Seminar: Techniques for implementing main memory data bases

Text analysis: TF-IDF

Dao Thuy Ngan Tran

23. Oktober 2017

TF-IDF has been a reliable model for term weighting in information retrieval. Full-text searches in data bases is among others an application of TF-IDF. In order to optimize query results, adjustments to the original model were made over the years, be it to encounter undesired effects of long documents or to determine the optimal result set of the query.

1 Overview

One of the state-of-the-art methods in information retrieval is the BM25 model, which has its roots in probabilistic modeling and has been proven to be a reliable means. Robertson outlined in 2004 that the mechanism behind the BM25 concept actually can be expressed as a TF-IDF weighting scheme, proving its subtle relevance in the field [4].

In 1972, Jones first introduced a new heuristic for weighing terms in information retrieval. As of then, term frequency was already known as a term weighing scheme. Jones outlined how the growing size of a document collection also reduced the specificity of a term. Hence, the inverse document frequency was conceived to diminish the weight of terms that occur in many documents throughout the collection. The term frequency and the inverse document frequency were combined to create a back then new weighing term heuristic [1].

Using the base TF-IDF model, other adjustments were introduced in more recent models in order to counter effects which appear through the difference between document collections, one of them, by Jiaul H. Paik, viewed in detail in this paper [2]. The same model also shows a way to apply TF-IDF to information retrieval by proposing a document ranking formula.

Tata et al. [3] additionally demonstrate an algorithm to use TF-IDF and cosine similarity to determine selectivity of a full-text search.

Finally, this work will deal with a new approach to TF-IDF, which does not introduce a modification to the algorithm itself but rather the target document

collection used for weighing the query terms [5].

2 Standard TF-IDF

Stephen Robertson [4] pools all variants of TF-IDF into a class of TF-IDF based weighting schemes. This means that all those schemes use the idea of TF-IDF as a base and look for ways to improve different aspects of the full-text search like the term weight itself or properties of the document collection.

2.1 Term Frequency

Term frequency (TF) as a measure of retrieving information is a commonly used heuristic. It denotes a term weighing means depending on the number of times a term occurs in a document. The simplest variant of this weight is the raw count of the term per document [4]. A term's weight therefor increases with every single occurrence in each document of the collection. This measure is to be generated for each term in the query and each document in the target document collection the search is performed on.

The C++ Code 1 shows the implementation the base TF model.

```
map<string, int> index(string fileName){
  map<string, int> index;
  map<string, int>::iterator it;
  ifstream file;
  file.open(fileName, ifstream::in);
  if(file.is_open()){
      string word = "";
      while (file.good()) {
          if(isalnum(file.peek())!=0){
              word += tolower(file.get());
          } else {
              it = index.find(word);
              if( it !=index.end()){
                  it->second++;
              } else {
                  index.insert(pair<string, int>(word, 1));
              }
              word = "";
              file.get();
          }
      }
    file.close();
    return index;
}

int termFrequency(string term, string documentName){
  string filename = collectionPath + documentName;
  map<string, int> indexDoc = index(filename);
  map<string, int>::iterator it = indexDoc.find(term);
  if(it!=indexDoc.end()){
      return it->second;
  } else{
```

```
33          return 0;
34      }
35  }
```

Code Snippet 1: `Term Frequency`

2.2 Inverse Document Frequency

Karen Spärck Jones first observed and published this heuristic in 1972 after observing the specificity of search terms in large document collections. She realized that if a term is found in too many documents, it is unlikely relevant for the specific query. It is thus to be assigned a lower *inverse document frequency* (IDF) value than terms which are less frequently found throughout the document collection [1]. The classification as a heuristic is used because there was no mathematical way to prove the inverse document frequency as right or wrong back then. Still, the application proved it to be nowadays essential in information retrieval. The base equation is shown in equation 1, with t_i as the term at index i in our query, N as the number of documents in the collection and n_i as the number of documents in which t_i occurs in at least once.

$$idf(t_i) = log(\frac{N}{n_i}) \tag{1}$$

Robertson published a paper in 2004, acknowledging the indispensability of Jones' finding and producing a link between Shannon's theory of information and communication. Based on Shannon's entropy formula, the information amount of an individual message in a channel of communication can be interpreted as in equation 2, with I_i as the information amount of the individual message i and p_i as the probability for the individual message to be transmitted in the channel.

$$I_i = -log(p_i) \tag{2}$$

Using equation 2, Robertson and Spärck Jones created the relevance weighting model (RSJ) in 1976, based on different assumptions about the number of documents relevant to our search in the collection. In the end, the most likely setup for our search is that we do not know how many documents will be relevant, leading to the RSJ depiction as in equation 3. w_i denotes the weight of query term i while N and n_i again denote the number of documents in the collection and the number of documents query term i occurs in, respectively.

$$w_i = -log(\frac{N + 0.5}{n_i + 0.5}) \tag{3}$$

IDF is hereby a a simplified version of the RSJ weight [4].
The following code snippet 2 shows the implementation of the IDF.

```
1 double idf(string term, vector<string> collection){
2     float documentFrequency = 0;
3     for(int i = 0; i < collection.size()-1; i++){
```

```
4           if(termFrequency(term, collection[i]) > 0){
5               documentFrequency++;
6           }
7       }
8       double result =
9       log((double) collection.size()/(double) documentFrequency);
10      return result;
11  }
```

Code Snippet 2: `Term Frequency`

3 Document Ranking using TF-IDF

In 2013, Jiaul H. Paik identified several drawbacks of the base TF-IDF scheme.

3.1 Two Factor Frequency

He realized that a linearly growing TF value does not properly reflect the relevance of a term and summarized the necessary properties of the TF value as

1. $F'_t(TF) > 0$
2. $F''_t(TF) < 0$.

Put into words, this means that the resulting TF value should grow with each increase of the raw TF count but that the growth factor itself should decrease at the same time.

Another problem with the raw TF count is the bias towards longer documents, which have an overall higher word count and therefor an increased probability for term term occurrence.

Paik additionally pursued the idea that the term frequencies within a document are dependent from another in order to determine the relevance of the query term within this document. This hypothesis puts a term frequency into relation to the average term frequency within a document, thus pinpointing whether a term is *relatively* relevant to the document compared to the other terms.

Combining this hypothesis with the requirements stated for the TF value, the formula 4 for the *Relative Intra-document TF* was conceived.

$$RITF(t, D) = \frac{ld(1 + TF(t, D)}{ld(1 + AverageTF(D)} \tag{4}$$

with t as the term for which we want to find the relative TF and D as the target Document. The logarithm is used to damp the otherwise linear growth of the TF and thus fulfill the requirement $F''_t(TF) < 0$. Effectively, this factor rewards terms and documents where a term seems to be essential to the documents content (a possible 'elite' term for a document, meaning that a document is äbouẗhe concept that is represented by a term [4]).

Equation 5 shows the TF variant *Length Regularized TF* constructed to counter the before mentioned undesirable effect of TF in long documents. This means that the longer a document is in relation to the average document length in a collection, the more it is punished.

$$LRTF(t,D) = TF(t,D) \cdot ld(1 + \frac{AverageDocumentLength(C)}{length(D)}) \tag{5}$$

with the same variables like the $RITF$ equation (4) and C as the document collection. In order to upper bound both factors to 1, the function $f(x) = \frac{x}{1+x}$ is used to result in the two equations $BRITF(t,D)$ (6) and $BLRTF(t,D)$ (7):

$$BRITF(t,D) = \frac{RITF(t,D)}{1 + RITF(t,D)} \tag{6}$$

and

$$BLRTF(t,D) = \frac{LRTF(t,D)}{1 + LRTF(t,D)} \tag{7}$$

.

The next step is combining those two TF factors into one by assigning the weights w and $1-w$ respectively as shown in equation 8. The weight w is chosen as $w = \frac{2}{1+ld(1+|Q|)}$ with $|Q|$ as the query length. This weight is chosen in order to increase with the query's length, since the document length should be punished less. This notion holds because long documents are more likely to contain all words of a query than shorter ones.

$$TFF(t,D) = w \cdot BRITF(t,D) + (1-w) \cdot BLRTF(t,D) \tag{8}$$

.

3.2 Modification of IDF

The regular IDF does not discriminate between documents that only mention a query term and documents in which a query is elite. To achieve this differentiation, the factor *average elite set TF* (AEF) is introduced as

$$AEF(t,C) = \frac{CTF(t,C)}{DF(t,C)} \tag{9}$$

.

with $CTF(t,C)$ as the total occurrence of the term in the document collection. Equation 10 finally shows the modified IDF as it is intended to be applied, with the AEF weight upper bounded to 1.

$$newIdf(t,C) = IDF(t,C) \cdot \frac{AEF(t,C}{1 + AEF(t,C)} \tag{10}$$

.

3.3 Ranking Model

After performing adjustments to the base models, the new TF-IDF values can be used to evaluate the relevance of each document in the document collection. The following C++-Code snippet 3 shows the implementation of the *Similarity Score*, which determines the document with the highest similarity to the query and therefor serves as the best match.

```
pair<string, double> bestMatch(vector<string> query, vector<string> collection){
   pair<string, double> documentScores("", 0.0);
   for(int j = 0; j < collection.size(); j++){
      for(int i = 1; i < query.size(); i++){
         double temp =
   tff(query[i], collection[j], query.size()) * tdf(query[i], collection);
        simScore += temp;
      }
      if(simScore > get<1>(documentScores) && !isnan(simScore)){
         get<0>(documentScores) = collection[j];
         get<1>(documentScores) = simScore;
      }
   return documentScores;
}
```

Code Snippet 3: `Similarity Score`

The similarity scores are computed by determining the TF-IDF values per document and query first and then adding up all TF-IDF values for the query up for one document which then yields the similarity Score [2].

4 Using TF-IDF for Selectivity Estimation

Other than using the before described similarity score based on cumulative TF-IDF values, Sandeep Tata and Jignesh M. Patel first used cosine similarity to determine the similarity between a query and a target document collection. In this case, the target document collection is an actual data base where each row represents a document. The cosine similarity is determined using dot product. Based on the the individual properties of each date base, the goal is to estimate the selectivity of TF-IDF, meaning how many rows will be approximately returned using the TF-IDF algorithm.

4.1 Summary Structure

The summary structure introduced by Paik et al. is used to determine the TF-IDF vector characteristical to the database. The idea behind this method is to obtain the probability distribution function within the rows of a data base. This can be viewed as the TF-IDF property of the *average* row in the database and hence denote the overall similarity of the data base to the given query.

It is possible to use cosine similarity for query/data base similarity if a query is viewed as a a vector with each entry as its own dimension. In order to properly

Row	**Token 1 weight**	**...**	**Token i weight**	**...**	**Token N weight**
1	w_1^1	...	w_i^1	...	w_N^1
...	...	...	...	...	...
j	w_1^j	...	w_i^j	...	w_N^j
...	...	...	...	...	...
M	w_1^M	...	w_i^M	...	w_N^M
	μ_1, σ_1, C_1	...	μ_i, σ_i, C_i	..	μ_N, σ_N, C_N

Tabelle 1: Summary Structure for Cosine Similarity

compare query and data base, the same dimensions have to be used for the TF-IDF vector and the data base vectors. After creating the vectors, the dot product is computed, yielding a float between 0 and 1 with 1 for identical vectors and 0 as orthogonal vectors.

Table 1 shows the concept of the summary structure as intended by Tata et al. consisting of the TF-IDF vectors with an additional row containing the values μ_i, σ_i and C_i. Hereby the data base has M rows while there are N distinct terms (tokens) in the whole database. The TF-IDF weight is determined for each token and every row, and is only > 0 if the token actually is contained in the row. The last row is defined as:

1. μ_i is the mean over all weights in column $i \neq 0$
2. σ is the standard deviation over all weights in column $i \neq 0$
3. C_i is the probability that column $i \neq 0$, namely the number of entries in this column divided by M. This can also be viewed as the document frequency.

4.2 Algorithm ES and EL

After obtaining the mean, standard deviation and document frequency, the cosine similarity between the TF-IDF vector the data base can be computed. Tata et al. have constructed two estimating algorithms, ES and EL, which both compute mean and standard deviation. This is done by calculating the dot product between the last row values of the data base and the TF-IDF query vector u as shown in 11 and 12.

$$\mu_{new} = \sum_{i=1}^{N} (\mu_i \cdot C_i \cdot u_i) \tag{11}$$

.

$$\sigma_{new} = \sum_{i=1}^{N} (\sigma_i \cdot C_i \cdot u_i) \tag{12}$$

.

Algorithm ES additionally adds the factors α and β which are scaling constants to account for the deviation from the actual mean and standard deviation of the data base. Determining those factors requires training sets and several runs of the algorithm.

Algorithm EL, on the other side, uses an additional step where all parameters are modified by a training function to improve the estimate.

Through experimentation, the approximate distribution of TF-IDF values was found to be closest to an inverse normal distribution. This distribution takes μ_{new} and σ_{new} as parameters to return the database specific distribution which is then used to provide an estimate for the query selectivity.

Even though both algorithms use components adjusted by training sets, algorithm EL proved to be slightly more accurate in test runs than algorithm ES in predicting a query's selectivity on a given data base [3].

5 Conclusion and Outlook

We can conclude that the TF-IDF model, even though conceived several decades ago, still holds great relevance in information retrieval research and has been subject to improvements in all aspects over the years. What started out as a heuristic from Jones proved to be a versatile and useful model currently implemented in data bases. Efficiency and resourcefulness becomes more important with every day, especially with main memory data bases.

One of the latest innovations in 2017 regarding TF-IDF is called TF-IDuF which does not modify the algorithm itself but the target document collection by determining TF-IDF values based on the personal document collection of the user. This gives a personal preference to the query, envisioning that each individual user is looking for different results for similar queries. The practical application intends to combine the TF-IDF weights based on the target document collection with the weights based on the personal collection [5].

This again shows how the TF-IDF model keeps evolving and adjusting to different purposes while the main idea still remains the same.

Literatur

[1] Karen Spärck Jones. A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL, Journal of Documentation, Vol. 28 Issue: 1, pp.11-21, 1972

[2] Jiaul H. Paik. A Novel TF-IDF Weighting Scheme for Effective Ranking, ACM, 2013

[3] Sandeep Tata, Jignesh M. Patel. Estimating the Selectivity of tf-idf based Cosine Similarity Predicates, SIGMOD Record (Vol. 36, No. 2), June 2007

[4] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for IDF, Journal of Documentation, Vol. 60 Issue: 5, pp.503-520, 2004

[5] Joeran Beel, Stefan Langer, Bela Gipp. TF-IDuF: A Novel Term-Weighting Scheme for User Modeling based on Users' Personal Document Collections, Proceedings of the iConference 2017, Wuhan, China, 2017