Duke University

Information Searching Models

Xianjue Huang

Math of the Universe

Hubert Bray

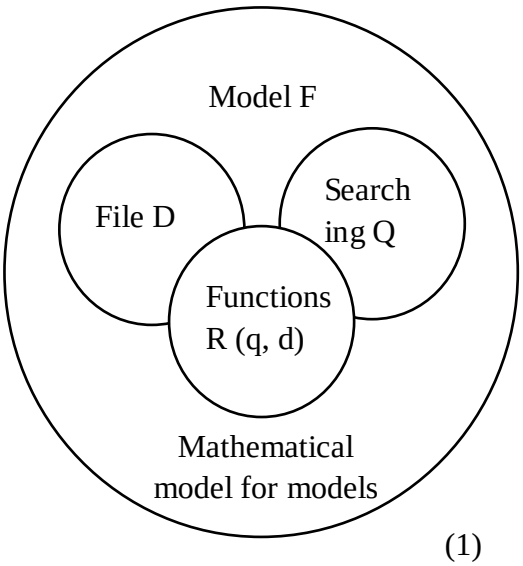
24 July 2017

Introduction

Information searching happens in our daily life, and even before the computers were invented. In the past time, people search medical recipe, and cooking advice on books or documents. However, things got little bit different when computers and the internet came up: after computers and the internet came to stage, millions of electronic books, documents are available in the computer drives and online. Then, Searching engines like Google appeared to help people to find what they want online, after all, if a person need to find something relate to his need, he will not be willing to go through those millions things right? Also, if he/she wants to search something about math, he/she probably does not need to find things about literature, right? This paper mainly focus on different models which focus on contents that had been invented to search things online.

Basic and Relative Concepts of Information Searching

First of all, I want to state the concept of information searching models: they are “containers” that have the structure and algorithms that can be used to show the electronic files and the “searching”, and then compute them with the similarity between them. So



(1)

basically it is modeling the “degree of association” between the files and the searching request. On the graph above, I showed the mathematical model of these models, there, the “Model F” is the container for constructing relations between files and searching requests; “File D” is a set of files that made up with a series of “Index Terms” (will be explained in the next paragraph); “Searching Q” is a set of searching requests; “Functions R (q, d)” are sorting functions that arrange the “degree of association” between each D and Q. (1)

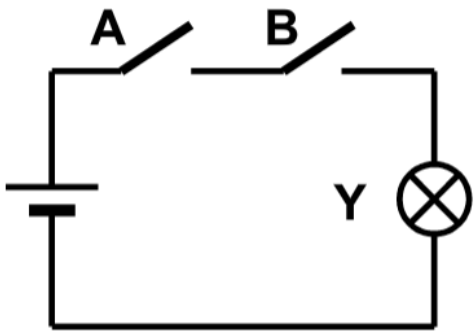
Then, the ideas for the model is that every files can be seen as a set of many “Index Terms”, which are always being made by different words. To further understand about “Index Terms”, “Key Words” can be treated as one kind of “Index Terms”. Moreover, in some models, each “Index Term” has its own weights. For example, if the “Index Terms” of a file are “cat”, “food” and “store”, but the file mostly talk about cat’s behavior, and then its eating habit the weight might be “cat” > “food” > ”store”. (1)

Last but not the least, there are different approaches on those algorithms. In the 1960s, the “Boolean Model” came to stage, and then, in the year 1968, the “Vector Space Model” by Salton, and then, in the year 1976, the invention of the “Probability Model” by Robertson came, and then the “Logic Model” by Rijsbergen in 1986. Some take cares about the contents of the files, and the others focus more on the structure, and in this paper I would talk about those focus on contents. (1)

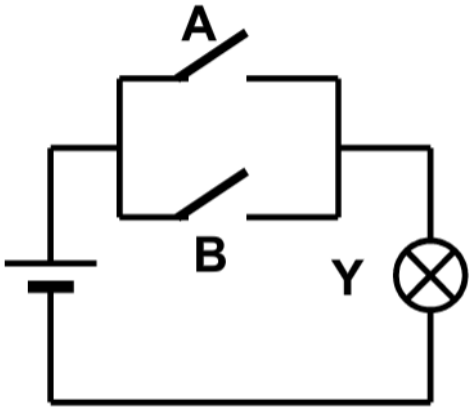
Boolean Model

Remember that I said in the introduction part that some models have "Index Terms" with weights? This "Boolean Model" does not have weights, which means this type of model only concern about whether the "Index Terms" show up or not show up in the files. This model is made base on the "Set Theory" and the "Boolean Algebra". Basically, you can think this model as the easiest model being used.

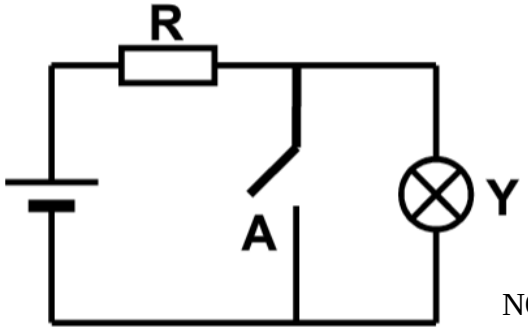
So much for introducing, I would like to talk about content now. The "Searching Q" will be "Searching Words" combined by "and", "or", and "not". For example if I search "cat animal", the "Q" will look like "cat and animal". Then, here, I would like to explain how the "and", "or" and "not" work. Let's see the three picture of electric circuit below:



AND



OR



NOT

(1)

Note: you can think the battery as the searching request, and “A” and “B” are the condition to check “Index Terms” and documents that may be contain the files. Lastly, the light bulb will glow when the conditions are matched.

The picture with label “AND” has two switches, “A” and “B”. As the common sense in the physics field, for a circuit with a charged up battery, a light bulb and switches, the circuit must be a closed circuit for the light bulb to glow, meaning that both switch A and B must be switched off. So, condition “A” and “B” must be both matched to meet the searching request. Suppose I searched “cat food” with the condition “and”, then for files which contains both “Index Terms” “cat” and “food” will meet the searching request, and will finally show up on the results, so you can be sure that the files will talk about cat and food inside the same file. Then about the picture with label “OR”, it changed the structure from “AND” a little bit, so that this time, no matter switch “A” or switch “B” is closed, or both closed, the light bulb will glow. Suppose again I searched “cat food” but with the condition “or”, then every file that contains either “cat” or “food” will show up, so this time, you might not be sure whether this file will only talk about the cat, the food or both. However, the picture with label “NOT” makes things totally different. So this time, when the switch is closed, the light bulb will not glow, completely opposite right? So, in this situation, everything without “A” will be shown in the result. Again, considering the “cat” as a searching request, then you can be completely sure that no files shown will talk about the cat. (1)

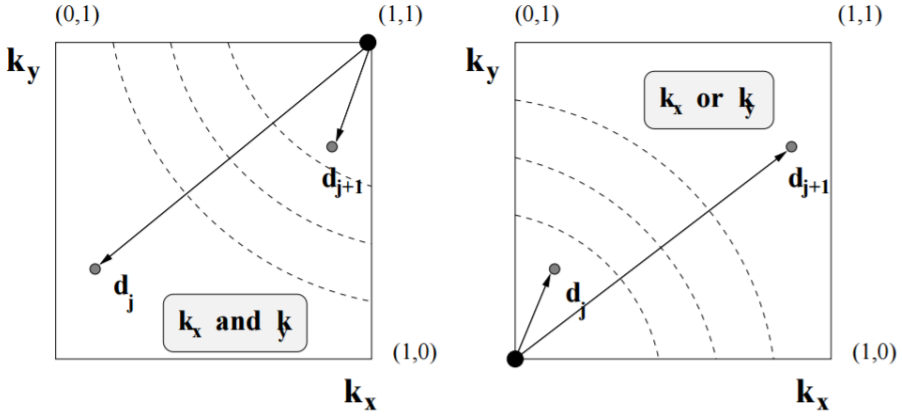
$$\text{sim}(d_j, q) = \begin{cases} 1 & \text{if } q_{cc} \in q_{dnf} \text{ and } d_j = q_{cc} \\ 0 & \text{otherwise } q_{dnf} \end{cases} \quad (2)$$

The formula listed above shows how the “Boolean Model” determines whether the file “dj” is relevant to searching request “q”, where “q_{dnf}” is the “disjunctive normal form” of “q”, and “q_{cc}” is any of the “conjunctive components” of “q_{dnf}” and if sim(d_j, q) = 1, then they are relevant. (2) (4)

This kind of model has its advantages and its disadvantages: first, about its advantages, it is widely used by commercial systems because this model makes things neat and elegant with concepts that can be understood easily. Then, it is also self-protected, meaning that it can lower the expectation from the users, because user will know that if the search results do not match their thought, it is their problem that did not give a good searching. So in this way, the responsibility for the searching is not in the hands of the developer and runner, but in those of the users. However, its disadvantage is obvious, and it is this kind of searching model can only determine the files as relevant or irrelevant to the searching request, meaning there is no ranking for the files. Moreover, sometimes, people might find it really difficult to convert their information in to a Boolean expression. (1)

Extended Boolean Model

However, “based on a critique of a basic assumption in Boolean algebra, Salton, Fox, and Wu introduced the “Extended Boolean Model””. (3)



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

$$sim(q_{and}, d) = 1 - \sqrt{\frac{(1-x)^2 + (1-y)^2}{2}} \quad (3)$$

The picture above shows the basic idea and the basic formula when the “Extended Boolean Model” is working where x and y are the corresponding term weights. The formula to calculate the weights is listed below, where Idfx is inverse document frequency (4):

$$w_{x,j} = f_{x,j} * \frac{Idf_x}{max_i Idf_i} \quad (4)$$

Note: the outcome of this formula gives the x and y in the last case.

By calculating using the formula above, the similarity can be determined, so that each file will be able to be ranked. For more complicated examples, we use the logic and formula below:

1) As a generalized conjunctive query, $q_{or} = k_1 \vee^p k_2 \vee^p \ldots \vee^p k_t$
   the similarity of q and dj is defined as: $sim(q_{or}, d_j) = \sqrt[p]{\frac{w_1^p + w_2^p + \ldots + w_t^p}{t}}$
2) As a generalized disjunctive query, $q_{and} = k_1 \wedge^p k_2 \wedge^p \ldots \wedge^p k_t$
   the similarity is defined as: $sim(q_{and}, d_j) = 1 - \sqrt[p]{\frac{(1-w_1)^p + (1-w_2)^p + \ldots + (1-w_t)^p}{t}}$

(4)

This kind of model improved the “Boolean Model” so that now, the “Index Terms” of the files are having their weights. Also, this model improved the correctness of searching, because now, users can use “and” to present the relationship between each terms in the query, and can also use “or” to link up the synonyms. Furthermore, because now the model is concerned about the weights of the terms, it can give a rank on the files, so that the best fit file always comes at the first. (3)

Vector Space Model

In this kind of model, suppose every “Index Term” in a file is a vector, and each vector presented as one dimension of a space, then the set of these vectors will define a space of the file. Then, every file can be treated as a vector too, so that the set of these files will define a space of the files.

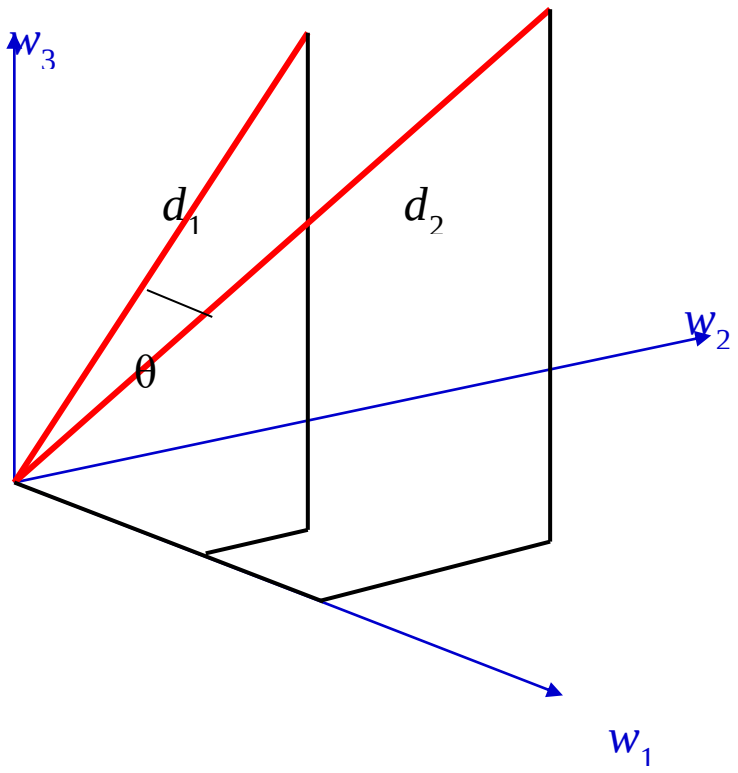
Let us discuss this model by using an example. First, let us define three document, and then let us find out their similarity, and the whole idea would apply to the calculation of the similarity between files and searching requests.

d_1 *ant ant bee*

d_2 *dog bee dog hog dog ant dog fox*

d_3 *cat gnu dog eel fox* (2)

The hypothesis is that the similarity measuring might take firstly, the length of the documents; secondly, the number of identical terms; thirdly, the time each term appears in document set; finally the degree of unusualness of the term into account. Then, the idea came out that the similarity between two documents or the searching request and the files is determined by the angle between their vectors in the term vector space. (2)



(2)

The formula to find the angle is given below, where x1 and x2 are the vectors of the documents, $|\mathbf{x}_1|$ and $|\mathbf{x}_2|$ are the length of x1 and x2 (2):

$$\text{sim}(\mathbf{x}_1, \mathbf{x}_2) = \cos(\theta) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{|\mathbf{x}_1|\,|\mathbf{x}_2|}$$

$$|\mathbf{x}|^2 = x_1^2 + x_2^2 + x_3^2 + \ldots + x_n^2$$

	ant	bee	cat	dog	eel	fox	gnu	hog
d_1	1	1						
d_2	1	1		1		1		1
d_3			1	1	1	1	1	

dot product

	d1	d2	d3
d1	2	**2**	0
d2	2	5	2
d3	0	**2**	5

consin angle

	d1	d2	d3
d1	1	**0.63**	0
d2	0.63	1	0.40
d3	0	**0.40**	1

(2)

The first chart below the formula shows the words those documents contained, so if the documents contained, say, “Index Term” “ant”, they will get a 1 at the column of the “ant” on the chart, otherwise, they will have nothing. Then, the chart with title “dot product” is the value they have for the x1•x2 part for the formula. Then, with the second formula, the $|\mathbf{x}_1|$ and $|\mathbf{x}_2|$ can be easily determined. Now, with all the value calculated, the similarity between each documents can be determined. So, suppose I search the key

word “ant”, “dog”, “hog”, “fox” and “bee”, the rank for the document will be d2>d1>d3. (2)

Then, as I said, each word has its own weight, so the situation for the chart will change. The weight of each word depends on its “Term Frequency”, and each time the word repeat itself, the weight of it increase. So basically the more the term repeat itself, the heavier the weight will be. However, the words like “a”, “and”, and “the” will not be taken into account. (2)

	ant	bee	cat	dog	eel	fox	gnu	hog
d_1	2	1						
d_2	1	1		4		1		1
d_3			1	1	1	1	1	

dot product

	d1	d2	d3
d1	5	3	0
d2	3	20	5
d3	0	5	5

consin

	d1	d2	d3
d1	1	0.3	0
d2	0.3	1	0.5
d3	0	0.5	1

(2)

See, the weights now looked very much different from the last one. Remember that the similarity between d1 and d2 was larger than that of d2 and d3? But now they switched.

Then this time, if I searched “ant”, “bee”, and “dog” for a searching request, then the charts will change into below (2):

	ant	bee	cat	dog	eel	fox	gnu	hog
q	1	1		1				
d_1	2	1						
d_2	1	1		4		1		1
d_3			1	1	1	1	1	

(2)

	d_1	d_2	d_3
q	$3/\sqrt{15}$ 0.77	$6/\sqrt{60}$ 0.77	$1/\sqrt{15}$ 0.26

$$\text{sim}(q,d_j) = \frac{\mathbf{q} \cdot \mathbf{d}_j}{|\mathbf{q}| \, |\mathbf{d}_j|}$$

(2)

The calculating ideas are the same, but now we added one document as a vector of the searching request “q” (see the picture below). So now, if we rank the three document, they will be d1=d2>d3. (2)

To summarize, any text can be seen as a vector in the space, and then the model will rank documents based on the cosine similarity of the documents to the query, or searching request. However, this model is based on the supposing that each index term is independent from each other, so taking the relevance between “Index Terms” into account will be a drawback.

(5)

Probabilistic Model

This Model is quite different from the three above. In this model, many assumption has been applied. The first assumption is that the model believes the probability it calculates only depends on the formation of the searching request and that of the files. Then, the model assumes that the first result might not be perfect so that it need some interaction with the users. Thirdly, it assumes that the weight of the "Index Terms" are also binary, meaning only 0 or 1. Lastly, it assumes that the formations are independent from each other. The essential idea of this searching model is that given a searching request, then there will be a set of the documents that is relevant to the request, and will be a set that is irrelevant, and the rules it applies is the Bayes' theorem.(1)

So much for the basic ideas, now I want to discuss about its internal math: let "R" be the relevant set of documents (the initial result), $\overline{R}$ be the probability that the document dj is not relevant to the searching request q. Then, let $P(R \mid d_j)$ be the complementary set of the "R", $P(\overline{R} \mid d_j)$ be the probability that document dj is relevant to the request q. So the similarity of the document and the searching request is defined by the formula listed below: (1)

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

(1)

So remember that I mentioned before this model applied the Bayes theorem? So base on the theorem, the formula for defining the similarity becomes this: (1)

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

(1)

Then, based on the assumption that every term is independent to each other, the formula will change to where $P(k_i \mid R)$ is a constant value for each "Index Term" ki, and the value is often being chosen as 0.5, whereas $P(k_i \mid \overline{R})$ will be calculated by the value of "ni/N" where ni is the count of the files that contains the "Index Term" ki, and N is the count of all the files in the document sets (the second formula takes the logarithm form of the first formula and assume to neglect all the factors that never changes). (1)

$$sim(d_j, q) \sim \frac{(\prod_{g_i(d_j)=1} P(k_i \mid R)) \times (\prod_{g_i(d_j)=0} P(\overline{k}_i \mid R))}{\prod_{g_i(d_j)=1} P(k_i \mid \overline{R})) \times (\prod_{g_i(d_j)=0} P(\overline{k}_i \mid \overline{R})}$$

$$sim(d_j, q) \sim \sum_{i=1}^{t} w_{iq} \times w_{ij} \times \left(\log \frac{P(k_i \mid R)}{1 - P(k_i \mid R)} + \log \frac{1 - P(k_i \mid \overline{R})}{P(k_i \mid \overline{R})} \right)$$

(1)

Because that the first few results set might not be the best fit set, then, the creator announced that the model will utilize the idea of recursive computation

where Vi is the count of the documents that contains ki in the document set, and V is the number of the files in the document set: (1)

$$P(k_i \mid R) = \frac{V_i}{V} \qquad P(k_i \mid \overline{R}) = \frac{n_i - V_i}{N - V} \tag{1}$$

So, by doing the recursive computation, the two values above will get more and more accurate so that the final result will be more and more accurate.

The best thing about this model is that in theory, the files will automatically being arranged by the relevance probability. However, this model will need to first cut all the files into two sets, one with relevant documents, and one with irrelevant documents, which will increase the work load of the computer. (1)

Conclusion

In this paper, I discussed about four models that make information searching come to reality, the “Boolean Model”, “Extended Boolean Model”, “Vector Space Model”, and “Probabilistic Model”. Among these four models, the “Boolean Model” is the oldest, and it can only treat the files with level relevant or irrelevant based on the Boolean theory of 0 and 1. Then the “Extended Boolean Model” improved the “Boolean Model” by making the “Index Terms” marked with weights so that now the files can be ranked with the degree of relevance. Thirdly, the “Vector Space Model” treated each file

and query as collections of vectors, and the similarity between them are determined by the angle between vectors. Lastly, the “Probabilistic Model” will work well when its basic assumptions work. However, these are not the only kinds of the searching models, and every searching engine might utilized different searching models ideas with same time, and I believe with the development of the world, more and more powerful models will be invented to help people searching.

Bibliography

陆铭 Richard Lu, 2011, 讲座-信息检索模型研究 Lecture-Information Searching Models Discussion, doc88.com, July 22, 2017, http://www.doc88.com/p-40795531397.html (1)

闻立杰 (Professor of Tsinghua University more Information : http://ise.thss.tsinghua.edu.cn/~wenlijie/) , January 26 – January 30, Field – Trip-Lecture on Information Searching Models, Tsinghua University School of Software, January 26 2017. (2)

Addison Wesley, 2006, Lecture on Modern Information Retrieval Chapter2 Modeling page 17, July 22, 2017, http://homepages.dcc.ufmg.br/~nivio/cursos/ri08/transp/slideschap02b.pdf (3)

Wikimedia Foundation, Extended Boolean Model, Wikipedia, July 23 2017, https://en.wikipedia.org/wiki/Extended_Boolean_model

Wikimedia Foundation, Disjunctive Normal Form, Wikipedia, July 23 2017, https://en.wikipedia.org/wiki/Disjunctive_normal_form (4)

Jaime Arguello, February 13 2013, Lecture - Vector Space Model, July 23 2017, https://ils.unc.edu/courses/2013_spring/inls509_001/lectures/06-VectorSpaceModel.pdf (5)