An Information Retrieval System for Product Reviews

Prajit Dhar and Janis Pagel
{dharpt,pageljs}@ims.uni-stuttgart.de

October 27, 2018

Abstract

We present a fully functional Information Retrieval system for 10,000 Amazon reviews. Two different types of systems were developed to evaluate the effectiveness of the retrieval systems: Vector space and probabilistic model systems. The effectiveness of the systems are evaluated for various metrics. While the probabilistic model systems performed better in the initial stages, the TF-IDF system from the vector space family of systems performed better as a whole. Future enhancements being considered are looked upon.

1 Introduction

Information Retrieval (IR) systems deal with retrieving relevant documents regarding a query and ranking these documents accordingly. Usually the collection of documents is of such a size that advanced techniques of optimizing the indexing and ranking of documents is inevitable.

In our approach of an IR system we focus on two criteria: on one hand, we attempt to retrieve all relevant documents for a specific set of queries using different systems; while on the other hand, we also try to optimize the retrieval process and certain aspects of storing the inverted index.

We decided to work on Amazon reviews, since such product based reviews cause a list of different problems for an IR system: The reviews are rather short, which makes it difficult to extract relevant terms from each document. Furthermore, the colloquial and often ungrammatical language of these reviews causes

problems during pre-processing steps such as tokenization and consequently for the ranking itself. In this paper, we will present how different types of IR systems deal with these problems and how effective ranking for the Amazon reviews can be realized.

2 Related Work

To our knowledge, no approaches for building an IR system for user reviews have been proposed. Le and Mikolov (2014) develop a paragraph vector model and report results of up to 3.82% error rate. Their dataset of movie reviews is comparable to our dataset of product reviews. A thorough discussion of using language models in IR is given by Zhai (2008). Mitra and Craswell (2017) present current state-of-the-art IR systems and also discuss models investigated in this paper. However, they concentrate on neural network approaches.

3 System Structure

In the following, the indexing procedure is described as well as our proposed systems TF-IDF. Furthermore, we compare it to several probabilistic models, which usually perform quite well on IR tasks.

3.1 Indexing

We applied different kinds of pre-processing steps to realize our indexing procedure. Tokens were separated at whitespaces, with punctuations being separated. Abbreviations were not taken into account. Additionally, we kept English clitics for later normalization. The tokens were then lowercased and lemmatized. The resultant unique tokens are termed as types.

These types were taken as the terms of our inverted index. Each term is associated with a postings list, which includes all documents that contain the term alongside its frequency and its position in the specific document. A minimal example for an inverted index is shown in Figure 1.

Document 1	cameras
Document 2	cheap camera good camera
Document 3	cheap backpack cheap

(a) Collection

backpack	$<3, 1, [1]>$
camera	$<1, 1, [0]>$, $<2, 2, [1, 3]>$
cheap	$<2, 1, [0]>$, $<3, 2, [0, 2]>$
good	$<2, 1, [2]>$

(b) Inverted Index

Figure 1: Example for a minimal collection (a) and its inverted index (b). Each row in (b) represents a term and its postings list. Each triple of a postings list is to be read as <document id,term frequency,[list of positions in the document]>.

3.2 Cosine-based Ranking

The system ranks documents according to their proximity to a query. This proximity can be explained by the cosine similarity between the document query pair. We explain cosine ranking using smoothed TF-IDF values. TF, IDF and TF-IDF given a term *t* and a document *d* are defined as

$$\text{TF}(t, d) = 1 + \log_{10}(f(t, d)) \tag{1}$$

$$\text{IDF}(t) = \log_{10}\left(\frac{|d|}{f(t, d)}\right) \tag{2}$$

$$\text{TF-IDF} = \text{TF} * \text{IDF} \tag{3}$$

where $f(t, d)$ denotes the frequency of t in d, $|d|$ is the collection size, i.e. the number of documents in the collection and $f(t, d)$ denotes the number of documents t occurs in.[1]

We represent documents and queries as vectors, where each dimension of the vector is the TF-IDF value for a term t in a document d or a query q, while the

[1] See (Ramos 2000) for a general discussion of TF-IDF approaches.

terms are axes of the space. We resolve the similarity between these vectors using cosine similarity:

$$\text{COS-SIM}(d, q) = \frac{\sum_{i=1}^{|V|} d_i q_i}{\sqrt{\sum_{i=1}^{|V|} d_i^2}\sqrt{\sum_{i=1}^{|V|} q_i^2}} \tag{4}$$

where $|V|$ is the vocabulary size and d_i and q_i denote the TF-IDF value of term i in the document or query, respectively. COS-SIM ranges between 0 and 1, with a score of 0 signifying that the document d and query q are completely distinct. For the example query *cheap camera*, the ranking of documents is given in Table 1, using the inverted index in Figure 1.

Document ID	Cosine Similarity Value
2	0.9642
1	0.7071
3	0.7071

Table 1: Ranking for the query *cheap camera* using the example index in Figure 1 with cosine similarity values.

3.3 IR Systems

Four IR systems are created: While TF-IDF is a vector space based system, the other three (BM-25,JM and DS) are probabilistic IR systems.

3.3.1 Vector Space IR Systems

As explained in the section, documents are imagined as non-negative vectors of TF-IDF weights.

System TF-IDF This system is as explained in Section 3.2.

3.3.2 Probabilistic IR Systems

System BM-25 This is one of the widely used IR systems (Zhai 2008). For a document query pair, BM-25 is defined as:

$$\text{BM-25}(q,d) = \sum_{\text{unique } t \in q} \frac{(k_1+1)\cdot \text{TF}(t,d)}{k_1 \cdot ((1-b)+b\cdot(l_d/l_{avg})) + \text{TF}(t,d)} \cdot \text{IDF}(t), \quad (5)$$

where k_1, k_3 and b are tuning parameters, and l_d and l_{ave} are the length of a document d and the average length of all documents, respectively.

The parameter k_3 is only used for very long queries, and for our experiments we set k_1 to 1.5 and b to 0.75 as per commonly set defaults.

Language Model based systems A language model (LM) is a probability distribution over word sequences. The intuition behind using LMs for IR systems is given by the Maximum Likelihood Estimation (MLE), i.e. given the LM of documents, we estimate which document LM would most likely have generated a query. The documents are then ranked based on this query likelihood.

The two systems presented below involve some kind of smoothing parameter, that normalizes both TF and collection frequency (CF), where $CF(t) = \sum_d TF(t,d)$. All three systems ignore the document frequency.

System JM For a document query pair, the Jelinek-Mercer smoothing is:

$$\hat{P}_\lambda(q|d) = \sum_{\text{unique } t \in q} \lambda \frac{TF(t,d)}{l_d} + (1-\lambda)\frac{CF(t)}{|d|}, \quad (6)$$

where $|d|$ is the collection size of all d documents. For our experiments, the best tuning parameter λ is taken from the interval $\lambda \in [0.1, \ldots, 0.9]$.

System DS For a document query pair, the Dirichlet Prior Smoothing is given by:

$$\hat{P}_\mu(q|d) = \sum_{\text{unique } t \in q} \frac{l_d}{l_d+\mu} \cdot \frac{TF(t,d)}{l_d} + \frac{\mu}{\mu+l_d} \cdot \frac{CF(t)}{|d|} \quad (7)$$

where the best tuning parameter μ is one of 500, 1000, 1500 or 2000.

3.4 System Enhancements

In order to tackle space and time complexities of the IR system, two major enhancements were considered.

Firstly , we improved the run-time of our ranking procedure by not computing the COS-SIM for every document, rather only for those which contain at least one of the query terms, thereby achieving around 84% time reduction in construction of the ranked list.

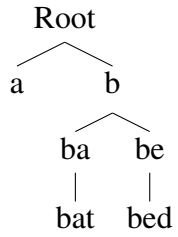


Figure 2: Example for storing the inverted index in an ordered tree.

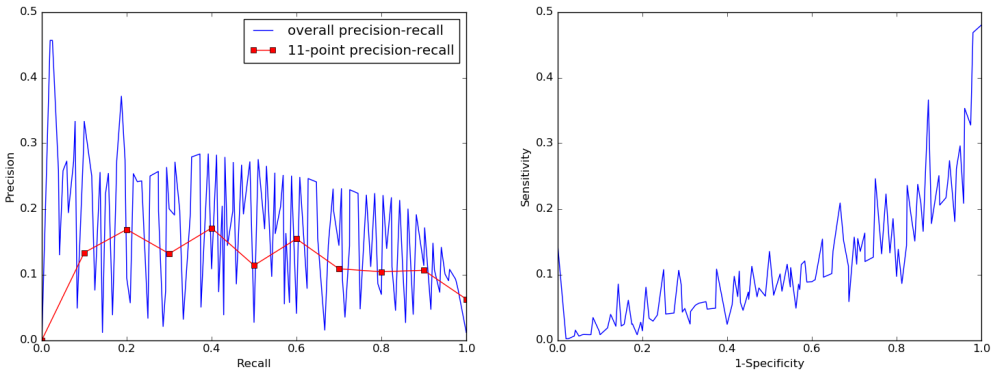
Secondly, we implemented an ordered B-tree (Bayer and McCreight 1970) to store the terms of the inverted index (shown in Figure 2). Previously, the index was stored in a hash table. Hash tables result in a rather inefficient IR system, as they store terms in an unordered fashion and the performance of the system depends on the number of terms n (time complexity of $O(n)$ on average). The B-tree implementation resulted in a 36% reduction in storage size of the inverted index, as well as ensuring that each retrieval has a time complexity of $O(\log n)$ at worst.

4 Experiments

We ran experiments for each system and evaluated their effectiveness on the given data.

4.1 Experimental Setting

The data used for the experiments consisted of 10,000 Amazon reviews on products such as vacuum cleaners, books and cameras. The average length for a review amounted to 7 sentences, 104 tokens and 60 types. For our system, we defined 30 queries (see Appendix A). None of our queries were Amazon centric, and generic enough to represent a regular web user. For the gold dataset, the top 2000 results from the ranked list were taken and manual annotation was performed for all the 30 queries. We also performed an inter-annotator agreement study using Cohen's Kappa (Cohen 1960), and got the value of $\kappa = 0.94$.



(a) Precision-Recall graph **(b)** ROC graph

Figure 3: Precision-Recall graph (a) and ROC graph (b) for the TF-IDF system, ran on 2,000 documents with 30 queries.

5 Results

For the evaluation of the ranked results for the four systems, we performed 11-point interpolated average precision (IAP), from 0.0 to 1.0 values of recall (shown in Table 2). A pair of graphs were generated for each experiment, namely precision-recall and ROC curves. At each stage of the IAP, the precision, recall, F-Score and specificity were computed and aggregated. The graphs are as shown in Figure 3. Finally the systems are evaluated on 21 commonly used IR metrics[2], such as precision at k docs, mean average precision, etc. The systems that recorded the highest and lowest scores for each metric was found and tabulated in Table 3.

6 Discussion

Queries like *camera good*, when compared to longer queries *tv with big screen high resolution*, consistently generated higher precision values. This could be due to importance of the words in smaller queries (low TF and low IDF values) outweighing the multiple terms in longer queries (high TF and high IDF values). Possible weight adjustments of the features could result in higher precision.

As seen from Table 2, the language model systems initially reported higher precision values. However, from recall of 0.4 onwards, system TF-IDF was consistently the best system.

[2]See Manning, Raghavan, and Schütze (2008).

Recall	Precision			
	TF-IDF	BM-25	DS	JS
0.0	0.107	0.110	**0.116**	0.110
0.1	0.084	0.071	**0.087**	0.081
0.2	0.069	0.062	0.073	**0.068**
0.3	**0.060**	0.056	**0.060**	0.056
0.4	**0.054**	0.048	0.053	0.047
0.5	**0.050**	0.043	0.048	0.041
0.6	**0.045**	0.040	0.042	0.038
0.7	**0.036**	0.031	0.033	0.030
0.8	**0.026**	**0.026**	0.024	0.025
0.9	**0.019**	**0.019**	0.017	0.017
1.0	**0.011**	0.010	0.010	0.010

Table 2: 11-point interpolated average precision for different systems. The highest precision value for each recall step is highlighted.

	H	L
DS500	2	0
DS1000	3	0
DS1500	4	0
DS2000	5	0
JS08	1	0
JS09	0	1
TF-IDF	9	0
BM-25	0	20

Table 3: System comparisons for how many times a system achieved the highest (H) or the lowest (L) score.

When comparing the different IR systems in Table 3, vector space model systems came on top. Even though system BM-25 is one of the widely used IR systems, it does not appear to work efficiently for product reviews. The system DS was overall the best language model system, and the increase in its tuning parameter μ seemingly increases its performance.

7 Conclusion and Future Work

The Amazon dataset considered was quite messy. It proved to be a challenge as each review varied vastly with another. In spite of this, and even with the usage of only the cosine similarity, we were able to get satisfactory results.

Currently in our system, documents and queries are being matched on a syntactic level (eg. *pretty* is matched to *pretty*), and semantic similarity between words not implemented (eg. *pretty* being matched to *beautiful* or *gorgeous*). Gaume, Hathout, and Muller (2004) have shown the improvements that semantics brought to information retrieval.

Furthermore, implementation of genetic algorithms has shown promising results (Oren 2002) and is worth exploring.

References

Bayer, R. and E. McCreight (1970). Organization and maintenance of large ordered indices. In *Proceedings of the 1970 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control*, SIGFIDET '70, New York, NY, USA, pp. 107–141. ACM.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement 20*(1), 37–46.

Gaume, B., N. Hathout, and P. Muller (2004). Word sense disambiguation using a dictionary for sense similarity measure. In *COLING 2004, Genève*, pp. 1194–1200. Association for Computational Linguistics.

Le, Q. and T. Mikolov (2014). Distributed representations of sentences and documents. In E. P. Xing and T. Jebara (Eds.), *Proceedings of the 31st International Conference on Machine Learning*, Volume 32 of *Proceedings of Machine Learning Research*, Bejing, China, pp. 1188–1196.

Manning, C. D., P. Raghavan, and H. Schütze (2008). *Introduction to Information Retrieval*, Chapter 8: Evaluation in information retrieval. New York, NY, USA: Cambridge University Press.

Mitra, B. and N. Craswell (2017). Neural models for information retrieval. cite arxiv:1705.01509.

Oren, N. (2002). Reexamining *tf.idf* based information retrieval with genetic programming. In *Proceedings of SAICSIT 2002*, pp. 224–234.

Ramos, J. (2000). Using TF-IDF to determine word relevance in document queries. Technical report, Department of Computer Science, Rutgers University.

Zhai, C. (2008). Statistical language models for information retrieval a critical review. *Foundations and Trends in Information Retrieval 2*(3), 137–213.

A Appendix: Queries

For our experiments the following 30 queries were used:

```
camera high memory
gift camera for my wife
32gb camera sd card
64gb camera
camera good
hd camera
camera good quality
camera good quality cheap
lightweight camera
small optical camera
camera with good memory
camera good quality expensive
camera with long battery runtime
phone camera resolution
phone camera with good resolution
I need a good camera with high resolution preferably cheap
camera low price
latest cameras
outdoor camera
camera sony
camera internet
good resolution mobile
phone with the best resolution
mobile samsung good quality
mobile big storage
phone cheap good quality
phone display
iphone 5 opinion
tablet screen
tv with big screen high resolution
```