Text Retrieval and Search Engines

Natural Language Processing

Natural language processing is any system used by a computer to extract information from human languages. There are various ways to do this and they vary greatly in the amount of information they return. The simplest way to extract information is to break down a text into a series of individual words. A more successful but much more difficult approach is to break down text into **Parts of Speech** and try to determine the meaning of the text as a whole.

Bag of Words

A bag of words is generated by taking a text and breaking it down into each individual word associated with its frequency in the text.

Pull Mode

Pull mode text retrieval occurs when the user generates the retrieval request, for example with a **search engine**. Pull mode can further be divided into different types of retrieval requests such as those generated by **queries** as opposed to those generated when **browsing.**

Push Mode

Push mode text retrieval occurs when the retrieval system initiates the request, for example in the case of a **recommender system.**

Text Retrieval

Text retrieval or **search technology** is the process of retrieving relevant documents from a collection in response to a query. It is a subset of **Information Retrieval** and even though text retrieval is often referred to as information retrieval the latter is much broader and includes such diverse systems as retrieving audio or video files from file collections.

Text retrieval differs from database retrieval in that the information being searched and the query to retrieve it are often either unstructured or ambiguous in some way. This makes text retrieval a more empirically defined problem relying on the judgement of users to determine accuracy.

Vocabulary

The vocabulary of a language *V* is the set of all the words in the language.

$$V = \{w_1, w_2, \dots, w_N\}$$

Query

The query *q* is the set of words to look for in a document.

$$q = \{q_1, q_2, \dots, q_N\} \, where \, q_j \in V$$

Document

A document *d* is defined to be equivalent to the set of words in the document.

$$d_i = \{d_{i1}, d_{i2}, \dots, d_{iL}\} \, where \, d_{ij} \in V$$

Collection

A document collection *C* is the set of documents to be searched.

$$C = \{c_1, c_2, \dots, c_M\}$$

Relevant Documents

The relevant documents *R* are contained in the set of documents returned as a result of the text retrieval process.

$$R(q) \subseteq C$$

Since text retrieval tasks can often be resource intensive it is often useful to concentrate on the creation of *R'(q)* which is an approximation of *R* instead of trying to generate *R* in its entirety.

Absolute Relevance

The absolute relevance of a document is a binary classifier that determines if a document is relevant or not. Documents selected by this method can be described in the following way:

$$R'(q) = \{d \in C | f(d,q) = 1\}, where\ f(d,q) \in \{0,1\}$$

Relative Relevance

The relative relevance of a document is used to rank its significance relative to other documents. It can be expressed in the following way:

$$R'(q) = \{d \in C | f(d,q) > \theta\}, where\ f(d,q) \in \Re$$

Where *f(d, q)* is a **ranking function** and θ is a user-defined cut off point.

Probability Ranking Principle

The probability ranking principle (Robertson, 77) states that returning a ranked list of documents in descending order of probability that a document is relevant to the query is the optimal strategy under the following two assumptions:

1. The utility of a document (to a user) is independent of the utility of any other document.
2. A user would browse the results sequentially.

Similarity Based Model

This text retrieval model ranks documents higher the more similar they are to the query. One well-known example of the similarity based model is the **vector space model** which states that the ranking function *f* approximates the similarity of the document *d* to the query *q* in vector space:

$$f(d,q) = similarity(d,q)$$

Probabilistic Model

The probabilistic text retrieval model uses classic probabilistic arguments to determine whether or not a document is likely to be relevant.

$$f(d,q) = p(R = 1|d,q), where\ R \in \{0,1\}$$

Probabilistic Inference Model

The probabilistic inference text retrieval model determines the likelihood that a given document is consistent with the query.

$$f(d,q) = p(d \rightarrow q)$$

Axiomatic Model

The axiomatic text retrieval model requires the ranking function *f(d, q)* to satisfy a set of constraints.

Popular Text Retrieval Models

The following models are all popular and known to perform almost equally well:

- Pivoted length normalization

- BM(Best Matching)25
- Query likelihood
- PL2

The most popular method is BM25.

Term Frequency

Term frequency is the number of times a **term** q from the query occurs in a document d.

$$TF = count(d, q)$$

Document Frequency

Document frequency is the number of documents in a collection that contain the term *q.*

Bit Vector

The simplest vector space model uses a **bit vector** to evaluate the similarity between a query and a document. For a query *q* and a document *d* the value of a point in vector space is either 1 or 0 depending on whether *q* is contained in d.

$q = (x_1, \dots x_N)$
$d = (y_1, \dots y_N)$

$xi, y_i \in \{0,1\}$
$1: word\ Wi\ is\ present$
$0: word\ Wi\ is\ absent$

$$f(q,d) = Sim(q,d) = q.d = x_1y_1 + \cdots + x_Ny_N = \sum_{i=1}^{N} x_iy_i$$

Term Frequency Vector

This is an improvement over the simple bit vector model. The term frequency vector uses the count of a word w_i in a query and in a document to determine the similarity between a query and a document.

$q = (x_1, \dots x_N)$
$d = (y_1, \dots y_N)$

$x_i = count\ of\ word\ w_i in\ query$
$y_i = count\ of\ word\ w_i in\ doc$

$$f(q,d) = Sim(q,d) = q.d = x_1y_1 + \cdots + x_Ny_N = \sum_{i=1}^{N} x_iy_i$$

Inverse Document Frequency

The inverse document frequency is a weight introduced into the ranking calculation to penalize popular terms in the document that may otherwise skew the result.

$x_i = count\ (w_i, q)$
$y_i = count\ (w_i, d) * IDF(w_i)$

$$IDF(W) = log\left[\frac{M+1}{k}\right]$$

Where *M* is the total number of documents in the collection and *k* is the total number of documents containing w_i (the document frequency).

The **ranking function incorporating term frequency and inverse document weighting** becomes:

$$f(q,d) = \sum_{i=1}^{N} x_iy_i = \sum_{w \in q \cap d} c(w,q)c(w,d)\ log\left[\frac{M+1}{k}\right]$$

BM25

$$f(q,d) = \sum_{i=1}^{N} x_i y_i = \sum_{w \in q \cap d} c(w,q) \frac{(k+1)c(w,d)}{c(w,d)+k} log\left[\frac{M+1}{k}\right]$$

Document Length Normalization

Longer documents are more likely to be returned by a text retrieval system due to the large number of words they contain. This can skew the results but any weighting system needs to avoid over-penalization so that documents with more significant content are not penalized unfairly.

The solution to this is a **"pivoted" length normalizer** using the average document length as a pivot (b is a user defined parameter whose value determines the significance of the penalty due to document length).

$$normalizer = 1 - b + b\left[\frac{|d|}{avdl}\right] \; where \; avdl \; is \; the \; average \; document \; length$$

Pivoted Length Normalization VSM (Singhal et al, 1996)

$$f(q,d) = \sum_{w \in q \cap d} c(w,q) \frac{ln\left[1 + ln[1 + c(w,d)]\right]}{1 - b + b\frac{|d|}{avdl}} log\left[\frac{M+1}{df(w)}\right]$$

Okapi BM25 (Robertson & Walker, 1994)

$$f(q,d) = \sum_{w \in q \cap d} c(w,q) \frac{(k+1)c(w,d)}{c(w,d) + k\left(1 - b + b\frac{|d|}{avdl}\right)} log\left[\frac{M+1}{df(w)}\right]$$