news"} \wedge \text{"goal"}\right) \wedge \left(\neg \text{"football"}\right)\right) \vee \text{"playoffs"}$$

- However, this can yield bad results:
  - **The list of results might be too small, or even empty:**
    - If the query terms are written differently in part of the collection
    - If the terms do not cover sufficiently many things mentioned in the desired kind of content
    - If the combination is too strict, e.g. using AND operators if documents typically use only one/few of many terms

Information retrieval

  - **The list of results might be too large:**
    - If the query terms are too broad (appearing in documents unrelated to the desired content)
    - If the combination is too permissive, e.g. using OR operators if each term alone is too broad
  - **The list of results is not ranked:** a document either satisfies the Boolean requirement or not

- **Filtering** denotes classifying documents as relevant/nonrelevant.
  - Filtering is sometimes done by known metadata of documents, e.g. publication time or a category such as a news category.
  - Metadata filtering is an additional Boolean AND operation for the metadata requirements, in addition to the content requirements.
- **Routing** denotes document ranking but unlike ad-hoc retrieval, using training infrmation of known relevance labels

Information retrieval

- Simple Boolean searching could be implemented using an **inverted index**: for each word/collocation/phrase of the vocabulary, a list of all documents where it occurs, and the frequency of the occurrence.
- Then it is easy to find sets of "hits" satisfying each clause of a Boolean statement.
- The whole statement is then satisfied by a subset, formed by intersections, unions, and negations of the individual sets of "hits":
  - $\left(\text{statement1} \wedge \text{statement2}\right)$ **is satisfied by** $\text{hits}\left(\text{statement1}\right) \cap \text{hits}\left(\text{statement2}\right)$
  - $\left(\text{statement1} \vee \text{statement2}\right)$ **is satisfied by** $\text{hits}\left(\text{statement1}\right) \cup \text{hits}\left(\text{statement2}\right)$
  - $\left(\neg\text{statement1}\right)$ **is satisfied by** $\mathrm{D} \cap \left(\text{hits}\left(\text{statement2}\right)\right)^{C}$ where D is the document collection and C denotes the complement operation

Information retrieval

- A more advanced inverted index can also store where inside each document each word occurs (**position information**).
- Position information allows fast searching of e.g. consecutive words ("new york city") even if the collocations were not indexed beforehand
  - Find documents sets where "new", "york", and "city" occur, take their intersection
  - Then find from that subset those documents where the words occur adjacently
  - This will not find variant phrasing like "city of new york"
- Often the indices are built on stemmed/lemmatized words and after stopword and other vocabulary pruning

- Several retrieval mechanisms can go beyond Boolean queries and rank documents by **how well they seem to match the query**. We will discuss the algorithms after looking at some evaluation methods first.

Information retrieval

- Several evaluation measures have been proposed for IR.
  - Assume that each document can be categorized as "relevant" or "not relevant" to the information need intended by the search.
  - Denote the set of all relevant documents in the collection D by D^{relevant} and all nonrelevant documents by $D^{\text{nonrelevant}}$ so that $D = D^{\text{relevant}} \cup D^{\text{nonrelevant}}$.
  - Suppose the search returns a set of documents: $D^{\text{retrieved}}$, and the rest are unretrieved $D^{\text{unretrieved}}$ so that $D = D^{\text{retrieved}} \cup D^{\text{unretrieved}}$
  - If the search returns a ranked list instead of a set, any cutoff for the ranking can divide documents as $D^{\text{retrieved}}$ and $D^{\text{unretrieved}}$.
  - **Precision** is the proportion of relevant documents among retrieved ones:

$$\text{precision} = \frac{|D^{\text{relevant}} \cap D^{\text{retrieved}}|}{|D^{\text{retrieved}}|}$$

  - **Recall** is the proportion of retrieved documents among relevant ones:

$$\text{recall} = \frac{|D^{\text{relevant}} \cap D^{\text{retrieved}}|}{|D^{\text{relevant}}|}$$

Information retrieval

- The value of precision and recall depends on how many documents are retrieved. It is easy to see that as $\mathrm{D}^{\mathrm{retrieved}} \rightarrow D$, recall approaches 1, but precision approaches $|\mathrm{D}^{\mathrm{relevant}}|/|\mathrm{D}|$
- If the retrieval mechanism works well, usually it can rank documents so that most of the top-ranked documents are relevant but less and less further down the ranking. In that case, when the amount of retrieved documents is increased, recall keeps increasing but precision keeps decreasing: a **precision-recall tradeoff**.
- Sometimes precision is evaluated at several amounts of retrieved documents: **precision at K**, e.g. **precision at 5** or **precision at 10**.
- The **F-measure** combines precision and recall: it is small if either precision or recall are small.

$$\text{F-measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

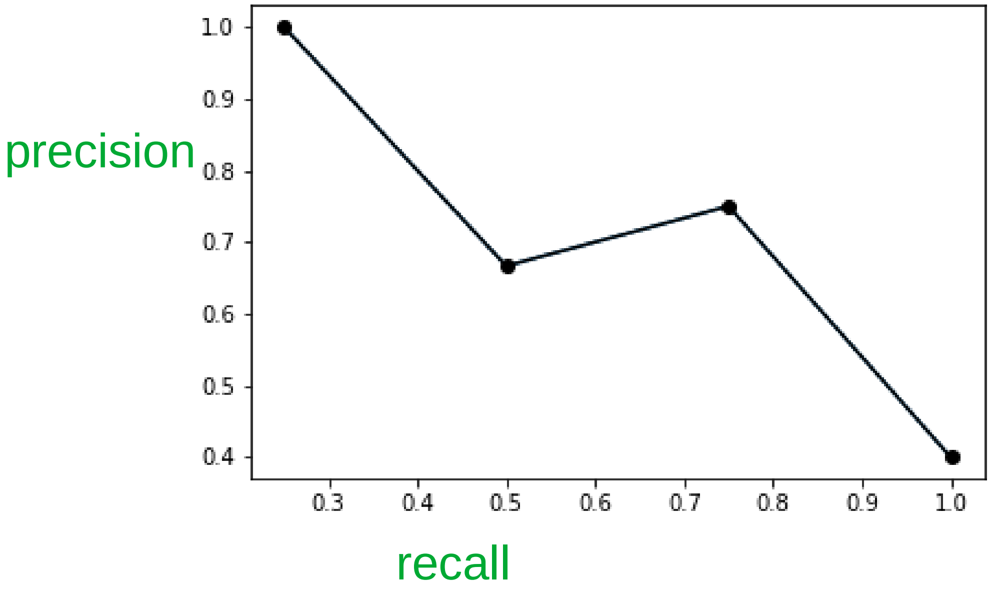
- Some approaches try to take into account the ranking of documents, not just a division into retrieved and unretrieved

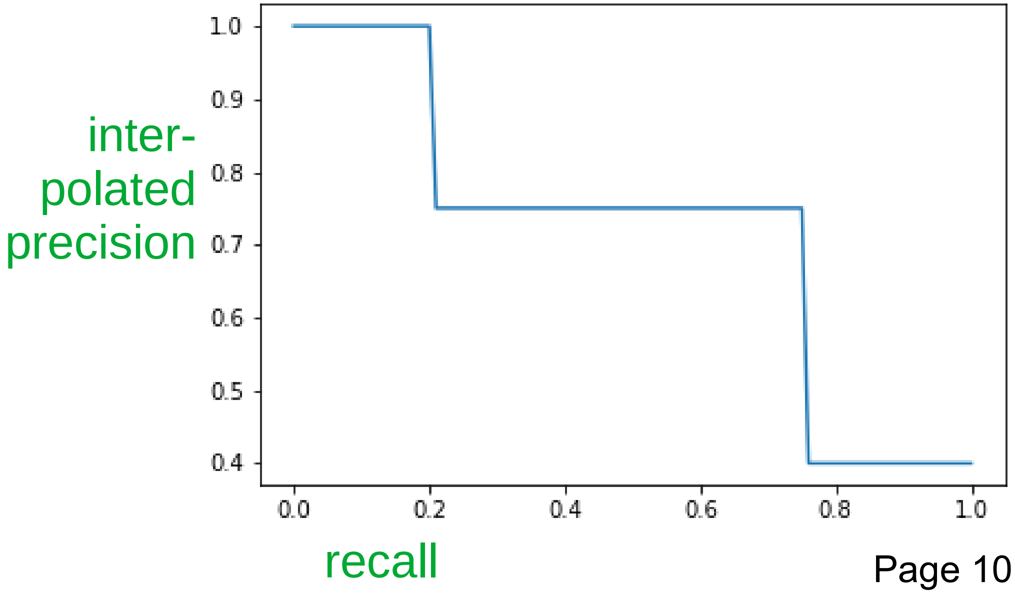
Information retrieval

- Suppose the retrieval mechanism provides a ranking of documents, 1, ..., |D|, and suppose we know for each document is it relevant or not. **Uninterpolated average precision** averages precision values at cutoffs corresponding to the **position of each relevant document**.
  - Example: suppose there are four relevant documents in the collection, at positions 1, 3, 4, and 10 of the ranking. The uninterpolated average precision is then the average of *precision at 1* (which is here 1), *precision at 3* (here 2/3), *precision at 4* (3/4), and *precision at 10* (4/10): $\frac{1}{4}\left(1+\frac{2}{3}+\frac{3}{4}+\frac{4}{10}\right)\approx 0.70$
- **Interpolated average precision** computes the average of precision at ranking positions that reach specified values of recall: e.g., at the positions where recall is 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100%. For each recall value, **interpolated precision** is taken as the maximum of any precision at or after that position.
  - In the example: the recalls are reached at positions 1 (0%), 1 (10%), 1 (20%), 3 (30%), 3 (40%), 3 (50%), 4 (60%), 4 (70%), 10 (80%), 10 (90%), 10 (100%). The maximal precisions at those points or later are: 1, 1, 1, 3/4, 3/4, 3/4, 3/4, 3/4, 4/10, 4/10, 4/10, average: 0.72

Information retrieval

- Precision-recall curves plot, over different values of recall, either the precision at that position in the ranking where the recall was achieved, or the interpolated precision (maximum precision at that position or later in the ranking)
  - In the example: For the uninterpolated plot, recalls 25%, 50%, 75% and 100% are reached at positions 1, 3, 4, and 10. The precisions at those points are: 1, 2/3, 3/4, and 4/10.
  - For the interpolated plot, let's plot at 1% intervals of recall. Recalls are reached at positions 1 (0-20%), 3 (21-50%), 4 (51-75%), 10 (76-100%). Interpolated precisions at those positions are: 1, 3/4, 3/4, and 4/10.





Information retrieval

- **Probability ranking principle**: documents should be ranked by decreasing probability of relevance (van Rijsbergen, 1979)
  - Retrieval becomes a greedy search to locate the most probably relevant documents
  - Does not take into account e.g. **redundancy**: (near-)duplicate documents containing the same information could appear
  - When several queries are considered, a document could be relevant to each of them, but not among the top-ranked in any of them. Such a document could still be overall relevant.
  - Relevances are estimated based on document content, and the model that estimates them may not capture the user's information need well, or (especially in simple models) may not model the document's content and its relevance well. Thus the probabilities of relevance are **uncertain**. Some models can estimate the amount of uncertainty - we will talk about this later in interactive retrieval.

Information retrieval

- The **vector space model** (lectures 3-4) can be used for information retrieval.
- The query text (e.g. "linux dvb-t driver install help") is processed into a TF-IDF representation just like the text documents in the collection. Any TF-IDF variant can be used.
- Then the cosine similarity is computed between the query vector **q** and each document vector **x** (here D denotes the dimensionality):

$$\cos(\boldsymbol{q},\boldsymbol{x})=\frac{\boldsymbol{q}^T\boldsymbol{x}}{\|\boldsymbol{q}\|\cdot\|\boldsymbol{x}\|}=\frac{\sum_{d=1}^{D} q_d x_d}{\sqrt{\sum_{d=1}^{D} q_d^2}\cdot\sqrt{\sum_{d=1}^{D} x_d^2}}$$

- Documents are ranked in decreasing order of the cosine similarity.
  - This is the same as ascending order of the angle between the query and document vectors
  - Same as ascending order of Euclidean distance between query and document vectors when their lenghts are normalized to 1.

Information retrieval

- Language models such as unigrams, n-grams, topic models, or **vector space model** (lectures 3-4) can be used for information retrieval.
- A language model is built for each document in the collection. Usually every document builds the same type of language model (e.g. a unigram model), and the models of the different documents d each have their own optimized parameters $\boldsymbol{\theta}_d$
- Then, each language model is used to compute the **probability of the query**: $p(\mathbf{q}|\boldsymbol{\theta}_d)$ probability to observe the contents of the query as a document generated from the language model.
- The likelihood of each language model is the probability that it gives to the query.
- Documents are ranked by the likelihood of their language models, highest first.

Information retrieval

- **Example with unigram language models:** a query is represented by its set of words (bag of words), or equivalently, how many occurrences of each vocabulary word appear in it.

$$\boldsymbol{q}=[n_1^{(q)},\ldots,n_V^{(q)}]$$

- The probability of the query in a unigram language model is:

$$p(\boldsymbol{q}|\boldsymbol{\theta}_d)=\frac{N^{(q)}!}{n_1^{(q)}!\,n_2^{(q)}!\cdots n_V^{(q)}!}\prod_{v=1}^{V}\theta_{d,v}^{n_v^{(q)}} \quad \text{where} \quad N^{(q)}=\sum_{v=1}^{V}n_v^{(q)}$$

- The part with the factorials is the same for every document d, so it can be treated as a constant for an individual query.
- Usually log-probability is used: $\log p(\boldsymbol{q}|\boldsymbol{\theta}_d)=const+\sum_{v=1}^{V}n_v^{(q)}\log\theta_{d,v}$
- Suppose that each document uses a maximum likelihood estimate from its own document content to build its language model, i.e.

$$\boldsymbol{\theta}_{d,ML}=\left[\frac{n_{d,1}}{N_d},\ldots,\frac{n_{d,V}}{N_d}\right]$$

then the log-probability of the query is

$$\log p(\boldsymbol{q}|\boldsymbol{\theta}_d)=const+\sum_{v=1}^{V}n_v^{(q)}\log(n_{d,v}/N_d)$$

Information retrieval

- **Problem: queries may contain a combination of words that do not occur in any document.**
- If any word v in the query does not appear in document d, the maximum likelihood probability estimate is zero in the model, and the query likelihood becomes minus infinity.
- This is overly strict for ranking, as documents would end at the bottom of the ranking even if they contain **most** of the query words, and there would be no way to rank among such documents.
- Solution: **smooth** the probabilities in the language model. Similar to MAP estimation of the language model.

Information retrieval

- **Simple Dirichlet smoothing:** add a pseudocount to every word to every document. Corresponds to maximum a posteriori (MAP) estimation of the unigram model with a Dirichlet prior

$$\theta_{MAP}=\left[\frac{n_1+\alpha_1}{N+\sum_i \alpha_i},\ldots,\frac{n_V+\alpha_V}{N+\sum_i \alpha_i}\right]$$

- **Jelinek-Mercer (JM) smoothing:** probability of the word is a mix of the maximum-likelihood estimate from the document $p_{d,v,ML}=n_{d,v}/N_d$ and the proportion of the word in the entire collection $p_{C,v}=n_{C,v}/N_C$.

$$\theta_{d,JM}=\left[\lambda\frac{n_{d,1}}{N_d}+(1-\lambda)\frac{n_{C,1}}{N_C},\ldots,\lambda\frac{n_{d,V}}{N_d}+(1-\lambda)\frac{n_{C,V}}{N_C}\right]$$

- The JM equation can also be written as a MAP estimate with a particular Dirichlet prior (exercise!)

- The JM equation can also be written as $$p_{d,v,JM}=(1-\lambda)p_{C,v}\left(1+\frac{\lambda}{(1-\lambda)p_{C,v}}p_{d,v,ML}\right)$$

- Then $$\log p(\boldsymbol{q}|\boldsymbol{\theta}_d)=const+\sum_{v=1}^{V} n_v^{(q)}\log\left(1+\frac{\lambda}{(1-\lambda)p_{C,v}}p_{d,v,ML}\right)$$ positive-valued score

Information retrieval

- **Language models beyond unigrams can be used in a similar way.** (Note: requires an efficient way to build the language model for each document.)
- **Unigrams can already work well in many cases** - web queries may not be written as grammatical sentences, so trying to model their word order in the same way as sentences inside documents may not always be beneficial.
- However, e.g. detecting phrases from queries can help (e.g. documents with "machine learning" instead of just machine and learning.
- There are several different search engines implementing unigram language models and smoothing such as Jelinek-Mercer.
  - Apache Lucene (https://lucene.apache.org/)
  - ElasticSearch (https://www.elastic.co/products/elasticsearch)
  - Apache Solr (https://lucene.apache.org/solr/)

Information retrieval

- The likelihood of documents can be combined with a **prior**, to yield a **maximum a posteriori** ranking: rank documents by sum of log-likelihood and **log-prior probability** of the document.
- The prior represents how important / relevant we expect the document to be before seeing the content.
- When documents are linked to each other (citations of scientific publications; hyperlilnks of webpages), the **PageRank** algorithm can be used as a prior for importance of documents
- Idea: importance of a document is computed based on importances of documents linking to it. This recursive/iterative computation can be solved to find importances of each document.
- Denote links as (d,d') where d is the page with the link and d' is the page it points to. Denote L as the total set of links.
- The PageRank value of a document d is

$$\pi_d = \sum_{(d',d)\in L} \frac{\pi_{d'}}{|\{(d',\cdot)\in L\}|}$$

outgoing links from d'

links pointing to d

Information retrieval

- Usually, a version with a "damping factor" is used instead (e.g. set to $\gamma=0.85$): when M is the number of documents in the collection, the result is

$$\pi_d=\frac{1-\gamma}{M}+\gamma\sum_{(d',d)\in L}\frac{\pi_{d'}}{|\{(d',\cdot)\in L\}|}$$

- The PageRank idea has a probabilistic interpretation: suppose there is a "random surfer" who, after arriving at each document (webpage), either: with probability γ **continues** by following a randomly chosen outgoing link of the document (equal probability to choose any link) or with probability $1-\gamma$ **restarts** by picking a random document of the collection.

- This is a Markov chain, where documents are states, outgoing links are possible transitions, .

- The PageRank equation corresponds to the requirement of a steady-state distribution of the Markov chain: the amount of probability mass ("importance") at each state/document must equal the amount of incoming probability mass from other states/documents.

Information retrieval

- The PageRank values are the steady-state probabilities of the chain: probabilities that after a very large number of steps the "random surfer" is at a particular document

- How to compute the steady-state probabilities: denote by **A** a Markov transition matrix of documents according to the damped random surfer model: the element in row i, column j is:

  $a_{ij} = \frac{1-\gamma}{M}$ if document j does not link to document i, and

  $a_{ij} = \frac{1-\gamma}{M} + \gamma \frac{1}{|\{(j,\cdot)\in L\}|}$ if it does.

  each column of this matrix sums to 1

- Then the steady-state distribution vector $\boldsymbol{\pi}$ of the Markov chain must satisfy the linear system

  $\boldsymbol{A}\boldsymbol{\pi} = \boldsymbol{\pi}$

  and therefore $\boldsymbol{\pi}$ must be the solution of the linear system.

- This is also an eigenvalue equation where the eigenvalue is 1 and $\boldsymbol{\pi}$ is the eigenvector. But note that **A** is not a symmetric matrix.

Information retrieval

- The solution can be found by a power method: start from some $\boldsymbol{\pi}$ and multiply repeatedly by **A** until $\boldsymbol{\pi}$ converges.
- Alternative: define **B** so that $b_{ij} = a_{ij} - \frac{1-\gamma}{M}$, then the steady-state equation becomes

  $$\boldsymbol{A\pi} = \left(\boldsymbol{B} + \frac{1-\gamma}{M}\mathbf{1}\mathbf{1}^T\right)\boldsymbol{\pi} = \boldsymbol{B\pi} + \frac{1-\gamma}{M}\mathbf{1}\left(\mathbf{1}^T\boldsymbol{\pi}\right) = \boldsymbol{B\pi} + \frac{1-\gamma}{M}\mathbf{1} = \boldsymbol{\pi}$$

  which is solved by $\boldsymbol{\pi} = \left(\boldsymbol{I} - \boldsymbol{B}\right)^{-1}\frac{1-\gamma}{M}\mathbf{1}$
- The logarithms of the resulting $\boldsymbol{\pi}$ can be added to the content-based scores to get a Maximum a Posteriori ranking of documents. The parameter $\boldsymbol{\gamma}$ defines the importance of the prior: when $\boldsymbol{\gamma}$ is near zero, **A** is nearly uniform, therefore $\boldsymbol{\pi}$ is near uniform, which does not affect the ranking of documents much.