BOOSTING ITEM FINDABILITY: BRIDGING THE SEMANTIC GAP BETWEEN SEARCH PHRASES AND ITEM INFORMATION

Hasan Davulcu, Hung V. Nguyen, Viswanathan Ramachandran
Department of Computer Science and Engineering, Arizona State University Tempe, AZ 85287, USA

Keywords: E-commerce, Data Mining, Frequent Itemsets, Web Data, Information Retrieval, Information Extraction, Relevance Feedback.

Abstract: Most search engines do their text query and retrieval based on keyword phrases. However, publishers cannot anticipate all possible ways in which users search for the items in their documents. In fact, many times, there may be no *direct keyword match* between a search phrase and descriptions of items that are perfect "hits" for the search. We present a highly automated solution to the problem of *bridging the semantic gap* between item information and search phrases. Our system can learn rule-based definitions that can be ascribed to *search phrases* with dynamic connotations by extracting structured item information from product catalogs and by utilizing a frequent itemset mining algorithm. We present experimental results for a realistic e-commerce domain. Also, we compare our rule-mining approach to vector-based relevance feedback retrieval techniques and show that our system yields definitions that are easier to validate and perform better.

1 INTRODUCTION

Most search engines do their text query and retrieval using keywords. The average keyword query length is under three words (2.2 words (Crescenzi, 2000)). Recent research (Andrews, 2003) found that 40 percent of companies rate their search tools as "not very useful" or "only somewhat useful." Further, a review of 89 sites (Andrews, 2003) found that 75 percent have keyword search engines that fail to retrieve important information and put results in order of relevance; 92 percent fail to provide guided search interfaces to help offset keyword deficiencies (Andrews, 2003), and seven out of 10 web shoppers were unable to find products using the search engine, even when the items were stocked and available.

The Defining Problem: Publishers cannot anticipate all possible ways in which users search for the items in their documents. In fact, many times, there may be no *direct keyword match* between a search phrase and descriptions of items that are perfect "hits" for the search. For example, if a shopper uses "motorcycle jacket" then, unless the publisher or search engine knows that every "leather jacket" is a "motorcycle jacket", it cannot produce all matches for user's search. Thus, for certain phrases, there is a ***semantic gap*** between the search phrase used and the way the corresponding matching items are described. A serious consequence of this gap is that it results in unsatisfied customers. Thus *there is a critical need to boost item findability by bridging the semantic gap that exists between search phrases and item information*. Closing this gap has the strong potential to translate web search traffic into higher conversion rates and more satisfied customers.

Issues in Bridging the Semantic Gap: We denote a search phrase to be a "*target search phrase*" if does not directly match certain relevant item descriptions. The semantics of items matching such "t*arget search phrases*" is i*mplicit* in their descriptions. For phrases with fixed meanings i.e. their connotations do not change such as in "animal print comforter", it is possible to close the gap by extracting their meaning with a thesaurus (Voorhees, 1998) and relating it to product descriptions, such as "zebra print comforter" or "leopard print bedding" etc. Where they pose a more interesting challenge is when their meaning is subjective, driven by perceptions, and hence their connotations change over time as in the case of "fashionable handbag" and "luxury bedding". The concept of a fashionable handbag is based on trends, which change over time, and correspondingly the attribute values characterizing such a bag also changes. Similarly, the concept of “luxury bedding” depends on the brands and designs available on the market that are considered as luxury and their attributes. Bridging the semantic gap therefore is in essence the problem of inferring the meaning of search phrases in all its nuances.

Our Approach: In this paper we present an algorithm that (i) structures item information and (ii) uses a frequent itemset mining algorithm to learn the “target phrase” definitions**.**

2 RELATED WORKS

In (Aholen, 1998), *generalized episodes* and *episode rules* are used for Descriptive Phrase Extraction. *Episode rules* are the modification of association rules and *episode* is the modification of frequent set. An *episode* is a collection of feature vectors with a partial order; authors claimed that their approach is useful in phrase mining in Finnish, a language that has the relaxed order of words in a sentence. In our previous work (Nguyen, 2003), we present a co-occurrence clustering algorithm that identifies phrases that frequently co-occurs with the target phrase from the meta-tags of Web documents. However, in this paper we address a different problem; we attempt to mine the phrase definitions in terms of extracted item information, thus, the mined definitions can be utilized to connect “search phrases” to real items in all their nuances.

The frequent itemset mining problem is to discover a set of items shared among a large number of records in the database. There are two main search strategies to find the frequent items set. Apriori (Agrawal, 1994) and several other Apriori like algorithms adopt Breadth-First-Search model, while Eclat (Zaki, 2000) and FPGrowth (Han, 2000) are well known algorithms that employ Depth-First manner to search all frequent itemsets of a database. Our algorithm also searches for frequent itemsets in a Depth-First manner. But, unlike the lattice structure used in Eclat or the conditional frequent pattern tree used in FPGrowth, we propose the so called 2-frequent itemset graph and utilize heuristic syntheses to prune the search space in order to improve the performance. We plan to further optimize our algorithm and conduct detailed comparisons to the above algorithms.

The relevance feedback (Salton, 1990) method can also be used to refine the original keyword phrase by using the document vectors (Baeza-Yates, 1999) of the extracted relevant items as additional information. In Section 6, we present experimental results and show that the rules that our system learns, by utilizing the extracted relevant item information, are easier to validate and perform better than retrieval with the relevance feedback method.

3 SYSTEM DESCRIPTION

I. Item Name Structuring: This component takes a product catalogue and extracts structured information for mining the *phrase based* and *parametric* definitions. Details are discussed in Section 4.
II. Mining Search Phrase Definitions: In this phase, we divide the phrase definition mining problems into two sub problems (i) mining the parametric definitions from extracted attribute value pairs of items, and (ii) mining phrase based definitions from the long item descriptions. Details are discussed in Section 5.

4 DATA LABELING

This section presents the techniques for an e-commerce domain, for the sake of providing examples. Our techniques can be customized for different domains. The major tasks in this phase are *structuring and labeling* of extracted data. The readers are also referred to (Davulcu, 2003) for more information in details.

4.1 Labeling and Structuring Extracted Data

This section describes a technique to partition the short product item names into their various attributes. We achieve this by grouping and aligning the tokens in the item names such that the instances of the same attribute from multiple products fall under the same category indicating that they are of similar types.

The motivation behind doing the partition is to organize data. By discovering attributes in product data and arranging the values in a table, one can build a search engine which can enable quicker and precise product searches in an efficient way.

4.2 The Algorithm

Before proceeding to the algorithm, it helps to identify item names as a sequence of *tokens* obtained when white-space is used as a delimiter. Since the sequences of tokens obtained from item names are all from a single web page and belong to the same category, they are likely to have a similar pattern. As mentioned before, our algorithm is designed to process collections of such item names without any labeling whatsoever. So it can be performed on the fly as and when data is extracted from the web sites. Following are the general properties of the data our algorithm can process:

Super-Tokens: Any pair of tokens t_1, t_2 that always co-occur together and occur more than once belong to a multi token instance of a type.

Context: All single tokens occurring between identical attribute types belong to the same type. This means that if two tokens t_1 and t_2 from distinct item names occur in between same types T_L and T_R then they should be of the same type.

Anchor Type: A token that uniquely occurs within all item names should belong to a unique type, which we call an *Anchor Type*.

Density: Attribute types should be densely populated. Meaning that, every type should occur within the majority of item names.

Ordering: Pairwise ordering of all types should be consistent within a collection.

Tokenization: The item names are tokenized by using white space characters as delimiters. Tokens are stemmed so using the Porter Stemmer (Porter, 1980).

Super Tokenization: The second step identifies multi-token attributes.

Initialization of Types: To initialize, every item name is prefixed and suffixed with a *Begin* and an *End* token.

Context Based Inference: This step aligns tokens from different item names under a single type. This step takes advantage of tokens repeating across descriptions and operates based on the first assumption, Context, that tokens within similar contexts have similar attribute types.

If a token sequences t_x,t, t_y and t'_x, t', t'_y exist in D such that t_x, $t'_x \in T_p$ and t_y, $t'_y \in T_q$, then combine and replace the types of tokens t and t' with a new type T_n = Typeof(t) U Typeof(t') .

Type Ordering: In this step, the set of inferred types T are sorted based on their ordering in the original item names. We utilize the Pairwise Offset Difference (POD) metric to compare different types.

POD between types T_i and T_j is defined as:

$$POD_{ij} = \sum_{x \varepsilon T_i, y \varepsilon T_j} (f_x - f_y) \qquad (1)$$

where f_x is the token offset of x from the start of its item name and f_y is the token offset of y. If this value is greater than zero, then the type T_i comes after type T_j in the sorted order.

Due to space constraints, tokens have been aligned such that those from the same type are offset at the same column. The type numbers the tokens belong to are indicated at the top.

Algorithm 1: Item Name Partition

```
Input: D, a collection of item names $D_0$, $D_1$,...$D_n$
Output: T, an ordered set of Types $T_0$, $T_1$,...$T_m$ where $T_i$ is a bag of
tokens that are instances of the same product attributes

1:  Tokenization(D)
2:  SuperTokenization(D)
3:  for each unique token $t_i \varepsilon D_i$ where $D_i \varepsilon$ D do
4:      create new type $T_i$
5:      add $t_i$ to $T_i$
6:      add $T_i$ to T
7:  end for
8:  repeat
9:      ContextBasedInference($D$, $T$)
10: until no inference is made
11: SortTypes($D$, $T$)
12: repeat
13:     MergeTypes($D$, $T$)
14: until no types are merged
15: repeat
16:     MergeConcatenateTypes($D$, $T$)
17: until no types were merge-concatenated
```

Type Merging: A careful observation shows that some of the neighbouring types are fillers for each other. Meaning that, they are not instantiated together for any item name. Such types are candidates for merging and are called *merge compatible*. Merging at this point is logical because of our assumption that the types are densely populated.

Merge Concatenation: Finally, merge-concatenation is performed to eliminate sparsely populated types. Sparsely populated types are those with a majority of missing values. By our assumption, collections of item names should have dense attributes. This implies that the tokens of a sparsely populated type should be concatenated with the values of one of the neighbouring types.

4.3 Experimental Results

To evaluate the algorithm, our DataRover system was used to crawl and extract list-of-products from the following five Web sites: www.officemax.com, www.officedepot.com, www.acehardware.com, www.homeclick.com and www.overstock.com.

Three metrics were used to measure the effectiveness of the algorithm. The first two evaluate the ability to identify fragments of the descriptions to the correct type and the last one indicates the correctness of the number of attributes.

Precision indicates how correctly type-value pairs are identified.

$$Precision = \frac{\text{type-value pairs identified correctly}}{\text{type-value pairs identified}}$$

Recall This quantity indicates if every existing type-value pair is being identified.

$$Recall = \frac{\text{type-value pairs identified correctly}}{\text{type-value pairs existing in the data}}$$

Attributes Error Rate indicates the error in the number of attributes described in the set of product names.

$$Error = \frac{|\text{actual attribute count - guessed attribute count}|}{\text{actual attribute count}}$$

Table 1: Summary of Evaluation Measures for Different Web Sites for the Items Name Structuring Algorithm

Web Site	Attribute Count Error Rate	Type-Value Precision	Type-Value Recall
www.officemax.com	0.07	0.89	0.94
www.officedepot.com	0.17	0.89	0.90
www.acehardware.com	0.06	0.92	0.94
www.homeclick.com	0.05	0.91	0.91
www.overstock.com	0.40	0.62	0.79

5 MINING THE DEFINITION OF A TARGET PHRASE

In this section, we introduce the problem of mining definitions of a phrase from product data extracted from the matching Web pages. Using extraction techniques discussed in Section 4 we can retrieve tabular parametric attributes of matching products as well as their long descriptions. Next, we apply frequent itemset mining algorithms to learn the parametric definitions and phrase-based definitions of target phrases from the extracted product data. First, in Sections 5.1 thru 5.4 we introduce an algorithm that finds all frequent itemsets from a database. Section 5.5 discusses the problem of mining parametric definitions. In Section 5.6 textual definition mining is discussed. Since their introduction in 1994 by Agrawal et al. (Agrawal, 1994), the frequent itemset and association rule mining problems have received a lot of attention among data mining research community. Over the last decade, many research papers (Han, 2001) have been published presenting new algorithms as well as improvements on existing algorithms to tackle the efficiency of frequent itemset mining problems. The frequent itemset mining problem is to discover a set of items shared among a large number of transaction instances in the database. For example, consider the product information database matching 'trendy shoes' that we extract from retail Web sites. Here, each instance represents the collection of product's <attribute, value>pairs for attributes such as brand, price, style, gender, color and description. The discovered patterns would be the set of <attribute, value> pairs that most frequently co-occur in the database. These patterns define the parametric description of the target phrase 'trendy shoe'.

5.1 Boolean Representation of the Database

The advantage of Boolean representation is that many logical operations such as superset, subset, set subtraction, OR, XOR, etc between any number of attribute vectors can be performed extremely fast.

5.2 Constructing 2-frequent Itemsets Graph

The set of 2-frequent itemsets plays crucial role in finding all frequent itemsets. The main idea is that, from the observation that if $\{I_{i....}I_j\}$ is a frequent itemset then all pairs of items in this set must also be a frequent itemset. Using this property of a frequent set, our algorithm will first create a graph that represents the 2-frequent itemsets among all items that satisfy the minimum support threshold.

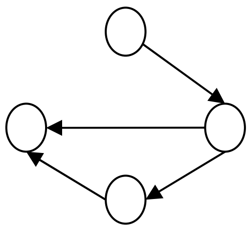
The the 2-frequent itemset graph is the directed graph G(V,E) which is constructed as follows:

V = *I*; *I* is the set of items that satisfy the minimum support in database *D*.

E = $\{(v_i,v_j) \mid \{i,j\}$ is a 2-frequent itemset and $i<j)$.

We sort the frequent single items into lexicographical order and for a 2-frequent itemset, we construct a directed edge from the node (item) whose index is lower to the node whose index is higher.

Example 1.



I_1	I_2	I_3	I_4
1	1	1	0
0	1	1	1
1	0	1	0
1	1	0	0
0	0	1	1
1	1	0	0
0	1	0	1
1	0	0	1

Figure 1: Database I and its 2-frequent item graph.

For this database, if minimum support δ is set to 25%, then the 2-frequent itemsets are I_1I_2, I_2I_3, I_2I_4, I_3I_4. The 2-frequent itemsets graph would be as in Figure 1.

5.3 Searching for Frequent Itemsets

The algorithm iteratively starts from every node in the graph and recursively traverses depth-first to its descendants. At any step k (k>1), the algorithm will choose to go to a child node *v* of the current node so that the path from the beginning node to *v* forms a k-frequent itemset. If so, the algorithm will continue expand to *v*'s children to search for (k+1)-frequent itemset and so on. There are several algorithms [8, 16] that generate frequent itemsets in depth-first manner. A distinguishing feature of our algorithm is that it searches on the 2-frequent itemset graph. Finding all 2-frequent set takes $O(n^2)$ operations where n is the number of frequent single items. Our algorithm utilizes the following heuristics to guide the search.

Heuristic 1: At step k, choose only children nodes of node v_{k-1} that have incoming degree greater than or equal to the number of visited nodes, counted from the beginning node. Incoming degree of a node *v*, denoted as deg(*v*) is the number of nodes that point to *v*. The meaning of this heuristic is that, if deg(*v*) is smaller than the number of visited nodes (nodes in the path) then there exists at least one node among the set of previously visited k-1 nodes that does not point to *v*. In other words, there exists at least one node in the current path that does not form a 2-frequent itemset with v. Therefore the k-1 nodes in the path (visited nodes) and *v* cannot form a k-frequent itemset hence it is pruned out without candidate itemset generation.

Heuristic 2: At step k, choose only children nodes of node v_{k-1} that have the set of incoming nodes that is a superset of the set of all k-1 nodes in the visited path. This heuristic, which is applied after Heuristic 1, ensures that all previously visited nodes in the current path, must point to the node in consideration. This is also a necessary precondition that each visited node forms a 2-frequent itemset with the node in consideration.

Heuristic 1 is efficient since the 2-frequent itemset graph is already constructed and the degree of all nodes is stored before the search proceeds. Heuristic 2 superset testing operation can also be performed efficiently using the bit-vector representation. Consequently, by utilizing these heuristic estimates, we can prune a lot of nodes that cannot be added to the visited nodes to form a frequent itemset and eliminate a lot of candidate itemset generation.

5.4 Mining Parametric Definition of Phrases

Note that, since we extract data from the Web by posing a search phrase query to a web search engine, all the instances in the data we get contain search phrase. Therefore, the association rule generation becomes simple by just putting the search phrase into the header of association rules and the body of rules is frequent itemsets. The support of obtained association rules equals to the support of frequent items set in their body since for a rule, the search phrase occurs in all instances that the frequent itemset (in the body of the rule) occurs. Next, we would like to utilize the extracted product information to mine parametric phrase definition rules made up from conjunctions of distinct <attribute, value> pairs, like:

Trendy shoe ←
brand = Steve Madden,
Color = black,
material = leather

5.5 Mining Textual Definitions of Target Phrases

Another resource of rich phrase definitions is the long product descriptions of the matching products. In the Section 4, we have already described how we plan to collect long product descriptions from product Web pages that matches a given target search phrase. In this section we describe the proposed algorithm for mining phrase definitions that can connect hidden phrases to product descriptions themselves. In order to generate candidate phrases first we perform part-of-speech (POS) tagging and noun and verb phrase chunking (Finch, 1997) on the long description to obtain a more structured textual description. Part-of-speech (POS) tagging and chunking the above description yields the following structure. In the next step, we utilize the noun phrases as transaction instances and mine frequently used phrases from all the noun phrases of all the product descriptions that we have collected from the Web documents.

Algorithm 2: Frequent Itemset Mining

```
Input: D, δ, I
Output: all frequent itemsets F(I,δ)
//main procedure
Construct all 2-frequent itemsets by counting the support of
all pairs of single items
Construct the 2-frequent itemsets graph G(V,E)
F[I, δ]:= {}
for all v in V do
        visited node list:= null
        add v to list of visited node list
        calculate_itemset (v)
endfor
end

Procedure calculate_itemset (v)
for each child v' of v do
    // Heuristic 1
    if deg(v') ≥ number of visited node then
        // Heuristic 2
        if the set of incoming nodes of v' is a superset of
        visited node then
            if the set of visited node and v' has counting ≥ δ
            then
                add v' to the list of visited nodes
                add visited nodes list to F(I, δ)
                calculate_itemset(v')
                //backtrack
                remove v' from the visited node set
            endif
        endif
    endif
endfor
end
```

Next, we use the mined frequent phrases as items and create transaction instances by marking all of the frequently used phrases matching anywhere in the long description. This would yield transaction instances made-up from frequently used phrases matching the product descriptions.

Next we mine the frequent itemsets among instances corresponding to the long descriptions to find the phrase definitions. Note that, due to our way to construct the items, all items are combinations of single words; therefore, there are items that subsume other items. As a subsequence, there are a lot of redundant final resultant frequent itemsets. For example a long description might yield the following items: "suede", "pump", "suede pump", "fashion", "savvy", "woman", "fashion savvy", "savvy woman", "fashion savvy woman". Hence, we only want to mine the frequent itemset "suede pump", "fashion savvy woman" because these frequent itemsets subsume the former frequent itemsets.

6 EXPERIMENTAL RESULTS

The tables below show some of the definitions that were mined. It is a relatively easy task for a domain expert to inspect and evaluate the quality of such rule-based definitions.

6.1 Comparison to Relevance Feedback Method

In order to compare the performance of our definition miner to standard relevance feedback retrieval method we mined a large database of shoes (33,000 items) from a collection of online vendors. Next, we keyword queried the database with the target exemplary search phrase "trendy shoe".From the 166 keyword matching shoes, we mined rule-based phrase definitions for "trendy shoes" yielding rules such as fashionable sneaker, platform shoes etc. that were validated by a domain expert. These mined rules matched 3,653 additional shoes. Alternatively, we also computed the relevance feedback query vector using the above 166 matching shoes. We also identified a similarity threshold by finding the maximal cosine theta, Θ, between the relevance feedback query vector and all of the 166 shoe vectors. Retrieval using the relevance feedback vector with this threshold yields more than 29,000 matches out of 33,000! The light colored bars in Figure 3 illustrates the histogram plot of the 29,293 instances that falls into various similarity ranges. Similarly, the dark colored bars plots the similarity ranges of the 3,653 shoes that were retrieved by matching with our mined definitions. As can be seen from the distributions in the above chart, the items retrieved with our mined definitions have a very uniform similarity distribution (with around 300 of these being below the threshold), as opposed to having a skewed distribution towards the higher values of similarity. Since dark colored bars correspond to relevant "trendy shoes" matching our rules, which were validated by an expert, most of these items should have ranked towards the higher end of the similarity spectrum. However, relevance feedback measure failed to rank them as such; hence, it performed poorly for this task.

6.2 Comparison to Relevance Feedback with LSI

The plot of similarity ranges obtained by ranking the 3,653 shoes, retrieved with our mined rules, using relevance feedback with and without latent semantic indexing (LSI) (Deerwester, 1990) technique is shown in Figure 2. The light colored dashed line represents the cosine theta threshold Θ for the relevance feedback ranking, similarly the dark colored dashed line represents the cosine theta threshold for the relevance feedback with LSI. The recall for relevance feedback is nearly 93%, however, since it matches 88% of a random collection of shoes, its precision is lower. On the other hand, even though the ranking of relevance feedback with LSI falls onto a higher similarity range, it appears to have a much lower recall (of 25%) for this experiment with exemplary target phrase "trendy shoes".

7 CONCLUSIONS AND FUTURE WORK

Our initial experimental results for mining phrase definitions are promising according to our retail domain expert who is the Webmaster of an affiliate marketing web site. We plan to scale up our experiments to hundreds of product categories and thousands of phrases. Also, we would like to perform experiments to determine how precisely our algorithm learns the definitions of phrases that changes their meaning over time.

	Parametric Rules	**Support**
Fashion handbags	Brand = Jil Sander, material = leather, type = clutch ➔ fashion handbags	4.25%
	Brand = Carla, design = mancini, material = leather ➔ fashion handbags	2.4%
	Brand = Butterfly, design =beaded ➔ fashion handbags	2.4%
	Brand = Sven, material = leather ➔ fashion handbags	10.2%
	Design = beaded, color = pink ➔ fashion handbags	2%
	Design = beaded, color = blue, type = tote ➔ fashion handbags	3.2%
Luxury beddings	Design = Baffled box, material = cotton ➔ luxury beddings	5%
	Design = Waterford, material = linen ➔ luxury beddings	6%
	Material = silk ➔ luxury beddings	3%
	Design = Sussex, material = polyester ➔ luxury beddings	6%
Sport beddings	Design = All American, material = polyester ➔ sport beddings	6%
	Design = All star, material = polyester ➔ sport beddings	9%
	Design = Big and bold ➔ sport beddings	17%
	Design = sports fan ➔ sport beddings	45%
	Textual Rules	**Support**
	Egyptian cotton mate-lass ➔ luxury beddings	0.6%
	Silk, smooth, King set ➔ luxury beddings	0.75%
	Piece ensemble ➔ luxury beddings	0.75%
	American sport ensemble ➔ sport beddings	0.4%
	Paraphernalia sport ➔ sport beddings	0.6%
	fashionable sneaker ➔ trendy shoes	7%
	Wedge edge ➔ trendy shoes	5%
	Platform shoes ➔ trendy shoes	6%

REFERENCES

R. Agrawal and R. Srikant. 1994, "Fast Algorithms for mining association rules". *In Proc. 20th Int. Conf. VLDB* pp. 487-499

H. Aholen, O. Heinonen, M. Klemettinen, and A. I. Verkamo. 1998, "Applying Data Mining Techniques for

Descriptive Phrase Extraction in Digital Collections*". In Proceedings of ADL'98,* Santa Barabara, USA

W. Andrews. 2003 "Gartner Report: Visionaries Invade the 2003 Search Engine Magic Quadrant",

V. Crescenzi, G. Mecca, and P. Merialdo. 2001 "Roadrunner: Towards automatic data extraction from large web sites", *In Proc. of the 2001 Intl. Conf. on Very Large Data Bases*.

H. Davulcu, S. Vadrevu, S. Nagarajan, I.V. Ramakrishnan. 2003, "OntoMiner: Bootstrapping and Populating Ontologies From Domain Specific Web Sites", in *IEEE Intelligent Systems*, Volume 18, Number 5.

Deerwester, S., Dumais, S. T., Landauer, T. K., Furnas, G. W. and Harshman, R. A. 1990, "Indexing. Latent semantic analysis", *journal of the Society for Information Science*, 41(6), pp. 391-407.

Steve Finch and Andrei Mikheev. 1997, "A Workbench for Finding Structure in Texts". *Applied Natural Language Processing* , Washington D.C.

J. Han J.Pei, Y.Yin, and R. Mao. 2000, "Mining frequent pattern without candidate generation." *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, volume 29(2) of SIGMOD Record, ACM Press.

J. Han, and M. Kamber. 2001, "*Data Mining: Concepts and* Techniques", Morgan Kaufmann Publishers.

Hung V. Nguyen, P. Velamuru, D. Kolippakkam, H. Davulcu, H. Liu, and M. Ates. 2003, "Mining "Hidden Phrase" Definitions from the Web". *APWeb*, Xi'an, China, Springer-Velag, LNCS Vol 2642, pp. 156-165.

M.F.Porter. 1980, "An algorithm for suffix stripping", Program, 14 no. 3, pp. 130-137.

G. Salton and C. Buckley. 1990, "Improving retrieval performance by relevance feedback", *journal of the American Society for Information Science*, pp. 288—297.

Ellen M. Voorhees. 1998, "Using WordNet for Text Retrieval". In WordNet: An Electronic Lexical Database, Edited by Christiane Fellbaum, MIT Press.

R. A. Baeza-Yates and Berthier A. Ribeiro-Neto. 1999, "Modern Information Retrieval", ACM Press / Addison-Wesley.

M.J. Zaki. 2000, "Scalable algorithms for association mining". *IEEE Transactions on Knowledge and Data Engineering*, 12(3), pp. 372-390.

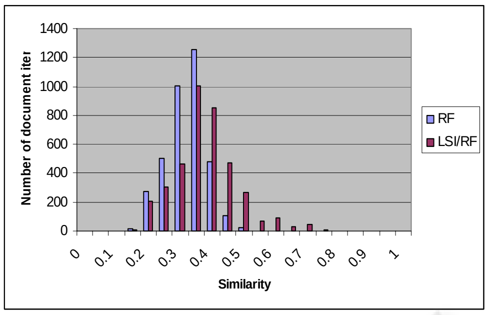


Figure 2: Similarity histogram for relevance feedback and relevance feedback with LSI

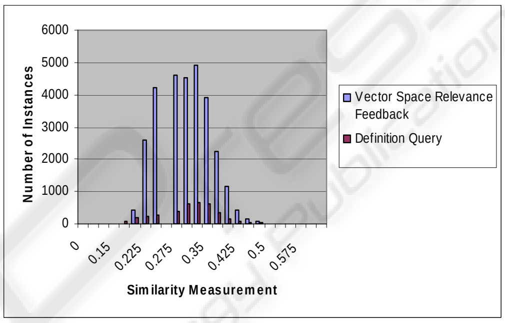


Figure 3: Similarity histogram for rule-based and relevance feedback based matches