Document ranking.

Ulvi Bajarani, Student ID 20539914

e-mail: ulvi.bajarani01@utrgv.edu
Class: CSCI 6370.01 - Web Search Engines and Info Retrieval
Semester: Summer I 2020

July 2, 2020

Abstract

The using of document ranking models are the important part of the Information Retrieval process. In the provided paper, the author provides the classical document ranking models and some of their extensions. In the final section, the author provides his own method.

1 Ranking models

1.1 The Boolean model and its extensions

The Boolean model and its extensions based on the set theory and Boolean Algebra. In the Boolean model, the terms either exist (in other words, have the weight 1), or don't exist (in other words, have the weight 0). The query expressions based on the Boolean Algebra and its operators: *and*, *or*, *not*.

The advantages of the Boolean Model are:

1. The easy implementation in the systems and clear formalization behind the expressions;
2. The easy calculation of the document similarity to the query. If every conjunctive component of the query is equal to the conjunctive component of the document, the documents are similar to each other, and sim$(d_j, q) = 1$. As a result, the user gets the desired results.
3. It is fast in small set of documents.

The disadvantages of the Boolean Model are:

1. The retrieved documents are not ranked. In this case, it is impossible to know which document has the highest relevance.
2. The results might have the only documents that is exactly match with the query expression. In other words, the partial matches are not retrieved by the Boolean model. As a result, the term might appear once or several times, which doesn't change the result of query. In addition to this, the users usually don't write complex queries.
3. Irrelevant in the large set of document. Firstly, it might be very slow. Secondly, it might retrieve either very few documents or a lot of documents.

One of the possible extensions is **Extended Boolean Model** [1]. It uses the Vector Space Model elements, such as weights and vectors, and combines them with the Boolean Model, solving some problems described in the disadvantages. In the Extended Boolean Model, each term k_x of the document d_j has the weight $w_{x,j}$:

$$w_{x,j} = \frac{f_{x,j}}{max_x f_{x,j}} \times \frac{IDF_x}{max_i IDF_i}$$

where $f_{x,j}$ is the frequency of term k_x in the document d_j, max$_x f_{x,j}$ is the maximum frequency in the document d_j, IDF_x is the Inverse Document Frequency of term k_x, and

$\max_i IDF_i$ is the maximum Inverse Document Frequency in the document.

In the two-dimensional space with two terms, there are two possible cases:

1. for distinctive query, $q_{or} = k_x \vee k_y$, The starting point of $k_x - k_y$ coordinate plane be taken as $(0; 0)$, because it is the least interesting one. The similarity can be calculated as

$$sim(q_{or}, d_j) = \sqrt{\frac{x^2 + y^2}{2}}$$

2. for conjunctive query, $q_{or} = k_x \wedge k_y$, The starting point of $k_x - k_y$ coordinate plane be taken as $(1; 1)$, because it is the most interesting one. The similarity can be calculated as

$$sim(q_{and}, d_j) = 1 - \sqrt{\frac{(1-x)^2 + (1-y)^2}{2}}$$

The additional advantage of Extended Boolean Model is that p-norm distances of document vectors could be used instead of Euclidean Distance, where $1 \leq p \leq \infty$. In that case, for the disjunctive queries $q_{or} = k_1 \wedge_p k_2 ... \wedge_p k_m$ and conjunctive queries $q_{and} = k_1 \vee_p k_2 ... \vee_p k_m$, the similarity might be calculated as

$$sim(q_{or}, d_j) = \left(\frac{x_1^p + x_2^p + ... + x_m^p}{m} \right)^{\frac{1}{p}}$$

$$sim(q_{and}, d_j) = 1 - \left(\frac{(1-x_1)^p + (1-x_2)^p + ... + (1-x_m)^p}{m} \right)^{\frac{1}{p}}$$

When $p = 1$, the similarity can be verified as

$$sim(q_{or}, d_j) = sim(q_{and}, d_j) = \frac{x_1 + x_2 + ... + x_m}{m}$$

When $p = \infty$, the similarity can be verified as

$$sim(q_{or}, d_j) = max(x_i)$$
$$sim(q_{and}, d_j) = min(x_i)$$

Another possible approach is creating the termsets and checking the vocabulary-set for every possible subset of the terms [2]. In this case, the problem is the possible number of subsets, which is equal to 2^t, where t is a number of terms in the termset. To avoid this, n-termsets might be used, where all $n - 1$ termsets are frequent. In other words, the number of documents $\mathcal{N}_i$ where the set occurs is greater or equal to a given threshold. The calculation of the ranking for the termsets are similar to the vector model terms:

$$\mathcal{W}_{i,j} = \begin{cases} (1 + \log_2 \mathcal{F}_{i,j}) * \log_2 \left(1 + \frac{N}{\mathcal{N}_i}\right) & \text{if } \mathcal{F}_{i,j} > 0; \\ 0 & \text{if } \mathcal{F}_{i,j} = 0. \end{cases}$$

where $\mathcal{F}_{i,j}$ is the raw frequency of termset S_i, $\mathcal{N}_i$ the number of documents where the termset S_i occurs, N is the total number of documents.

As a result, the cosine similarity is calculated by

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q_j}}{|\vec{d_j}| \times |\vec{q_j}|} = \frac{\sum_{S_i} \mathcal{W}_{i,j} \times \mathcal{W}_{i,q}}{|\vec{d_j}| \times |\vec{q_j}|}$$

where $\vec{d_j} = \{\mathcal{W}_{1,j}, \mathcal{W}_{2,j}, ..., \mathcal{W}_{i,j}\}$ and $\vec{q} = \{\mathcal{W}_{1,q}, \mathcal{W}_{2,q}, ..., \mathcal{W}_{i,q}\}$

Another possible approach is using the Fuzzy set theory [3], where the term relationships are defined with the correlation matrix, where an element $c_{i,l}$ is defined by

$$c_{i,l} = \frac{n_{i,l}}{n_i + n_l - n_{i,l}}$$

where n_i is the number of documents containing the term k_i, n_l is the number of documents containing the term k_l, $n_{i,l}$ is the number of documents containing both the terms k_i and k_l.

Hence, the degree of membership $\mu_{i,j}$ of the document $d_{i,j}$ that receives the value in the interval $[0; 1]$, is calculated by

$$\mu_{i,j} = 1 - \prod_{k_l \in d_j} (1 - c_{i,l})$$

1.2 The Vector Space model and its extensions

In the Vector Space Model, every query and every document is represented as the vector $\vec{d_j} = \{w_{1,j}, w_{2,j}, ..., w_{i,j}\}$ and $\vec{q} = \{w_{1,q}, w_{2,q}, ..., w_{i,q}\}$, where for each i and j, $w_{i,j}$ are the non-binary and non-negative weights of each term, and $w_{i,j} > 0$.

In this case, the cosine similarity between a query and a document might be calculated as

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q_j}}{|\vec{d_j}| \times |\vec{q_j}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

There are several advantages of the Vector Model:

1. Term-weighting scheme improves retrieval quality of queries;
2. As a result, it provides the partial matching strategy that retrieve the documents approximately matching query conditions.
3. The sorting of documents by the value of cosine similarity to the query
4. It provides the length weight normalization, which helps to build-in the reliable ranking.

However, the Vector Model has own disadvantages:

1. In the huge collections, the calculation of rankings are slow.
2. The partial matching could provide the results with low weights.

One of the extension of the Vector Model is the Generalized Vector Model [4]. Unlike the classical Vector Model, which is usually restricted to the fact that for each pair of index vectors $\vec{k_i}$ and $\vec{k_j}$, $\vec{k_i} \bullet \vec{k_j} = 0$, the Generalized Vector Model assumes that the index term vectors are composed of smaller components derived from the collection by hand. The main idea is to creating the unique vector for the document which doesn't exist in other documents. In this case, from the vocabulary $V = \{k_1, k_2, ..., k_t\}$, the 2^t miniterms are created:

$$\begin{aligned} & (k_1, k_2, k_3, \ldots, k_t) \\ m_1 &= (\ 0,\ 0,\ 0,\ \ldots,\ 0) \\ m_2 &= (\ 0,\ 1,\ 0,\ \ldots,\ 0) \\ m_3 &= (\ 0,\ 0,\ 1,\ \ldots,\ 0) \\ & \vdots \\ m_{2^t} &= (\ 1,\ 1,\ 1,\ \ldots,\ 1) \end{aligned}$$

In this case, $on\,(i, m_r)$ defines if the term k_i is in m_r, and

$$on\,(i, m_r) = \begin{cases} 1 & \text{if } k_i \text{ is in } m_r; \\ 0 & \text{if } k_i \text{ is } \textbf{not} \text{ in } m_r. \end{cases}$$

The index term vector $\vec{k_i}$ is calculated as

$$\vec{k_i} = \frac{\sum_{\forall_r} on\,(i, m_r)\, c_{i,r}, \vec{m_r}}{\sqrt{\sum_{\forall_r} on\,(i, m_r)\, c_{i,r}^2}}$$

where $\vec{m_r}$ is the unit vector, and

$$c_{i,r} = \sum_{d_j \ |\ c(d_j)=m_r} w_{i,j}$$

1.3 The Probabilistic model and its extensions

The idea of The probabilistic model is to create the probabilistic description of the ideal answer set that matches to the query [5]. For this reason, the results of cooperation between users that decide the relevancy of a document and the system might be used. For the query q, the ratio $\frac{P(d_j \text{relevant-to q})}{P(d_j \text{non-relevant-to q})}$ is calculated. In this case , $w_{i,j}$ in $\vec{d_j} = w_{1,j}, w_{2,j}, ..., w_{i,j}$ is equal either to 0 if the word doesn't exist in the document d_j or to 1 if the word exists in the document d_j. Thus, the similarity might be calculated as

$$sim(d_j, q) = \frac{P(R \mid \vec{d_j}, q)}{P(\overline{R} \mid \vec{d_j}, q)}$$

The advantage of the Probabilistic model is that the documents are ranked with the decreasing order of the probability. However, the Probabilistic model has several disadvantages:

1. It requires the user intervention to define what documents are relevant;

2. The weights document vector based on the fact if the keyword exists or not. As a result, like Boolean Model, it doesn't count how many times the word is used in the document.

3. The length of a document is not normalized.

One of the possible extension is to use the Bayesian Network Models [6]. It is based on the relationships between the nodes of acyclic graph, where arcs describes the strength between nodes with random values. For the G Bayesian network containing x_i with the set of parents Γ_{x_i}, any function $F_i\left(x_i, \Gamma_{x_i}\right)$ that satisfies

$$\sum_{\forall x_i} F_i\left(x_i, \Gamma_{x_i}\right)$$

and

$$0 \leq F_i\left(x_i, \Gamma_{x_i}\right) \leq 1$$

could be the influence function.

2 The method of the author

Analysing the models, the author provides the Elo Model. The idea is to compare the documents against each other to define the weight of documents. The first parameter is the number of terms in documents, which should be compared. The second parameter is the length of the document. By comparing all documents in all terms, the Elo of document d_j term should be calculated by the formula

$$Elo_{k_{i,j}} = f_{i,j} + \log_2 L_j$$

where $Elo_{k_{i,j}}$ is the Elo of the document d_j in the term k_i, $f_{i,j}$ is the frequency of the term in the document d_j and if $f_{i,j} = 0$, the $Elo_{k_{i,j}}$ is equal to 0, L_j is the total number of terms (length) of the document d_j. In this case, the total Elo of the document is equal to

$$EloTotal_{d_j} = \sum_{i=1}^{k} Elo_{k_{i,j}}$$

In the Boolean *or*, *not* expressions, the highest values of term ELOs of the document satisfying the term should be retrieved. If ELOs in the terms are equal, the highest EloTotal should be retrieved. In the *and* queries, if there are several documents satisfying the query, the average ELO is calculated by:

$$EloAverage = frac(\sum_{j=1}^{k} \frac{EloTotal_{d_j}}{j})$$

,and k is equal to the number of consecutive $EloTotal_{d_j}$ involved in the *and* operation. As it might be seen, it is always k > 2.

The possible advantages are:

1. The document with the highest count of a term will be retrieved.
2. In the equal conditions, the documents with either higher diversity, or higher length will be retrieved.

The possible disadvantages are:

1. The bitmap of each document comparing to the total vocabulary should be calculated.
2. The ELO Calculation in the huge systems might be time-consuming.

References

[1] Gerard Salton, Edward A Fox, and Harry Wu. Extended Boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.

[2] Bruno Pôssas, Nivio Ziviani, Wagner Meira Jr, and Berthier Ribeiro-Neto. Set-based vector model: An efficient approach for correlation-based ranking. *ACM Transactions on Information Systems (TOIS)*, 23(4):397–429, 2005.

[3] Yasushi Ogawa, Tetsuya Morita, and Kiyohiko Kobayashi. A fuzzy document retrieval system using the keyword connection matrix and a learning method. *Fuzzy sets and systems*, 39(2):163–179, 1991.

[4] SK Michael Wong, Wojciech Ziarko, and Patrick CN Wong. Generalized vector spaces model in information retrieval. In *Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 18–25, 1985.

[5] Stephen E Robertson and K Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146, 1976.

[6] Howard Turtle and W Bruce Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems (TOIS)*, 9(3):187–222, 1991.