# Text Retrieval and Search Engines

Jiahao Zhao
Computer Science
Emory University
Decatur, GA, US
jiahao.zhao@emory.edu

ABSTRACT

**This paper briefly reviews and introduces text retrieval and its application, especially text retrieval and search engines.** Text retrieval is an example of information retrieval. A brief definition is that Text retrieval models or algorithms can respond to users' queries and return a list of relevant docs or web pages. The score of relevance ranks those documents. Moreover, a feedback system based on users' feedback or simulated feedback is necessary because we need feedback to optimize our models. Such feedback usually is placed after the text retrieval model ends.

**Text retrievals are frequently used and play a crucial role in our daily life.** The most representative example would be the search engine. You enter a set of strings to find the online web pages you want to get. Another example can be a website recommendation system, i.e., you browse a product on Amazon, then Amazon will recommend similar products to you in the next few days. Also, YouTube has its recommendation system. When you watch a video, YouTube will recommend similar videos to you. All those examples belong to text retrieval. We call it the pull mode for users actively searching for some documents feedback, such as search engines. On the contrary, for those users who need to passively get input from the system, such as the YouTube recommendation system, we call it to push mode.

Note that in the pull model, there are two different cases. The first case is that the user knows what they are searching for. The user can directly query the system. Another possibility is that the users don't even know the word they are exploring. The algorithm needs to let users browse a list of documents or web pages. Then, based on those content, the model can predict what you want to search.

Unlike the information retrieval on the database, those data are formatted and well organized, so we can grab the information much easier by using some query functions such as NoSQL or SQL. On the contrary, the information retrieval on plain text could be much more complex because it is empirically defined. Suppose we need to search the information "health lunch recipe for loose weight," which includes several important information. This person wants to get a healthy recipe. The beneficial purpose specifies "losing weight." More than that, this person would like to get lunch in such a recipe, neither dinner nor breakfast. Different information has different importance, and various combinations of the level of importance might change the search results. For example, when we are ranking the search results, should we emphasize "losing weight," "lunch recipe," or "healthy recipe"?

Another problem is the same word or sentence might imply different semantics. Such as "play," we usually say play means have fun with something, like "play video games" "play with my brother," but it also has different semantics, like "play around with your code, see if you can find some bugs," which means workaround with your code, it has nothing to do with have fun with something. So, we want our model can differentiate different semantics. Moreover, there are trillion documents/web pages out on the internet. Besides the accuracy, instead of letting the user wait a long time to return the results, we also want to place the prevalent document to users fastly.

There are even more problems in this field. We want to build up reliable and efficient text retrieval methods for users. All those problems are needed to be solved.

## INTRODUCTION

**Formulated problem definition:** We have three elements for this problem there are Query (the input), Documents (the input), and ranking models (the algorithm). Lastly, we should return a list of relevant docs or web pages to users.

1 Query:

$$Q = q_1 q_2 q_3 q_4 \ldots$$

Where Q is the query as a whole, and each $q_i$ is just each substring in that query. For example, "health lunch recipe for loose weight," where Q is the entire sentence, and $q_1$ is "health," $q_2$ is "lunch" and so on.

2 Documents :

$$D = d_1 d_2 d_3 \ldots$$

Where D is the Universe of document on the internet, or some databases based on the system. Each $d_i$ is just each document or webpage in the universe.

Ranking models/algorithms:

$$f(Q, D) = d, d \in D$$

The function f will take two inputs: query and documents, and the output list of documents d, subsets of the D in the universal.

There are several models to do text retrieval. Two of the most important models are the vector space and probabilistic models.

The Vector space model is built upon the idea of "similarity." Given a query Q and documents in the universe, we need to traverse through each document, find how each is compared to, then see the compatibility between. Lastly, based on the compatibility, we rank those related documents and then return them to users.

But there are some drawbacks to this model. When users are trying to search some specific information on the internet, they tend to include prevalent words like "the" "a," "what," "and," etc., when the model is trying to match those words with documents, almost every document includes those words. Still, we cannot include such documents every time. There are several solutions to this problem, we can ignore those kinds of words or normalize such words by dividing such terms by the number of occurrences of those words, or we can also weigh such words by multiplying some constant. For a more detailed mathematical explanation, you can see it in the following few sections.

Another model is the probabilistic model. The big idea of this model is conditional probability. Given a pair of query q and document d, what is the likelihood that the relevance of 1 and d is equal to 1? Namely:

$$f(1,d) = P(R = 1|q,d)$$

By doing this, we traverse through each q and d, then see the relevance, and sum up all the relevance divided by the total number of documents that q appears in d. A different approach might be more accessible. It is called "Query likelihood". It says that given a document, what kind of query are the users likely to ask? Which is strongly relevant to the inverse way of thinking about the problem. For more detail, I will discuss it in the following sections**.**

**Outline:** This paper will explain the vector space model and the probabilistic model in the following sections in more detail. Besides that, we need to introduce feedback methods and show how it works with our text retrieval model. Lastly, to show how the entire thing can be applied to the search engines and other systems.

**Possible Outcomes**: The paper aims to show you two lower level text retrieval models. There are **vector space model and probabilistic model (query likelihood)**. specifically, we want to show the mathematical expression and implications of the two models, and what kind of problem raises during developing the models, and how we are going to solve it. And in the future, modern search engines need personalized search results for various users. Also, we want to integrate all different features such as pull mode and push mode into one information management system. All those open questions still need to be considered in the future.

THE VECTOR SPACE MODEL

One of the possible solutions to this problem is the vector space model. The basic idea is simple. As we mentioned before, the query is a list of words, and clearly, the candidate documents are also formed by a list of words. So, we set a Boolean value for each term in the query. If it appears in the query, we set it as the 1, otherwise 0—same thing for documents. Then we take the doc product operation with respect to the query and documents. Namely

$$Q = q_1 q_2 q_3 q_4 \ldots$$

$$D = d_1 d_2 d_3 \ldots$$

For each q_i and d_i:

$$q_i \in \{0, 1\}$$

$$d_i \in \{0, 1\}$$

Then the similarity of the document can be represented as

$$f(Q,D) = \sum_{1}^{N} x_1 y_1 + x_2 y_2 + \cdots + x_N y_N$$

For example, the query is "the healthy recipe for the Italian food", and the documents are D1 = "……the health food…." D2 =" …recipe for Italian food…", D3 = "…. health recipe … in Italian...". Clearly, by comparing the number of the same words, the score for D1 is 3, D2 is 4, and D3 is 3.
But suppose let us consider the document D4 = "…the…for the….", then the score is 3, which has the same similarity score as D1 and D3, but if we take a look at the content, D4 doesn't likely to be the result that users want, because the words "the" " a" "for" are very frequently used words in real life.
So, the question arises, how can we ignore such words when doing the text retrieval? To do so, we need to introduce a text weighting model called TF/IDF mode. TF is the model such that we take the number of words divided by the size of the document. Whereas IDF is

$$IDF = \lg\left(\frac{total\ number\ of\ document}{number\ doc\ contain\ that\ word}\right)$$

The more frequent the word, the lesser the IDF weight of those words. In that case, by multiplying the IDF weight for each term, then we can get normalized scoring.

$$f(Q,D) = \sum_{1}^{N} x_1 y_1 IDF(x_1) + x_2 y_2 IDF(x_2) + \cdots + x_N y_N IDF(x_N)$$

However, another problem arises. If we consider this document D5 = "…food…food…food…food…food…food…food…food…".
Even if we lower those weights for those most frequent words by plugging the document into that function, this document doesn't seem likely to be the ideal result because we are overly weighting

some keywords multiply times and ignoring words like "health" "Italian."

So, we need to restrict the appearance of frequent keywords somehow. The word "food" appears frequently is good, but we don't want to overly award it, which means we want to increase the weight of the D5, but we don't want to reward D5 overly. We need an increase to be a limited function to weigh each word. In this case, we use the weighting function

$$f = \frac{(k+1)x}{k+x}$$

Where k is the constant, x is the frequency of words x in document D.

To sum this up the formula becomes

$$f(Q,D) = \sum_{w \in D \& w \in Q} c(w,q) * \frac{(k+1)c(w,d)}{c(w,d)+k} * IDF\ (W)$$

Here we call it BM25 ranking function.

THE PROBABLISTIC MODEL

Let's start with a question, given a query and group of documents, how likely this document is the document the users want. This problem is roughly equivalent to giving a document, how likely the users ask the query Q. The formalized expression will be

$$f(Q,D) = \mathrm{P}(\mathrm{R}=1|\mathrm{D},\mathrm{Q})$$

Where D is the document, Q is the query, R is the relevance as binary expression. If the query and document are relevant ,then R=1, otherwise R=0. And the definition of this relevance formula is

$$\mathrm{P}(\mathrm{R}=1|\mathrm{D},\mathrm{Q}) = \frac{C(Q,D,R=1)}{C(Q,D)}$$

We call this model a query likelihood model. It means that given the query Q and documents D, the likelihood Q and D are relevant is equal to the overall number of occurrence of Q and D in which they are relevent (number of users think the D is relevant to Q) divided by all number of occurrences for QD pair. For example, we have (q_0, d_0, R=1), (q_0, d_0, R=0), (q_0, d_0, R=1), (q_0, d_0, R=1). Then the overall occurrence of q_0 and d_0 is 4 and 3 times the case that they are relevant to each other, so the query likelihood is 75%.

But this model is not that perfect since we cannot go through all documents for each query. What about those unseen documents? In this case, we need to change our model respectively. Take a look at the equation. We are conditioning the pair the query Q and document D. If we only condition the document, say here is the document and users think it is relevant, how likely the query Q will be asked? This way is easier to be evaluated. So the function will be

$$f(Q,D) = \mathrm{P}(\mathrm{Q}|\mathrm{D}\ \&\ \mathrm{R}=1)$$

But how to evaluate the relevance of the document? We need to introduce a language model called the unigram language model. Since a list of words forms a document, the overall probability of the document can be derived by taking the multiplication on each word of probability.

$$D = \{w_1 w_2 \dots \dots w_k\}$$

then

$$P(D) = P(w_1)P(w_2) \dots \dots P(w_k)$$

Assume each word are independent to each other.

Then, each word will have a different probabilistic for the topics of the different documents. For example, the term "chicken" more frequently appears in recipe or cooking documents, and lesser appears on documents related to real state or house decoration. So, the probability for each word will be the occurrence of the word divided by the size (number of words) of the document. Namely

$$P(w) = \frac{C(w,\ D)}{|D|}$$

the scoring function is

$$f(Q) = \prod\nolimits_{w \in Q} P(w) = \prod\nolimits_{w \in Q} \frac{C(w,\ D)}{|D|}$$

However, a similar problem arises, remember in the vector space model, some words like "the" "a" "and" "of," etc., appear so frequently that they will influence the score of the documents. Those words are so frequent that they will appear on any document. So, we need to normalize those terms in the probabilistic model.

The idea is that those words have their universal probability, which is that the word will appear in all documents. Namely, the number of words that appear in the universal divide is the summation of the size of each document in the universe. Then all we need to do the divide the raw probability of the word by the universal probability of the word..

$$P(w|d)/P(w|U) = \frac{C(w,\ D)}{|D|P(w|U)}$$

where

$$P(w|U) = \frac{number\ of\ occerance\ of\ w\ in\ the\ universe}{sum\ of\ size\ of\ documents\ in\ the\ universe}$$

However, there is another problem, since we are summing up the probability for each word, we said there might be some words that appear so frequently that we need to do some normalization, but in the case that there are some words that don't show up, what we should do this time?

For example, the query is "newly healthy Italian recipe for lunch." The documents are d1 = "healthy recipe …… in Italian …… for lunch …… released newly at ……", d2 = "healthy recipe …… for lunch …. in Italian", d3 = "new Italian recipe…. healthy … lunch options….". D1 will have the highest score because it contains all keywords in the query. But something interesting will happen to d2 and d3. In d2, the word "newly" never shows up, which means the score for d2 will become 0, since the C ("newly," d2) = 0, so the numerator is 0, the entire equation ends up with 0. Still, even "newly" doesn't show up, based on the content of d2 (indeed, d2 is not as relevant as d3), but d2 is still worth getting recommended to the users. Also, the similar issues happen in d3, d3 includes the word "newly," but it doesn't have the word "for," then the score for d3 is also 0, so d3 is even more wronged, because "for" is a word that not worth to mention, d3 should have closer importance as d1.

We need to consider the background scoring for all words to solve this problem. Even if the word "newly" doesn't show up, we still need to count it in its background probabilities. For example, the likelihood of the word "newly" in all documents appearing is 0.002. Then we use it as the scoring of the function. Another thing is that, remember we are multiplying the score for each word. Still, even the word's score is not 0, since we are multiplying the background scoring of the phrase, and likely the background score of the word is very low, so the overall score of the entire document will be significantly dragged down! So instead of multiplying those scores, we can take the summation of the score, then each word will be counted, and a sudden significantly low score will not affect the overall score that much.

So, the probabilistic model will be something like:

$$f(Q,D) = \sum_{w} C(w,q) \log \mathrm{p}(w|d)$$

Still, we take logarithm is to avoid too small values, C (w, q) = 0 if the word w is not in the query, which makes sense because we are scoring with respect to the documents.

Future direction

As we mentioned before, the most important text retrieval application is a search engine. Nowadays, the development of search engines is dramatically based on users' needs, so a customized search engine is one of the important future features. For example,

Special group of users

When users put the same query term, the expected documents might be different from a different group. For example, the query "graph." When a college student in the mathematics department is trying to search this term, the results are more likely to be "graph theory," "bipartite graph," "graph coloring," "Turan graph," and so on very math-tensed academic terms. But when a police officer searches the term "graph," a visualized city crime graph might be the proper documentation to be returned. So, the different group contains different target ideal results.

Individually customized search engines

Consider the query "healthy recipe." The word "healthy" might have multiple meanings. For example, before the user searched "healthy recipe," they also searched "losing weight" in the earlier time, then the recipe for low calories should be returned to users. So, each individual might have a different preference.

Timely customized search engines

Even the same person searched "healthy recipe," but it means differently at different times. Say five years ago, this person was overweight and wanted to lose weight, so a "healthy recipe" that includes those recipes might help lose weight. But right now, after losing weight, this person wants to become a fitness person, then now this person is trying to search "healthy recipe," those high protein recipes should be get returned. The search engine needs some time-based learning mechanism to progress users' preferences.

**Combination of Pull and Push mode**

When a user searched something, for example, "baby doll." Then the recommendation system in google webpages should push some relevant search results to users. In the case of "baby doll," the model can infer that this user might get a baby recently. Then the webpage system should push diapers or dried milk to this users, or something related to tiny baby to this user in the future search

## REFERENCES


[1] D. L. Lee, Huei Chuang and K. Seamons, "Document ranking and the vector-space model," in *IEEE Software*, vol. 14, no. 2, pp. 67-75, Mar/Apr 1997, doi: 10.1109/52.582976..

[2] Jones, K.S., Walker, S. and Robertson, S.E., 1998. A probabilistic model of information retrieval: Develepment and STatus. Department o(f Information Science, City University, London, 74.

[3] Howard R. Turtle, W. Bruce Croft, A Comparison of Text Retrieval Models, *The Computer Journal*, Volume 35, Issue 3, June 1992, Pages 279–290, https://doi.org/10.1093/comjnl/35.3.279.