An entropy-based term weighting scheme and its application in e-commerce search engines

Yang Jiao, Matthieu Cornec, Jérémie Jakubowicz

An entropy-based term weighting scheme and its application in e-commerce search engines

Yang JIAO
Cdiscount, Institut Mines-Télécom
yang.jiao@telecom-sudparis.eu

Matthieu CORNEC
Cdiscount
matthieu.cornec@cdiscount.com

Jérémie JAKUBOWICZ
CNRS, Institut Mines-Télécom
jeremie.jakubowicz@telecom-sudparis.eu

Abstract

Term weighting schemes are commonly used in information retrieval field to extract the most relevant terms of documents. The main contribution of this paper consists in defining a new term weighting scheme based on entropy. We believe that this scheme is particularly well adapted to compare queries from e-commerce sites. These queries have their own specificities. They tend to be short and a large proportion of them are unique queries, i.e. *have no historical record. We claim that widely used weighting schemes, such as* tf-idf*, are not well-adapted to this kind of queries. This claim is backed up by numerical experiments where the proposed entropy-based approach is incorporated into a collaborative filtering framework. In this framework, well suited to e-commerce search engines, we found out, on real e-commerce purchase data, that the proposed weighting scheme outperforms the tf-idf weighting scheme.*

Keywords: search engine, e-commerce, entropy, query similarity, collaborative filtering

I. Introduction

Nowadays large e-retailers offer up to millions of products in their marketplace. In order to guide customers rapidly to the most relevant products, the majority of e-retailers rely on a search engine. Therefore search engines have become a vital tool for the e-commerce industry. A now widely acknowledged procedure to boost the performance of a search engine consists in incorporating users feedback information in its design [1, 2]. In particular, in the context of e-commerce, purchase data is a useful feedback. They are a collection of pairs having the form $\langle \texttt{query}, \texttt{product} \rangle$, where `product` corresponds a purchase made just after `query` has been submitted, if any. It is indeed arguably less noisy than other feedback sources, such as clicks because it involves monetary transactions [11]. However, purchase data can be challenging to exploit, mainly for two reasons. Firstly, the vast majority of queries are unique, *i.e.* they do not appear elsewhere in the database. Secondly, a given user at a given time rarely buys many products: purchase data are extremely sparse (a few products among millions are associated to a given query). Their sparsity pattern is thence of order of magnitude 10^{-5}, while in other contexts, sparsity patterns of order 10^{-3} are already considered challenging [16].

In order to deal with such highly valuable but sparse data, a certain amount of *regularization* is needed. A popular way of performing such regularization is the so-called "collaborative filtering" [6] methodology. Roughly speaking, it consists in suggesting products not only associated with the given query, but also associated with other *similar* queries. The starting point is therefore a similarity function between two queries. There are basically two ways of comparing queries. The first way is to compare queries *via* the products purchased after them. For instance, "Apple tablet" and "Ipad" are similar in the sense that they usually yield the same purchases; although their content, *i.e.* the terms they are made of, are not similar. The second way is to compare their constituting terms. In this line of thoughts, it is important not to give the same weight to each term. Indeed, some terms are more informative than others. For instance query "sony black ps4" is closer to query "promo ps4" than to "sony black smartphone", even though the it is not the order implied by the number of common words. In this example, giving more weight to the term "ps4" than to the term "sony" or "black" can solve the problem. This weighting is meaningful, as the term "ps4" is arguably more informative than the term "sony", as it, alone, can limit considerably the relevant products range while "black" and "sony" can be used to describe a wide range of other products.

It is the purpose of this paper to introduce a term weighting scheme that is meaningful for e-commerce. Let us first argue why the existent schemes are not fully satisfying. The weighting scheme tf-idf [7] is a commonly used scheme in the information retrieval field. The "idf" part of the scheme is based on the assumption that rare terms are more relevant than frequent ones. Our claim is that the tf-idf scheme, although relevant in a large amount of situations [13, 9], is not relevant in the context we are interested in. In the tf-idf scheme, *rare* terms mean terms that do not appear frequently in the database. For instance, the term "ps4", that appears relatively frequently in the database, because the product "Playstation 4" is popular, is not considered as important as the term "color", which appears less in our database. We are not interested in exact figures at this stage but more on conceptual matters. Let us argue that "color" is less informative than "ps4": "color" is not related to a specific product; while "ps4" is likely related to the product "Playstation 4". In our proposed method, contrarily to the tf-idf weighting, we believe that the importance of a term should not solely be based on its number of occur-

rences, but should be mainly based on the diversity of purchases it has lead to. More precisely, we advocate that when the same term used in a large variety of purchases, it is less important than another term which is systematically associated to the very same purchase. Shannon entropy is a quantitative way to measure how much diverse a given term is. This is the reason why our proposed weighting scheme is based on entropy. We claim that this entropy-based weighting scheme gives interesting results in practice, compared to tf-idf; at least on our database. Notice that both methods are conceptually distinct since tf-idf only uses the queries, while the entropy weighting scheme uses both queries *and* products.

The rest of this paper is organized as follows. We first present the problem framework in Section II. Then in Section III, we introduce the proposed entropy-based term weighting scheme. Its application to e-commerce is described in Section IV, based on real data. Finally, we conclude in Section V.

II. Problem framework

Before detailing the main contribution of this paper, namely, a novel entropy-based term weighting scheme, we need to present the general framework in which we used this weighting. This general framework is the one of e-commerce search engines. The basic problem is to associate products to a given query entered by a user, relying on historical data. In this section, we first describe a few notations we use in this paper, then give a brief description of the data we used, and last, we detail the collaborative filtering methodology behind the search engine.

I. Notations

When searching through the search engine, a customer types a query *i.e.* a string denoted by q. Thus, q belongs to the set of all possible queries denoted by $\mathcal{Q}$. The search engine will then return an ordered list of the most relevant products from the catalog. We will denote the catalog by $\mathcal{P}$ and a product by p. In our case, the catalog contains millions of products.

In general, a search engine could be defined as a map denoted by f from the set of queries to the set of all possible ordered lists of products. In this paper, for the sake of simplicity, we define a search engine as a map $f_r : \mathcal{Q} \to \mathcal{P}^r$ from the set of queries onto the set of product r-tuples. More specifically, we are interested in the case where r is small, say $r \leq 10$, since we focus on the first page of results.

The *learning set* or *database*, consists of N pairs $(q_i, p_i)_{1 \leq i \leq N}$ where $q_i \in \mathcal{Q}$ is the last query before the purchase of product $p_i \in \mathcal{P}$.

II. Data description

This work is based on real purchase data, gathered by a major French e-retailer. The training (respectively test) set contains one million (resp. one hundred thousand) observations.

We observed the same qualitative features as the ones reported in [5], namely:

1. Power-law distribution: few distinct queries are very common and explain a large amount of purchases. In the database we used, around 4% of the queries account for 55% of the purchases. Indeed, many customers are looking for the same things. And a large number of very rare queries account for a significant amount of purchases. In the database we used 87% of the queries explain about 30% of all purchases. Similar phenomenon is reported in other e-commerce databases [11]. Those long tails queries should not only be neglected, but be treated with care [4].

2. Sparsity: every query is related to a very small percentage of the product catalog. Indeed, the frequency of the queries is very small in comparison with the size of the catalog.

3. New queries: queries without historical purchase data occur on a daily basis. Existing studies confirm what we observed: a single day contains over 20% of new queries in a 4 months time window [5].

In order to deal with these challenges, *collaborative filtering* [6] is a commonly used tool, that we describe now. Its main feature is a regularizing effect that can efficiently cope with the inherent sparsity of the data.

III. Neighborhood-based collaborative filtering

Neighborhood-based collaborative filtering is one of the simplest, yet powerful, recommendation procedure. Start from a raw score function $S_0 : \mathcal{Q} \times \mathcal{P} \to \mathbb{R}$. Score $S_0(q, p)$ indicates, *a priori*, how good product p is likely to match query q. For example, it could be the number of times product p was purchased when query q was issued (and, for instance, 0 if query q is not in the database), or any non-necessarily linear transformation of this number (log, etc.). In this example, it appears that whenever a query q has never been seen, all scores $S(q, p)$ are uniformly 0, whatever the product p. Nevertheless, one can encounter *similar* queries in the database, and hence start making interesting recommendations, pushing the products associated with these similar queries. What is needed to implement such a procedure is a similarity function $\text{sim} : \mathcal{Q}^2 \to \mathbb{R}$ such that $\text{sim}(q, q')$ assesses how close query q' is from query q (the higher the similarity, the closer q' is to q).

$$S(q, p) \stackrel{def}{=} \sum_{q'} \text{sim}(q, q') S_0(q', p) \qquad (1)$$

Given a query q, natural candidates are the products that are given the highest scores $S(q,p)$. The similarity function sim plays a crucial role in this method. A few examples are: $\text{sim}_{\text{Pearson}}(q,q') = \frac{\sum_{p\in\mathcal{P}} S_0(q,p)S_0(q',p)}{\left(\sum_{p\in\mathcal{P}} S_0(q,p)^2 \sum_{p\in\mathcal{P}} S_0(q',p)^2\right)^{1/2}}$ and $\text{sim}_{\text{Jaccard}}(q,q') = |\{q\}\cap\{q'\}|/|\{q\}\cup\{q'\}|$, where $\{q\}$ denotes the set of terms composing query q. The first example $\text{sim}_{\text{Pearson}}$ compares two queries q and q' based on their associated purchases: if q and q' led to the same purchases, they are considered similar, even if they do not have common terms. For example, "ipad" and "apple tablet" could be considered similar in this regard. However, this similarity only makes sense when there are some historical data related to q and q'. If q is seen for the first time, there are no products associated to it and Pearson similarity is not well defined. Notice, on the contrary, that $\text{sim}_{\text{Jaccard}}$ is well defined, as soon as q and q' are not made of unseen terms. In the next section, we focus on an extension of Jaccard-like similarity functions that measures the similarity between queries based on their constituting terms.

III. The proposed entropy-based term weighting scheme

The Jaccard similarity function basically counts the number of common terms between the queries, conveniently renormalized. A natural generalization consists in giving each term a separate weight, according to its importance. The aim of this section is to propose a new measure for the importance of terms.

I. Term importance

Consider the query "apple ipad". The term "ipad" carries most of the information, as it, alone, can tell us what kind of product is expected; while the term "apple" can be associated to a broader range of products. When computing query similarities, we should consider queries sharing the term "ipad" to be more similar than those sharing the term "apple". Therefore "apple ipad" should be more similar to "ipad 128g" than to "apple fuji". All three queries occur commonly in our database.

A classic way to assess the importance of a term is the so-called *tf-idf* (term frequency - inverse document frequency) term weighting scheme [7], which is widely applied in document retrieval. It is based on two assumptions.

1. *idf assumption*: rare terms are more informative than frequent terms.

2. *tf assumption*: multiple occurrences of a term in a query document are more relevant than single occurrence.

This scheme is perfectly relevant for large size documents, however, we claim that it is less relevant for e-commerce queries. The "tf" component, *i.e.* the frequency within a query, is nearly useless for e-commerce queries: a user rarely repeat a term in a query. Thus only the "idf" part matters: a term is informative when it is rare in the query database. However in e-commerce query logs, best-seller products are, by definition, frequent in the database. Those terms are thus groundlessly penalized by the tf-idf weighting scheme.

In our proposed method, contrarily to the tf-idf weighting, we believe that the importance of a term should not solely be based on its number of occurrences, but should be mainly based on the diversity of purchases it has lead to. More precisely, we advocate that when the same term used in a large variety of purchases, it is less important than another term which is systematically associated to the very same purchase.

To implement this idea, we employ the notion of Shannon Entropy of a discrete probability distribution [14], which we shall explicitly describe now.

II. Mathematical framework of entropy-based term weighting

Recall the notion of Shannon Entropy of a discrete probability distribution [14]. Given a probability distribution π on a finite set I, the Shannon Entropy is defined as:

$$H(\pi) \stackrel{def}{=} -\sum_{i\in I} \pi_i \log \pi_i \tag{2}$$

Now, to each term t, associate the following probability distribution, referred to as *term purchase distribution*:

$$\pi(t) = \frac{1}{Z_t} \sum_{\langle q,p\rangle\in\mathcal{D}} \mathbb{I}\{t\in q\}\delta_p \tag{3}$$

where δ_p denotes the probability distribution with all its mass on product p and Z_t, corresponding to the number of purchases associated to t is a normalization term such that π_t be a probability distribution over $\mathcal{P}$. For the sake of simplicity we denote $H(t) = H(\pi(t))$.

Table 1 shows a toy example with four queries ("hp printer" two times, "hp 3050a", "hp pc") and three products ("p1", "p2", "p3"). Table 2 describes the same example from the terms viewpoint: there are four terms in this example, "hp", "printer", "3050a" and "pc".

The entropy of terms in the previous sample can be calculated as follows.

$$H(hp) = -\tfrac{1}{2}\log(\tfrac{1}{2}) - 2\times\tfrac{1}{4}\log(\tfrac{1}{4}) = \tfrac{3}{2}\times\log 2$$
$$H(printer) = -2\times\tfrac{1}{2}\log\tfrac{1}{2} = \log 2$$
$$H(3050a) = -\log 1 = 0$$
$$H(pc) = -\log 1 = 0$$

Query	Product
hp printer	p_1
hp printer	p_2
hp 3050a	p_1
hp pc	p_3

Table 1: *Toy example*

	p_1	p_2	p_3
hp	0.5	0.25	0.25
printer	0.5	0.5	0
3050a	1	0	0
pc	0	0	1

Table 2: *Term-products*

Among frequent terms, it is a consequence of the definition of entropy that those with dispersed purchase distribution have higher entropy values than those with concentrated ones. We studied a subset of terms of the toy example, namely "hp" and "3050a". Both terms also happen to appear in our real-world e-commerce database, not equally frequently though. For instance, on the training set, coming from the mentioned real-world database, purchase distribution of "hp" and "3050a" are presented in figure 1 with pie charts. We can clearly see that the purchase distribution of "hp" is more dispersed than the one associated with "3050a" is, which explains the higher entropy value of the former. So far, we have seen that term importance is

(a)$H(hp) = 5.79$ (b) $H(3050a) = 1.05$

Figure 1: *Purchase distributions associated to "hp" and "3050a". The products are pictured as sectors and the arc length of each sector is proportional to the corresponding purchase frequency. It appears that "hp" leads to a larger variety of purchases than "3050a" which explains its high entropy level than the latter.*

inversely related to its entropy. We further apply an exponential transformation, that leads to a weight homogeneous to a probability:

$$w(t) = \exp\left(-\lambda \times H(t)\right) \tag{4}$$

The smoothing parameter λ can be tuned, depending on the application. It is worth noticing that our weighting scheme takes values in $(0, 1]$. The lowest weight occurs on terms with extremely dispersed purchase distribution. In our real-world dataset, "woman" and "man", have the lowest weights since a large range of products are associated to them.

Let us now develop how this weighting scheme can be used in the collaborative filtering framework to improve query similarity computations.

III. Entropy-based query similarity metrics

Query similarity is a key element in eq. (1). It is well acknowledged that lexical similarity metrics performs poorly when queries in question are extremely short [8], which is precisely our case since the average length of a search query in our database is around three. Techniques based on query reformulation are proposed in various papers [15, 10] to rewrite a query into a more meaningful form before any further processing. In this work we propose instead to assign different weights to different terms using eq. (4). For instance, $\text{sim}_{\text{Jaccard}}$ becomes:

$$\text{sim}_{\text{WeightedJaccard}}(q, q') \overset{def}{=} \frac{\sum\limits_{t \in \{q\} \cap \{q'\}} w(t)}{\sum\limits_{t \in \{q\} \cup \{q'\}} w(t)} .$$

How to use this similarity function in practice is addressed in the following section.

IV. Numerical experiments

In order to demonstrate the effectiveness of the proposed entropy-based term weighting scheme, we conducted several numerical experiments on real e-commerce data. Let us first present our experiments setting, then follow by introducing the evaluation metric we used, and lastly analyze the results we obtained.

I. Experiment setting

The data is described in II.II. As we worked on a French corpus, each query was preprocessed as follows: French accent removal, stop-words removal, special characters replacement by space, lowercasing and stemming. We used Porter's stemmer [12] to aggregate syntactically similar queries. It allows to alleviate term plurality and French gender mismatch issue.

II. Evaluation metric

In order to compare the performances of different ranking functions, there are several well known metrics [3]: MAP, NDCG, or simply the Precision@r metric which is the one we used. In our context, this metric is defined by:

$$\text{Precision@}r(f_r) = \frac{1}{T} \sum_{\langle q,p \rangle \in \mathcal{T}} \sum_{i=1}^{r} \mathbb{I}\{f_r(q)_i = p\} \tag{5}$$

where, we recall that f_r is a function returning the top ranked r products by ranking function (1), $\mathcal{T}$ is the test set and T its cardinality. Notice that Precision@r depends on r. Consequently, it may happen that a given search engine performs better at a given r but worse at another r'. In that case, using integrated metrics such as MAP can help. However, it is going to turn out in our experiments that such a sophistication is not needed.

III. Results and analysis

We used function $S(q,p)$ defined by eq. (1) to rank all the products for query q in the test data with the following weighted similarity functions sim and S_0.

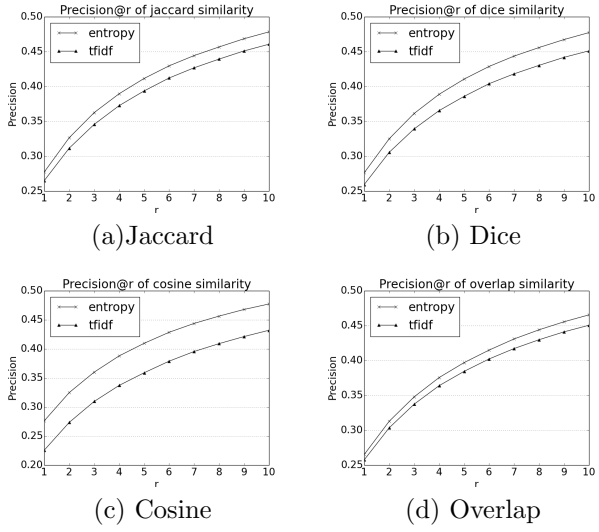
$$\begin{aligned} &S_0(q,p) = \log(1+ps(q,p)),\\ &\mathrm{sim}(q,q') = (1-\alpha)\,\mathrm{sim}_w(q,q') + \alpha \times \mathbb{1}_{q=q'}, \end{aligned} \tag{6}$$

where $ps(q,p)$ denotes the number of purchases of product p after the search of query q; α is the weight given to exact matching of queries optimized on a held-out training set; sim_w denotes a weighted similarity metric. Four commonly used similarity metrics: Jaccard, Cosine, Dice and Overlap were implemented in their weighted version:

$$\begin{aligned} \mathrm{sim}_{\mathrm{wjaccard}} &\frac{\sum\limits_{t\in\{q\}\cap\{q'\}} w(t)}{\sum\limits_{t\in\{q\}\cup\{q'\}} w(t)}\\ \mathrm{sim}_{\mathrm{wcosine}} &\frac{\sum\limits_{t\in\{q\}\cap\{q'\}} w(t)}{\sqrt{\sum\limits_{t\in\{q\}} w(t)^2 + \sum\limits_{t\in\{q'\}} w(t)^2}}\\ \mathrm{sim}_{\mathrm{wdice}} &\frac{2\sum\limits_{t\in\{q\}\cap\{q'\}} w(t)}{\sum\limits_{t\in\{q\}} w(t) + \sum\limits_{t\in\{q'\}} w(t)}\\ \mathrm{sim}_{\mathrm{woverlap}} &= \frac{\sum\limits_{t\in\{q\}\cap\{q'\}} w(t)}{\min(\sum\limits_{t\in\{q\}} w(t), \sum\limits_{t\in\{q'\}} w(t))} \end{aligned} \tag{7}$$

Both tf-idf and the proposed entropy-based weighting scheme were set as weights $w(t)$ into these four similarity metrics.

Experimental results using Precision@r metric with different values of r are presented in figure 2. We observe that entropy-based term weighting consistently outperforms tf-idf whatever the similarity function used and for all values of r. Comparing



(a)Jaccard (b) Dice

(c) Cosine (d) Overlap

Figure 2: *Performance comparison of entropy-based weighting and tf-idf weighting using different similarity metrics. X-axis is the number of allowed recommendations, corresponding to the parameter r of* (5)*. Y-axis is the Precision@r value.*

to tf-idf which assigns higher weights to rare terms and lower weights to frequent terms, the proposed entropy-based term weighting scheme share some common points but also differs in some others. Rare terms have, in average, a low entropy and thus a high importance as it can be observed in eq. (4). But if a frequent term has a relatively concentrated distribution such as "galaxy3", it can still have a relatively low entropy value, thus high importance. Only terms with high frequency *and* dispersed purchases are considered not important. Some examples are presented in table 3.

Term	Entropy	Explanation
hp	5.8	high freq., dispersed purchases
galaxy3	0.69	high freq., concentrated purchases
cn046a	0.5	Low frequency

Table 3: *Examples of term entropy*

Moreover since the terms describing best-sellers occur quite often, high frequency terms could be more important than less frequent ones. For example, the term "ps4" is more frequent than the term "black" in our query log, however the former is clearly more informative than the latter about what products the user is looking for, see table 4.

term: t	$w_{entropy}(t)$	$w_{tfidf}(t)$
term: sony	1	1
term: ps4	840	1.25
term: black	8.05	1.30
term: promo	4.95	1.57
term: smartphone	8.2	1.4

Table 4: *Examples of entropy-based term weighting with $\lambda = 2$ and tf-idf term weighting. Both schemes are normalized on the weight of "sony" in order to have a relative view of term importance.*

sony black ps4	Jaccard	tf-idf	entropy
sony black smartphone	0.5	0.46	0.01
promo ps4	0.25	0.24	0.98

Table 5: *Similarities with "sony black ps4" on different metrics*

Consider the query "sony black ps4" for example. It is more similar to "sony black smartphone" than to "promo ps4" using tf-idf, according to figures from table 5. Entropy-based weighting reveals that "ps4" is more informative than the other terms in the query. The reason is that most queries containing "ps4" end up with a purchase of a playstation 4. Thus "promo ps4" is considered as very similar to "sony black ps4" regardless of the number of terms in common.

V. Conclusion and future work

We have seen in this paper that query similarity measurement was an important issue, at the core of higher level tools, such as collaborative filtering. After having reviewed a popular weighting scheme, namely tf-idf, which is based on the idea that corpus-wise rarest terms are the most important, we introduced a novel term weighting scheme. This scheme is based on the idea that the importance of a term cannot be decided on its number of occurrences in the database alone. Rather, term importance, as we defined it, is based on how concentrated were the purchases it led to. This notion was implemented through the computation of term entropy that we defined in this paper. Numerical experiments, performed on real-world purchase data, showed encouraging results for the entropy-based term weighting over tf-idf. Many questions still remain open. Terms can have joint effects in a query that is not properly captured by the weighting scheme we propose. For example, some terms can be masked by others such as "apple" in "apple ipad", or some terms can have stronger meanings such like "case" in "iphone6 case". A tool that takes into account these joint effects would probably improve the overall performances.

References

[1] Ricardo Baeza-Yates and Yoelle Maarek. Usage data in web search: benefits and limitations. In *Scientific and Statistical Database Management*, pages 495–506, 2012.

[2] Vimala Balakrishnan and Xinyue Zhang. Implicit user behaviours to improve post-retrieval document relevancy. *Computers in Human Behavior*, 33:104–112, 2014.

[3] W Bruce Croft, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*. Addison-Wesley Reading, 2010.

[4] Sharad Goel, Andrei Broder, Evgeniy Gabrilovich, and Bo Pang. Anatomy of the long tail: ordinary people with extraordinary tastes. In *Proceedings of the third ACM international conference on Web Search and Data Mining*, pages 201–210, 2010.

[5] Mohammad Al Hasan, Nish Parikh, Gyanit Singh, and Neel Sundaresan. Query suggestion for e-commerce sites. In *Proceedings of the fourth ACM international conference on Web Search and Data Mining*, pages 765–774, 2011.

[6] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 230–237, 1999.

[7] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[8] Donald Metzler, Susan Dumais, and Christopher Meek. Similarity measures for short segments of text. In *Advances in Information Retrieval: 29th European Conference on IR Research, ECIR 2007, Rome, Italy, April 2-5, 2007, Proceedings*, volume 4425, 2007.

[9] Jiaul H Paik. A novel tf-idf weighting scheme for effective ranking. In *Proceedings of the 36th international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 343–352, 2013.

[10] Nish Parikh, Prasad Sriram, and Mohammad Al Hasan. On segmentation of ecommerce queries. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1137–1146, 2013.

[11] Nish Parikh and Neel Sundaresan. Inferring semantic query relations from collective user behavior. In *Proceedings of the 17th ACM conference on Information and Knowledge Management*, pages 349–358, 2008.

[12] Martin Porter. Snowball: A language for stemming algorithms, 2001.

[13] Rajendra Kumar Roul, Omanwar Rohit Devanand, and SK Sahay. Web document clustering and ranking using tf-idf based apriori approach. *IJCA Proceedings on ICACEA*, 2014.

[14] Claude E Shannon. A mathematical theory of communication. *Bell System Tech. J*, 27, 1948.

[15] Bishan Yang, Nish Parikh, Gyanit Singh, and Neel Sundaresan. A study of query term deletion using large-scale e-commerce search logs. In *Advances in Information Retrieval*, pages 235–246. 2014.

[16] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management*, pages 337–348. 2008.