TAR

3. Basics of Information Retrieval

TAR-03-IR.pdf

Information retrieval

- Information retrieval: The activity of obtaining information resources relevant to an user's information need from a collection of information resources.
  - Elements of an information retrieval system:
    - **Information needs** (expressed by users in the form of queries)
    - **Information (re)sources** (typically unstructured - text, images, video, audio, etc.)
    - **Component for efficient retrieval** of relevant sources for a given expressed information need, typically from a large collection of information sources
- Information need: Information need is an individual or group's desire to locate and obtain information to satisfy a conscious or unconscious need. Needs and interests call forth information.
  - (Un)conscious needs for information are expressed via **queries**
    - **Words and phrases** in text information retrieval (e.g., “ISIS attacks”)
    - **Images** in image content retrieval
- Text information retrieval: Relevance of the document is most often given as a score (and not a binary decision). Documents are ranked according to the assigned scores for the given query. Relevance scores usually incorporate an element of uncertainty

---

Text representations in IR

- Representations:
  - **Unstructured** representation:
    - Text represented as an unordered set of terms (the so-called bag-of-words representation – pairs word-count)
    - Considerable oversimplification: ignoring syntax and semantics (despite oversimplifying, satisfiable retrieval performance)
  - **Weakly-structured** representations:
    - Certain groups of terms given more importance (other terms' contribution downscaled or ignored) – bag of nouns, bag of named entity terms…
  - **Structured** representations:
    - Virtually not used in IR context
    - Information extraction (IE) techniques not sufficiently accurate
    - IE models can be time-costly - not acceptable in IR

- <u>Preprocessing</u>: reduces the cardinality of the bag-of-words set of the document and generally boost IR performance
  - **Morphological normalization**: stemming or lemmatization
    - Conflating various forms of the same word to a common form
    - Important for morphologically rich languages such as Croatian
    - Stemming (e.g., kućom → kuć) more often used than lemmatization (e.g., kućom → kuća)
  - **Removal of stop words**
    - Removing semantically-poor terms such as articles, prepositions, conjunctions, pronouns, etc.
    - Keeping just content words - nouns, verbs, adjectives, adverbs
- <u>A basic retrieval model</u> is a triple (f_d, f_q, r) where:
  1. f_d is a function that maps documents to their representations for retrieval, i.e., $f_d(d) = p_d$, where p_d is the retrieval representation of the document d
  2. f_q is a function that maps queries to their representations for retrieval, i.e., $f_q(q) = s_q$, where s_q is the retrieval representation of the query q
  3. r is a ranking function
     - Takes into account document representation p_d and query representation s_q
     - Associates a real number that indicates the potential relevance of the document d for the query q based on p_d and s_q
       - $relevance(d, q) = r\left(f_d(d), f_q(q)\right) = r\left(p_d, s_q\right)$
  - **Index terms** are all the terms in the collection (i.e., the vocabulary)
    - The set of all index terms - $K = \{k_1, k_2, \ldots, k_t\}$
    - Each term k_i is, for document d_j, assigned a weight ω_{ij}
    - The weight of index terms not appearing in the document is 0

    - Document d_j is represented by the term vector $[\omega_{1j}, \omega_{2j}, \ldots, \omega_{tj}]$, where t is the number of index terms
    - Let g be the function that computes the weights, i.e., $\omega_{ij} = g\left(k_i, d_j\right)$
    - Different choices for the weight-computation function g and the ranking function r define different IR models
  - Information retrieval models roughly fall into **three paradigms**:
    - Set theoretic models (Boolean model)
    - Algebraic models (Vector space model)
    - Probabilistic models (Classic probabilistic model, Language model)

    - Additionally, there are IR models that utilize link analysis algorithms (e.g., PageRank, HITS), typically used in web retrieval where documents are (hyper)linked

Boolean retrieval model

- Documents represented as **bags of words**
- Term weights are all binary - $\omega_{ij} \in \{0,1\}$
  - $\omega_{ij} = 1$ if index term k_i can be found in the bag of words of document d_j
- Query q is given as a propositional logic formula over index terms
  - Index terms are connected via Boolean operators (^,˅) and can be negated (⌐)
  - Each query q can be transformed into disjunctive normal form (DNF), i.e., $q = q_{c_1} \vee q_{c_2} \vee \ldots V\ q_{\ c_n}$ where q_{c_l} is the l-th conjuctive component of q's DNF
- The relevance of the document d_j for the query q is given as follows:

$$relevance(\ d_j, q) = \begin{cases} 1, if\ \exists q_{c_l} \mid \forall k_i \in terms(q_{c_l}), \omega_{ij} = 1 \\ \qquad\qquad 0, otherwise \end{cases}$$

- Inverted index: a data structure for computationally efficient retrieval
  - **Inverted file index** contains a list of references to documents for all index terms (e.g., L(Frodo) = $\{d_1, d_2, d_3\}$)
    - Inverted index allows for handling Boolean queries via set intersections and set unions
      $rel(D, q) =\ L$(Frodo) $\cap\ L$(stab) = $\{d_1, d_2, d_3\} \cap\ \{d_1, d_2\} =\ \{d_1, d_2\}$
  - **Full inverted index** additionally contains the positions of each word within a document (e.g. ,"Frodo" : $\{(d_1, 1), (d_2, 1), (d_3, 3)\}$)
- Boolean retrieval model **advantages**:
  - Only one: simplicity (computational efficiency)
  - Popular in early commercial systems
- Boolean retrieval model **shortcomings**:
  - Expressing information needs as Boolean expressions is unintuitive
  - A pure model:
    - No ranking - documents are either relevant or non-relevant
    - Relative importance of indexed terms is ignored
  - Extended Boolean model - a variant of the Boolean model that accounts for the partial fulfillment of the Boolean expression

Vector space model

- Documents and queries are represented as **vectors of index terms**
- Weights are real numbers ≥ 0
  - $d_j = [\omega_{1j}, \omega_{2j}, \ldots, \omega_{tj}]$
  - $q = [\omega_{1q}, \omega_{2q}, \ldots, \omega_{tq}]$
- The relevance of the document for the query is estimated by computing some distance or similarity metric between the two vectors
  - **Distance metrics** (more relevant when distance is lower): Euclidean, Manhattan

Euclidean distance

$$dis_E(\mathbf{d_j}, \mathbf{q}) = \sqrt{\sum_{i=1}^{t} (w_{ij} - w_{iq})^2}$$

Manhattan distance

$$dis_M(\mathbf{d_j}, \mathbf{q}) = \sum_{i=1}^{t} |w_{ij} - w_{iq}|$$

- **Similarity metrics** (more relevant when similarity is larger): Cosine, Dice

$$Cosine(\mathbf{d_j}, \mathbf{q}) = \frac{\mathbf{d_j} \cdot \mathbf{q}}{\|\mathbf{d_j}\|\|\mathbf{q}\|} = \frac{\sum_{i=1}^{t} w_{ij} w_{iq}}{\sqrt{\sum_{i=1}^{t} w_{ij}^2}\sqrt{\sum_{i=1}^{t} w_{iq}^2}}$$

$$Dice(\mathbf{d_j}, \mathbf{q}) = \frac{2\sum_{i=1}^{t} w_{ij} w_{iq}}{\sum_{i=1}^{t} w_{ij} + \sum_{i=1}^{t} w_{iq}}$$

- <u>How are weights ω_{ij} of index terms for documents computed</u>:
  1. The relevance of an index term for the document is proportional to its frequency in the document (**term frequency** component)
     - i.e., more frequent, more relevant
  2. The relevance of an index term for any document is inversely proportional to the number of documents in the collection in which it occurs (**inverse document frequency** component)
     - i.e., more common across documents, less relevant (e.g., stopwords such as "the")
- <u>Weighting schemes</u>:
  - **Binary** "weighting scheme"
    - Not really a weighting scheme, ignores the two aforementioned assumptions
    - $\omega_{ij} = 1$ if document d_j contains term k_i
  - **TF-IDF** weighting scheme
    - The weight computed as the product of the term frequency (TF) component and the inverse document frequency (IDF) component:
      - $\omega_{ij} = tf(k_i, d_j) \cdot idf(k_i, D)$
    - The most popular local and global schemes:
      - $tf(k_i, d_j) = 0.5 + \frac{0.5 \cdot freq(k_i, d_j)}{\max(freq(k, d_j) \mid k \in d_j)}$

Classic probabilistic retrieval

- <u>Probability</u>:
  - **Bayes' rule**:
    - $P(B|A) = \frac{P(A|B)P(B)}{P(A)}$
    - Allows us to compute $P(B|A)$ using $P(A|B)$
    - Particularly useful if the former is difficult to compute directly
  - **Chain rule**:
    - $P(A, B, C) = P(A|B, C)P(B|C)P(C)$
    - Allows us to write a joint probability using conditional probabilities.
- <u>Probabilistic retrieval models</u>:
  - View retrieval as a problem of estimating the probability of relevance given a query, document, collection, etc.
  - Documents are ranked in decreasing order of this probability
  - Some probabilistic models from this category: classical probabilistic model, two Poisson model, BM25, language model

- Probability ranking principle: If an IR system's response to each query is a ranking of the documents in the collection in order of decreasing probability of relevance, the overall effectiveness of the system to its user will be maximal.
  - Random variables:
    - $D = \{D_1, \dots D_t, \dots D_N\}$ - Document (set of terms)
    - $Q = \{Q_1, \dots Q_t, \dots Q_L\}$ – Query (set of terms)
    - $R \in \{0,1\}$ - relevance judgment
      - $R = 1$ if D is relevant for Q, $R = 0$ otherwise
  - "What is the probability that a user will judge this document as relevant for this query?"
    - Estimate: $\mathrm{P(R = 1 \mid D = d, Q = q)}$
- The heart of probabilistic retrieval models:
  - Let r be a shorthand for $R = 1$, and $\bar{r}$ for $R = 0$
  - We apply the logit function to probability (1)
    - $logit(p) = \log\left(\frac{p}{1-p}\right)$
  - This is a rank preserving transformation giving:
    - $log\frac{p(r|D,Q)}{1-p(r|D,Q)} = log\frac{p(r|D,Q)}{p(\bar{r}|D,Q)}$
  - Now we can apply the Bayes rule:
    - $log\frac{p(r|D,Q)}{p(\bar{r}|D,Q)} = log\frac{p(D,Q|r)p(r)}{p(D,Q|\bar{r})p(\bar{r})}$
    - A benefit of having used logit is that $P(D,Q)$ cancels out
  - Finally, we use the chain rule:

$$log\frac{p(D,Q|r)p(r)}{p(D,Q|\bar{r})p(\bar{r})} = log\frac{p(D|Q,r)P(Q|r)p(r)}{p(D|Q,\bar{r})P(Q|\bar{r})p(\bar{r})}$$
$$= log\frac{p(D|Q,r)}{p(D|Q,\bar{r})} + log\frac{P(Q|r)p(r)}{P(Q|\bar{r})p(\bar{r})} \propto log\frac{p(D|Q,r)}{p(D|Q,\bar{r})} \quad (2)$$

  - The right term in the second row doesn't depend on D, we can remove it without affecting the ordering (it can be interpreted as a measure of query difficulty)
  - Expression (2) is the heart of probabilistic retrieval models
- Binary Independence Model:
  - Our first attempt at estimating expression (2)
  - Documents are represented with a vector of binary random variables $D = < D_1, \dots, D_N >$ with one dimension for each term in vocabulary V
    - $D_i = 1$ denotes the presence of term i
    - $D_i = 0$ denotes the absence of term i
  - Similarly, the query is a vector $Q = < Q_1, \dots, Q_L >$
  - **Assumption 1**: given relevance terms are statistically independent
    - Now we can write the terms in (2) as follows:
      - $p(D|Q,r) = \prod_{i=1}^{|V|} p(D_i|Q,r)$
      - $p(D|Q,\bar{r}) = \prod_{i=1}^{|V|} p(D_i|Q,\bar{r})$
    - Turning expression (2) into:
    - $log\frac{p(D|Q,r)}{p(D|Q,\bar{r})} = \sum_{i=1}^{|V|} log\frac{p(D_i|Q,r)}{p(D_i|Q,\bar{r})}$ (3)

- **Assumption 2**: The presence of a term in a document depends on relevance only when that term is present in the query.
  - For a fixed query $Q = q =< q_1, \ldots, q_L >$, if $q_i = 0$ according to this assumption $P(D_i|q,r)$ does not depend on relevance:
    - $p(D_i|Q,r) = p(D_i|Q,\bar{r})$
    - $log\frac{p(D_i|Q,r)}{p(D_i|Q,\bar{r})} = 0$
  - We can simplify (3) by ignoring all summation terms not in q:
    - $\sum_{i=1}^{|V|} log\frac{p(D_i|Q,r)}{p(D_i|Q,\bar{r})} = \sum_{t\in q} log\frac{p(D_t|q,r)}{p(D_t|q,\bar{r})}$ (4)
  - We can view expression (4) as summing weights of terms
    - $\sum_{t\in q} log\frac{p(D_t|q,r)}{p(D_t|q,\bar{r})} = \sum_{t\in q} \omega_t$
- Although assumptions are often violated these models work well
- **Common practice** is to **approximate** $p(D_t|q,r)$ with 0.5 and $p(D_t|q,\bar{r})$ with $\frac{n_t}{N_d}$, where n_t - number of documents containing term i and N_d - total number of documents
- A very similar alternative is to use IDF scores as ω_t
- These probabilities can be reestimated through relevance feedback
- **Example**:
  - d_1: “Frodo and Sam stabbed orcs.”
  - d_2: “Sam chased the orc with the sword.”
  - d_3: “Sam took the sword.”
  - Query: “Sam stabbed orc”

	d_1			d_2		d_3
t	Sam stabbed orcs			Sam orc		Sam
$P(D_t\|q,r)$	0.5	0.5	0.5	0.5	0.5	0.5
$P(D_t\|q,\bar{r})$	3/3	1/3	2/3	3/3	2/3	3/3
ω_t	0.5	1.5	0.75	0.5	0.75	0.5
$\sum \omega_t$	2.75			1.25		0.5

yields d_1 as the most relevant result, followed by d_2 and

- <u>Two Poisson Model</u>:
  - A more realistic document representation, a vector of word frequencies
  - Uses the Poisson distribution to model frequencies.
  - **Assumption**: all documents are of equal length
  - Can be approximated by the following expression:
    - $\sum_{t\in q} \frac{f_{t,d}(k_1+1)}{k_1+f_{t,d}} \cdot \omega_t$ (6)
    - Where $f_{t,d}$ is the frequency of term t in document d and k_1 a constant (typically $1 \leq k_1 < 2$). Higher frequency words get boosted weights.

- BM11 (Best Matching):
  - **Removes document length assumptions** of the two Poisson model
    - matches in longer documents should be less important
  - We can correct the frequency $f'_{t,d} = f_{t,d}\left(\frac{l_{avg}}{l_d}\right)$
    - l_{avg} - the average length of a document
    - l_d - the length of document d
    - **dampens/boosts** word frequencies based on **above/below** average document length
  - Now we can rewrite (6) as:
    - $\sum_{t\in q}\frac{f_{t,d}(k_1+1)}{k_1\left(\frac{\boldsymbol{l_d}}{\boldsymbol{l_{avg}}}\right)+f_{t,d}}\cdot\omega_t$
- BM25:
  - While BM11 removes the problem with assuming equal document length in practice it has problems
    - Long relevant documents are getting too much dampening
    - Short irrelevant documents are getting too much boosting
  - To control the amount of correction we introduce b (often set to 0.75)
    - $\sum_{t\in q}\frac{f_{t,d}(k_1+1)}{\boldsymbol{k_1(1-b)}+k_1\left(\frac{l_d}{l_{avg}}\right)\boldsymbol{b}+f_{t,d}}\cdot\omega_t$
    - This expression represents the famous BM25 ranking function, which gives state-of-the-art results

---

Language modeling for retrieval

- Approaching the probabilistic information retrieval problem from a different perspective. Instead of modeling document probability given the query we model the query probability given the document.
- Probabilistic modeling of language
- A sentence t is a vector of terms $< t_1, \ldots, t_n >$
- Ideally, the probability of sentence t is:
  - $p(t) = p(t_1)\cdot p(t_2|t_1)\cdots p(t_i|t_1 \cdots\ t_{i-1})\cdots p(t_n|t_1,\cdots,t_{n-1})$
    E.g. for $t = \{one, ring, to, rule\}$ we have
    $$p(t) = p(one)\cdot p(ring|one)\cdot p(to|one, ring)\cdot p(rule|one, ring, to)$$
  - In practice the sparseness problem makes this intractable
- Unigram language models:
  - Solve the sparseness problem by completely ignoring conditioning
  - Probability of sentence t under the unigram model is:
    - $p(t|M) = p(t_n)\cdots p(t_i)\cdots p(t_1)$
  - The probabilities that define the model are estimated from a collection:
    - $p(t_i) = \frac{n_i}{n_T}$
    - n_i - number of times term t_i occurs in the collection
    - n_T - total number of term occurrences in the collection
  - Example:
    - d_1: “Frodo and Sam stabbed orcs.”
    - d_2: “Sam chased the orc with the sword.”
    - d_3: “Sam took the sword.”

t_i	Frodo	Sam	orc	chased	sword
$P(t_i)$	1/16	3/16	2/16	1/16	2/16

- <u>Bigram language models</u>:
  - We simplify the conditionals by leaving only the previous word
  - A better approximation of reality than unigram models
  - Probability of sentence t under the bigram model:
    - $p(t|M) = p(t_n|t_{n-1}) \cdots p(t_i|t_{i-1}) \cdots p(t_1)$
  - The probabilities that define the model are estimated from a collection:
    - $p(t_i|t_{i-1}) = \frac{n(t_{i-1},t_i)}{n(t_{i-1})}$
    - $n(t_{i-1}, t_i)$ - number of times bigram (t_{i-1}, t_i) occurs in the collection
    - $n(t_{i-1})$ – number of times term t_{i-1} occurs in the collection
  - Example (documents same as above):

t_{i-1}, t_i	Frodo, chased	the, sword	the, orc
$P(t_i|t_{i-1})$	0	2/3	1/3

- <u>Query likelihood model</u>:
  - Given a document collection D and a query q
  - A language model M_d **is built for each document**
  - Documents are scored according to the probability $P(q|M_d)$
  - A **unigram** language model of document d can be estimated by dividing the number of times term i occurs in d by the number of terms in d
    - $P(t_i|M_d) = \frac{n_{i,d}}{n_d}$
  - **Bigram** language models are rarely applied to individual documents due to sparsity problems
- <u>Smoothing language models</u>:
  - Models we've considered so far give **probability 0** to queries which **contain terms that do not occur** in the document
  - We can prevent this by using smoothing techniques
  - Smoothing adds a small probability under the model even to unseen words
  - **Laplace smoothing** - Adding a fixed small count (e.g. 1) to all word counts (even the unobserved ones) and renormalizing to get a probability distribution
    - $p'(t_i|M_d) = \frac{n_{i,d}+\alpha}{n_d+|V|\alpha}$
    - adds an artificial count α for each possible word in our vocabulary V
    - Laplace smoothing assumes all unseen words are equally likely!
  - **Jelinek-Mercer smoothing** rather builds a language model M_d of the entire document collection, and interpolates:
    - $p'^{(t_i|M_d)} = \lambda p(t_i|M_d) + (1-\lambda)p(t_i|M_D)$
    - Words absent from a document will still get some probability mass from the right term. However, the amount each word gets will vary depending on their likelihood in the collection as a whole
  - A **Dirichlet smoothed unigram model** is given by:
    - $p(t_i|M_d) = \frac{n_{i,d}+\mu P(t_i|M_D)}{n_d+\mu}$
    - Each word gets an artificial extra count
      - How much, depends on its probability in the collection
      - Considers length (the shorter the document the more weight is given to global knowledge from M_D)
    - Laplace smoothing is a special case of Dirichlet smoothing