Technical Report of Ranking Model for Product Search Engine

Kevin Hu/Yingfeng Zhang ©B5M.com

January 6, 2014

Contents

1 Problem Description

The problem we meet is a general problem for search engine: for a given query, which document is most relevant. For an example, for query "cellphone", both cell phone battery and cell phone's title all have the key words, how come the machine knows which one should be on top in the result set. Figure 1 has shown the classic model to solve that problem. The objective function is as following:

$$\arg\max_D \{\prod_i^N P(T_i|D,Q)P(D)P(Q)\}$$

$$\Rightarrow \arg\max_D \{\prod_i^N P(T_i|D)P(T_i|Q)P(D)\},$$

st. Q and D are independent.

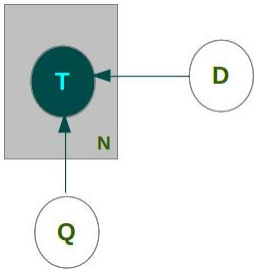

Figure 1: Classic generation model. T for term; D for document; Q for query.

It's a generation model which means that terms are generated by document, and we want to get a document who maximize the likelihood of generating the query terms.

The classic BM25 ranking model can be deduced from it. TF represents the P(T|D) part, and IDF represents the P(T|Q) part. Unfortunately, for a product search engine and our current situation, we are lack of NLP data resources to calculate these two parts P(T|D) and P(T|Q). Anyway, we figure out a new way to calculate the ranking score using other model to avoiding our weakness.

2 A New Ranking Model

2.1 An Assumption

Actually, a product can be described as a multiple dimensions(attributes) vector, dimensions like title, brand, model type, color, category and so on. And every dimension has a different importance for describing a product. For the above problem, "cellphone" probably match an important dimension like category for cellphone, and match less important dimension for cellphone battery except that they all match title. So, we made an assumption, for a given query, the ranking score of a document related to importance of matched dimension.

2.2 Formalize and Prove

We use a probabilistic graphic model to formalize the assumption as Figure 2.

A document generates M attributes and a category, then, for every attribute, it generates N terms which also can be generated by a query. The total probability of the entire model can be represents as following:

$$\prod_{i}^{N}\prod_{j}^{M} P(T_{i,j}|A_j,Q)P(A_j|D)P(C|D)P(Q|C)P(D)P(Q) \tag{1}$$
$$st.\ T_{i,j} \in Q, A_j \in D, \text{Q and A are independent.}$$

(1)

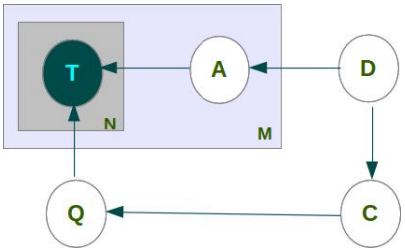

Figure 2: The new generation model. T for term; D for document; Q for query; A for attribute; C for category.

The graph has 5 variables(the circle with capital char in it).

M is the number of attribute for a document.

N is the number of terms for an attribute.

P(D) which stands for the quality of the document which can be measured by, say, number of attributes, the date of the document and so on. This is actually an very important parameter, and more detail on section 2.4

P(C|D) which is a constant for a particular document, 0 or 1.

P(Q|C) which is from manually mapping, it's a constant too, 0 or 1.

P(A|D) which can be interpreted as importance of an attributes(dimension) to a document.

P(T|A) which can be interpreted as the similarity between a term and a attribute. It is a portion of the length of the entire string.

P(T|Q) which is a length portion of the entire length of a query.

2.3 The Estimation of Parameters

The model is presented as a probabilistic model, but the relation between variables which are called parameters are not calculated using probablistic method, like, using maximum likelihood estimation. Firstly, we manipulate the formular 1 into following:

$$\begin{aligned}\log Rank(D) = &\sum_{q_i \in Q, a_j \in D} \log(P(q_i|a_j)P(a_j|D)P(q_i|Q)) \\ &+ \log P(C|D) + \log P(Q|C) + \log P(D),\end{aligned} \tag{2}$$

Q composes of $q_i; a_j$ is attributes in document; Q and $a_j$are independent.

To interprete the mathmatical expression vividly, here's a pseudo code for rank score calculation.

Data: Q ← query, D ← document
Result: Ranking score of a document
r ← 0;
for *q in Q* **do**
  for *a in D* **do**
    r ← Length(q)/Length(a); // for $P(q_i|a_j)$;
    r ← r * Importance(a, D); // for $P(a_j|D)$;
    r ← r * Length(q)/Length(Q); // for $P(q_i|Q)$;
  end
end
if *Q and D belongs to the same category C* **then**
  // This ensures doc has the same category
  // as query alway rank ahead of docs that doesn't belong to it;
  r ← r + LargeConstant;
end
r ← r + Quality(D);

Algorithm 1: Pseudo code for ranking score

The hard point here is to calculate the attribute importance to a document. We made another assumption here. The more important the attribute is to a product, the less probably the attribute exits in every product category. So, we use information entropy to simulate the attribute importance as following. And the result turns to be not bad.

$$Importance(a) = -\sum_i P(a|C_i)\log_{10} P(a|C_i),$$

a is for attribute; C is for category; this calculate the importance of an attribute.

With the equation above, we can get the global importance of an attribute independent to document, which we use to estimate $P(a_j|D)$.

2.4 The Estimation of Document Quality

By then, the parameter $P(D)$ hasn't been well explained. It's actually a weight aggregation of several measurements: number of comment, number of sale, date to market. The equation is as following:

$$\begin{aligned} &Quality(D) = \sum_i w_i(\exp(x_i/C_i) - 1)/(\exp(x_i/C_i) + 1), \\ &\text{st. } w_i \in [0,1], x_i/C_i \in [0,10] \end{aligned} \tag{3}$$

Typically, we set w_i for comment number to a large portion like more than 0.6.

We can see from both models above, $P(D)$ is just as important as relevance between query and document. For a general search engine, $P(D)$ composes of page-rank score

and the quality of the site, for which, search engine like baidu as I know has a very complicated calculation for the quality of the site, baidu doen't index the site if the quality is blow a threshold.

$P(D)$ is a reflection of users' data request. The calculation of $P(D)$ should vary from different domains, since in different domain, user's request is very different. Within the domain of product search engine, for different categories, the measurements is quite different, though they has some measurements in common, like, number of sale, number of comment and time to market and so on.

Given by query log, there're two category user search most, clothing and 3C, and query for clothing is much more than 3C, it's even more than 90%. For 3C product, the completeness of description of attributes, price comparison are very important but not for clothing. For clothing, the picture quality and quantity are at the first place and it will be great if it has buyer show. So, we'd better separate clothing from other products to optimize $P(D)$ and the service form.

3 Use B5T's Log Data to Enhance Ranking

3.1 Another Assumption

By the illustration of new ranking model, we can see it's not perfect. $P(q_i|a_j)$ and $P(q_i|Q)$ is just portion of the entire string. So, we plan to use another hyper parameter.

B5T's log data is a golden mine. In the log, we have track of a typical Internet user's action when they use Internet services. From the log, we can have a good sense of what user place value on, like, why they use baidu, what they buy on Taobao, and so on. Besides we get closer to user, user's input is also a valuable NLP data resources. As users' query on taobao, they can tell us which term is more important than others. We make another assumption that query generates terms, the more frequent the term exist in query, the more important the term is. Use the PGM to illustrate it as Figure 3.

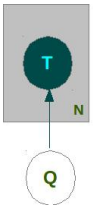

Figure 3: The query-term model. T for term; Q for query.

The calculation of term weight is as following:

$$Weight(T_i) = P(T_i) = \sum_Q P(Q)P(T_i|Q),$$

st. T is for term; Q is for query.

3.2 Another New Ranking Model

With help of term weight, we can add one more parameter to the generation model, which is a very critical hyper parameter, a N-dimension vector that illustrate the global importance of every term in the dictionary.

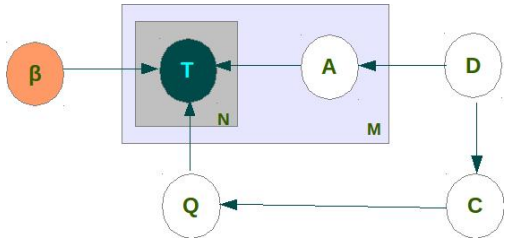

Figure 4: The model with term weight. T for term; D for document; Q for query; A for attribute; C for category.

$$\prod_{i}^{N}\prod_{j}^{M} P(T_{i,j}|A_j, Q; \beta)P(A_j|D)P(C|D)P(Q|C)P(D)P(Q)$$
$$\Rightarrow \log Rank(D) = \sum_{q_i \in Q, a_j \in D} \log(P(q_i|a_j;\beta)P(a_j|D)P(q_i|Q;\beta)) \quad (4)$$
$$+ \log P(C|D) + \log P(Q|C) + \log P(D),$$
$$st.\ T_{i,j} \in Q, A_j \in D, \text{Q and A are independent.}$$

P(T|Q;β) for which we calculate the portion of weight of the term in all the weighted query terms;

P(T|A;β) for which we calculate the portion of weight of the term in all the weighted terms in an attribute;

The pseudo code has a new version as following:

Data: Q ← query, D ← document
Result: Ranking score of a document
r ← 0;
// weight(q) and weight(term) are given by β ;
for *q in Q* **do**
  for *a in D* **do**
    r ← weight(q)/sum(weight(term in a)); // for $P(q_i|a_j;\beta)$;
    r ← r * Importance(a, D); // for $P(a_j|D)$;
    r ← r * weight(q)/sum(weight(term in Q)); // for $P(q_i|Q;\beta)$;
  end
end
if *Q and D belongs to the same category C* **then**
  // This ensures doc has the same category ;
  // as query alway rank ahead of docs that doesn't belong to it;
  r ← r + LargeConstant;
end
r ← r + Quality(D);

Algorithm 2: Pseudo code for ranking score with term weight