Towards the Web of Concepts: Extracting Concepts from Large Datasets

Aditya Parameswaran
Stanford University
adityagp@cs.stanford.edu

Hector Garcia-Molina
Stanford University
hector@cs.stanford.edu

Anand Rajaraman
Kosmix Corporation
anand@kosmix.com

ABSTRACT

Concepts are sequences of words that represent real or imaginary entities or ideas that users are interested in. As a first step towards building a web of concepts that will form the backbone of the next generation of search technology, we develop a novel technique to extract concepts from large datasets. We approach the problem of concept extraction from corpora as a market-basket problem, adapting statistical measures of support and confidence. We evaluate our concept extraction algorithm on datasets containing data from a large number of users (e.g., the AOL query log data set), and we show that a high-precision concept set can be extracted.

1. INTRODUCTION

The next generation of search and discovery of information on the Web will involve a richer understanding of the user's intent and a better presentation of relevant information instead of the familiar "ten blue links" model of search results. This transformation will create a more engaging search environment for the users, helping them quickly find the information they need. Search engines like Google, Yahoo! and Bing have already started displaying richer information for some search queries, including maps and weather (for location searches), reviews and prices (for product search queries), and profiles (for people searches). However, this information is surfaced only for a small subset of the search queries, and, in most other cases, the search engine provides only links to web pages.

In order to provide a richer search experience for users, Dalvi et. al. [22] argues that web-search companies should organize search back-end information around a web of *concepts*. Concepts, as in [22], refer to entities, events and topics that are of interest to users who are searching for information. For example, the string "Homma's Sushi", representing a popular restaurant, is a concept. In addition to concepts, the web of concepts contains meta data corresponding to concepts (for example, hours of operation for Homma's Sushi) and connections between concepts (for example, "Homma's Sushi" is related to "Seafood Restaurants"). A web of concepts would not only allow search engines to identify user intent better, but also to rank content better, support more expressive queries and present the integrated information better.

Our definition of a concept is based on its usefulness to people. That is, a string is a concept if a "significant" number of people say it represents an entity, event or topic known to them. For instance, "Flying Pigs Shoe Store" is not a concept if only one or two people know about this store, even though this store may have a web page where "Flying Pigs Shoe Store" appears. As we discuss later, we also avoid "super-concepts" that have shorter equivalent concepts. For example, "The Wizard Harry Potter" is not a concept because "Harry Potter" is a concept identifying the same character. As we will see, other reasons why we restrict our definition to sequences of words that are popular and concise are scalability and precision.

At Kosmix (`www.kosmix.com`), we have been working on designing search around a web of concepts. Our aim at Kosmix is to automatically build a "concept page" with information for each known concept, giving among other things useful links and a list of related concepts. As soon as the search engine knows a user is looking for a known concept, it can display the concept page to help the user navigate and discover useful information.

The first step in building the Kosmix web of concepts is to extract concepts from data sources of various kinds: web-pages, query logs, tag datasets etc. A good source of concepts is the online user edited encyclopedia Wikipedia (`wikipedia.org`), currently with about 2.5 million articles. Since each article has been carefully selected by many Wikipedia editors, the titles of these articles are likely to correspond to what many people regard as concepts.

Even though Wikipedia can yield many concepts, we still wish to identify additional concepts. In particular, new concepts may arise because of current events. Also, many well known restaurants, hotels, scientific concepts, locations, companies, and so on, do not have Wikipedia entries. Thus, the concept extraction scheme we present plays a vital role in building the Kosmix web of concepts.

In addition to assisting in the search back-end, extracting concepts can also be very useful for other tasks, e.g., query correction, selective indexing and tag/query suggestion [25].

In this work we develop a novel technique of extracting concepts from datasets. We use frequency or popularity of words as an inherent signal to distinguish whether a sequence of words forms a concept relative to the word sequences contained within it (sub-concepts) and relative to the word sequences that contain it (super-concepts). Our technique can be applied to any dataset containing aggregated information from a large number of users, for example, a query log or a tag dataset, or even a large web-document crawl. Frequency information allows us to prune away the additional words in "super-concepts", leaving only the core concept. Since popularity of words plays a key role, it is more effective to use a dataset with a high "density" of concepts, such a query log or a set of tags. If we use a set of web pages for extraction then we need to do a lot more processing to identify the popular terms.

Our work builds upon a lot of related research in the areas of

Proceedings of the VLDB Endowment, Vol. 3, No. 1

market-basket mining and term recognition. However, because concepts are not quite frequent item-sets or terms, most existing methods do not directly apply. For example, a simple market-basket approach cannot be used to extract concepts as frequent item-sets [2], because we wish to discard both sub-concepts and super-concepts while only keeping the concept that we consider to be "good" (as we will see in Sec. 2.5). For example, we wish to discard 'Rage against the' or 'Against the machine' (sub-concepts), and 'Rage against the machine music' or 'Rage against the machine band' (super-concepts), and keep the concept 'Rage against the machine'(A popular rock/metal group). Similarly, most term recognition algorithms [14, 17, 19] do not consider popularity. For example, a word-sequence 'Ayocaria Rnfection' (a misspelling of 'Myocardial Infarction') may occur once in a document (and hence is not a concept, since it is not a popular entity), but will be considered as a term. However, we do compare our algorithm against a statistical technique for term recognition [15] that takes popularity into account. We return to this point in Sec. 4. We do not primarily rely on Natural Language Processing (NLP) techniques because they would not be of use when concepts are devoid of a free-text context [14, 19, 26], such as in the query log or tag dataset case.

In summary, our contributions are the following:

- We formulate the problem of concept extraction from large datasets.
- We provide a novel technique for solving the problem of concept extraction, and prove the correctness of this technique under some assumptions.
- We experimentally evaluate our technique against those used for term recognition [17, 19].
- We experimentally study our technique on varying various parameters for both precision and recall.

2. CONCEPT EXTRACTION

We first define a k-gram as a ordered sequence of k words occurring one after the other in text. e.g., "Mary had a little lamb" has three 3-grams: 'Mary had a', 'had a little' and 'a little lamb', and one 5-gram, which is the entire sequence of words.

2.1 Definition of a Concept

We define a *concept* (as in [22]) to be a k-gram that represents a real or imaginary entity, event or idea that many users may be interested in (i.e., is popular), and does not contain any extraneous words such that excluding them would identify the same entity (i.e., is concise). In this paper, we do not delve into any philosophical discussions on the correctness of this definition, involving ourselves instead on how we may use frequency information or popularity to give indications as to whether a sequence of words forms a concept.

For our evaluation of automatically extracted concepts, we assume all Wikipedia article titles to be concepts[1], since Wikipedia has a rigorous screening process to delete any article that "does not indicate why [the concept] is important or significant" [1]. In addition, we use human evaluators to judge if the k-grams we extract are concepts.

Note that enforcing the properties of popularity and conciseness is also helpful in other ways. For instance, by restricting ourselves to popular concepts, we exclude concepts of interest to only a small fraction of people, and by restricting ourselves to concise concepts, we remove duplicates in the web of concepts, thereby allowing us to keep our web of concepts smaller and more manageable.

[1]Note that there are Wikipedia articles that are not concepts, for example 'List of holidays by country'. These are either lists or disambiguation pages. We exclude them from our concept set.

2.2 Extraction of k-grams

Consider a large dataset C, that could be a query log, a web document crawl, text from a set of books, etc. A pre-processing step extracts all k-grams and annotates them with the frequency of their occurrence in C. For example, if 'Milky Way Galaxy' was a query term in dataset C, when we process this query, we would increment the frequency of 1-grams 'Milky', 'Way' and 'Galaxy', 2-grams 'Milky Way' and 'Way Galaxy' and 3-gram 'Milky Way Galaxy'. For documents from the Web, we treat each sentence as independent of the others and so we do not count k-grams that span two sentences. For example, 'This is Los Angeles. Here ...', we would not increase the count of 'Los Angeles Here'. For the web query log, we treat each web query as an independent sentence, as in the dataset of documents from the Web.

To prevent a blowup of k-grams being counted, we impose a cutoff on k, i.e., that $k \leq n$. Since very few concepts have greater than 4 words (as seen in Wikipedia), we stop at $n = 4$. In the following, we assume that we have a k-gram set, for $k \leq 4$, annotated with frequencies of occurrence in the dataset C. A k-gram x *contains* a (k–m)-gram y, where $m > 0$, if x can be written as w followed by y followed by z, where w and z are sequences of words themselves (not both empty), e.g., 'United Nations Security Council' (a 4-gram) *contains* 'Security Council' (a 2-gram).

2.3 A Crucial Empirical Property of Concepts

We claim that concepts *most often* satisfy the following empirical property. We describe the intuition behind the claim, and then provide some justification by examining Wikipedia concepts. We use this claim to develop the algorithm in Sec. 2.4 and beyond.

CLAIM 1. *If a given k-gram $a_1a_2 \ldots a_k$ $(k > 2)$ is a concept, then it is not true that both of the following k–1-grams are concepts: $a_1a_2 \ldots a_{k-1}$ and $a_2a_3 \ldots a_k$. If a 2-gram a_1a_2 is a concept, then at least one of a_1 and a_2 are concepts.*

The claim relates whether or not a k-gram can be a concept if the k–1-grams that it contains are concepts. For example, "Who framed Roger" and "Framed Roger Rabbit" do not make sense independently (i.e., are not concepts), but the Walt Disney film, "Who framed Roger Rabbit" makes sense as a concept. In this case, both the 2-grams contained in the 3-gram are not concepts. It is not necessary that both the k–1-grams are not concepts, for instance, consider 3-gram "Manhattan Experimental Theater", which contains the 2-grams "Manhattan Experimental" and "Experimental Theater", the latter of which is a concept.

However, for $k = 2$ there are examples of concepts that are formed from two 1-gram concepts. For example, while both "Computer" and "Networks" are independent concepts, "Computer Networks" is a concept in itself.

k	Total	Both k-1-grams	%age	$\geq$ One k-1-gram	%age
2	2245301	1250613	55.69	2147305	95.63
3	1357852	105515	7.77	688324	50.69
4	681447	12150	1.78	202763	29.75
5	350481	1822	0.51	64630	18.44
6	166827	525	0.31	22075	13.23

Table 1: Claim Justification.

To provide some justification for the claim, we performed the following experiment on the set of articles in Wikipedia: We assume that the title of each article represents a concept as mentioned in Sec. 1. For a given k, we took the subset of the set of Wikipedia concepts that are k-grams, and counted how many k-grams violated the claim above. The data is represented in Table 1. For example,

consider the row corresponding to 4-grams. The total number of 4-gram concepts is 681447, out of which 12150 contain two 3-gram concepts ($12150/681447 = 1.78\%$). 202763 contain one or more 3-gram concepts ($202763/681447 = 29.8\%$). While a large fraction ($\approx 56\%$) of concept 2-grams contain two concept 1-grams, this is not true for k-gram concepts for $k > 2$, none of which have $> 8\%$ that contain two k–1-gram concepts.

Note also that the number of k-gram concepts where at least 1 of the k–1-grams contained is a concept, is large for k-grams in the range $2 \ldots 5$. We capture this statistic in Claim 1 by saying that we sometimes have one of the k–1-grams as concepts. Additionally, almost 96% of the concept 2-grams contain at least one concept 1-gram. We capture this special case in Claim 1 by saying that all concept 2-grams contain at least one concept 1-gram.

Because Claim 1 does not always hold, our algorithm, which is based on Claim 1, will make false negative errors, i.e., it will miss some concepts. However, as we will see, the number of such false negatives will be relatively small.

2.4 Procedural Overview

To extract concepts, we evaluate the k-grams annotated with frequency as in Sec. 2.2, and extract the ones that we regard to be concepts. We maintain a preliminary set of k-grams and we *discard* the k-grams that we do not consider to be concepts. We proceed by evaluating k-grams in the order of increasing k (in a bottom-up fashion). If the k-gram x, for $k > 2$, is a concept, then Claim 1 states that at least one of the k–1-grams that x contains needs to be discarded, since both the k–1-grams cannot be concepts. We do the following while analyzing k-grams for a given k:

- Either we discard the k-gram, OR
- We discard some of the k–1-grams that the k-gram contains, and keep the k-gram. The k-gram *overrides* the k–1-grams that get discarded.

2.5 Indicators of a Concept

From the k-gram set above, we wish to decide which of them are concepts. In order to make this decision, we use certain indicators.

There are three indicators that we expect k-grams that are concepts to possess. Let us take a closer look at these indicators:

Indicator 1*: Frequent*

We expect the frequency of occurrence (as in Sec. 2.2) of a concept a, also called the support S_a, to be high, i.e., S_a is large for a to be a concept.

Indicator 2*: Better than sub/super-concepts*

There are two parts to this indicator: We want our concept to be "better" than sub-concepts and "better" than super-concepts.

Firstly, we expect a concept k-gram to be better than any k–m-grams that it contains, i.e., we want it to have high "confidence", as defined below. Secondly, we expect the k-gram representing the concept to be "better" than super-concepts, i.e., there is no k+1-gram (of all the k+1-grams that contain the k-gram) that is a better concept than the k-gram.

However, to evaluate both the parts of this indicator, it is sufficient to evaluate k-grams in increasing order of k, and ensure that they are "better" than the k–1-grams (i.e., sub-concepts). We provide a justification that this strategy is correct in Appendix A (after making some assumptions).

To decide if a k-gram is better than k–1-grams, we define a set of metrics. Let the k-gram be a. If $a = t_1 t_2 \ldots t_k$, then let $b = t_1 t_2 \ldots t_{k-1}$ be the *prefix* k–1-gram of a, and $c = t_2 t_3 \ldots t_k$ be the *suffix* k–1-gram of a. There are two measures of confidence: pre-confidence *pre-conf* C_{1a} and post-confidence *post-conf* C_{2a}, where $C_{1a} = S_a/S_b$, and $C_{2a} = S_a/S_c$. The *pre-conf* is nothing but the probability that t_k is seen given that we have already seen $t_1 t_2 t_3 \ldots t_{k-1}$.

We define min-confidence *min-conf* as the minimum of the *pre-conf* and the *post-conf*. The *min-conf* is useful because it is a lower-bound on the confidence we have on the k-gram relative to the k–1-grams. If a k-gram has its *min-conf* higher than a threshold (a function of k), then we would prefer keeping the k-gram as a concept than the k–1-grams that it contains (assuming that it satisfies other properties).

Similarly, max-confidence *max-conf* is defined as the maximum of *pre-* and *post-conf*. The metric *max-conf* is useful when a sub-concept can be followed by many possible phrases/words, e.g., 'John Lennon' (a well-known musician) would have a low *min-conf* because the *pre-conf* is very low (since 'John' can be followed by any number of "popular" surnames), however, the *max-conf* is high because the surname uniquely identifies the first name. Thus, if the *max-conf* is higher than a certain threshold (a function of k), we would prefer to still consider the k-gram to be a concept. Typically this threshold will be higher than that for *min-conf* for a given k.

Another useful metric that we define is the relative confidence, *rel-conf* of a k-gram. This metric is defined as

$$\textit{rel-conf}(\mathrm{a}) = \frac{\textit{min-conf}(\mathrm{a})}{\max(\textit{max-conf}(\mathrm{b}), \textit{max-conf}(\mathrm{c}))}$$

where a, b, c are as before. This metric captures our notion that for a k-gram to be a concept, it should be better than any of the k–1-grams that it contains, i.e., if the *rel-conf* is beyond a certain threshold, we have higher confidence in the k-gram than in either of the k–1-grams.

The reason we have this metric *in addition* to the *min-conf* metric is that it is not sufficient to look at *min-conf* to decide if a k-gram is worth keeping relative to the k–1-grams that it contains. A high *min-conf* does not mean that the k–1-grams are not concept-worthy relative to the k–2-grams that they contain, in fact they might be more so than the k-gram relative to the k–1-grams. A high *rel-conf* (greater than some value dependent on k) would ensure that we are not making a mistake by keeping the k-gram as a concept, and discarding the k–1-grams that it contains.

We outline the differences between our metrics and term recognition metrics in detail in Sec. 4.

Indicator 3*: Contains only portions of sentences that convey a single meaning or idea*

Note that the metrics outlined above may not be sufficient to differentiate between concept and non-concept k-grams for some data sources. Consider a dataset of news documents over the past year and consider the example, 'George Bush said yesterday'. This 4-gram is not a concept, but could still be evaluated highly by the metrics above. Commonly occurring portions of sentences, e.g., 'How do I' and 'What does this'. would be other examples of text fragments that could score well on the metrics but are not concepts.

We therefore need an evaluator that works on k-grams and returns whether or not they can be concepts based on whether or not they contain portions of sentences that do not form a single concise concept. Note that this check is not as complicated in the web query log case, because it is "dense" in concepts. For other datasets, we may need an extensive procedure to prune noise words.

2.6 Using the Indicators

We define Algorithm 1: *conceptExtraction* which takes as input the set of k-grams annotated with frequencies and proceeds to extract a set of concepts in a bottom-up fashion. Note that the procedure maintains a set of k-grams that will be discarded at the end of every phase (D).

Algorithm 1 calls Algorithm 2: *candidateConceptCheck* to check

if a k-gram satisfies the properties mentioned indicator 1 and 2 in Sec. 2.5. A k-gram that passes the check in Algorithm 2 is called a *candidate concept*.

Once the extraction of candidate concepts is complete, the procedure *containsStopWords* is called by the *conceptExtraction* procedure (lines 12-17) to eliminate candidate concepts that we do not return as concepts because they do not satisfy indicator 3.

We now describe the details of algorithm *candidateConceptCheck*. We will consider *containsStopWords* subsequently. In *candidateConceptCheck*, a k-gram is first checked to see if it has requisite support(a function of k) in line 4; i.e., the k-gram occurs frequently enough to be a candidate concept.

As per the procedure *candidateConceptCheck*, a k-gram, $k > 2$ is a candidate concept if it falls into one of three cases, corresponding to various values of *pre-conf* and *post-conf* of the k-gram. Either the candidate concept on the whole is "good" (case 1) or is formed by attaching words to the end or beginning of "good" candidate concepts (case 2/3). The first case (9-13) corresponds to both *pre-* and *post-conf* being very close to 1. This corresponds to the situation where neither of the k–1-grams contained in the k-gram are judged to be concepts. The *min-conf* and *rel-conf* in this case have to surpass some threshold (a function of k). In this case, we keep the k-gram and discard both the k–1-grams. 'Who framed roger rabbit' would be an example of such a concept. The second case (15-17) corresponds to the *post-conf* being very close to 1, and the *pre-conf* small. This corresponds to the case where the prefix k–1-gram is extracted as a concept by itself. We would then expect the *post-conf* to be larger than some threshold, and also larger than the *pre-conf* by a factor of *dominance-threshold* (> 1). We then discard only the non-concept suffix k–1-gram. An example of such a concept would be 'Home Alone 3'. The third case (19-21), e.g., 'Intel Research Labs', is symmetric to the second case.

Note that we do not consider the case where both *pre-* and *post-conf* are low. This is because of Claim 1 which states that a k-gram concept cannot be formed out of two k–1-gram concepts. The exception in the claim for $k = 2$ is absorbed into case 1, with the *min-conf* threshold set suitably low. In this case, both of the 1-grams are concepts, and hence cannot be discarded.

We prefer a bottom-up over a top-down approach for the algorithm because we do not want to deal with a exploding number of k-grams for larger k. Typically, k-gram concepts will occur with many other words both before and after, giving rise to many k+1-grams which contain the k-gram. If we are analyzing bottom-up, all of these can be discarded because the k-gram has already been isolated as a concept. If we were analyzing top-down, we can't discard any of the k+1-grams because we cannot be sure if there is a k-gram that will override them.

Procedure *containsStopWords* is used to prune candidate concepts that are not likely to be concepts because they violate the third indicator. For instance, they could

- begin or end with a conjunction, article or pronoun, e.g., by, for, and, an, that, from, because, where, how, yesterday or you
- begin or end with a verb
- contain no nouns

Thus this procedure detects special words (also called *stop words*) as in the first item in the list above, and also does part-of-speech (POS) tagging [6], a technique used to tag each word in a phrase with its part-of-speech, for the second and third items. Note that the example 'George Bush said yesterday' in Sec. 2.5 will be detected by the first item.

Note that POS tagging and advanced techniques are not required for query logs because they are a much more "refined" data source.

Algorithm 1 $conceptExtraction$: Extract k-grams from a set of k-grams and their frequencies

Require: $l \leftarrow$ set of k-grams with frequencies S_a
1: $C \Leftarrow \emptyset$ {Set of Candidate Concepts}
2: **for all** $k = 1$ *to* n **do**
3: $D \Leftarrow \emptyset$ {Set of Sub-concepts to discard}
4: **for all** $a \in l$, a is a k-gram **do**
5: $(D, Answer) =$ *candidateConceptCheck*(a, C, D)
6: **if** $Answer == true$ **then**
7: $C \Leftarrow C \cup \{a\}$
8: **end if**
9: **end for**
10: $C \Leftarrow C - D$
11: **end for**
12: $F \Leftarrow \emptyset$ {Set of Concepts}
13: **for all** $a \in C$ **do**
14: **if** *containsStopWords*$(a) ==$ *false* **then**
15: $F \Leftarrow F \cup \{a\}$
16: **end if**
17: **end for**
18: **return** F

Query logs typically involve queries having multiple concepts one after the other with no extraneous words. For query logs, *containsStopWords* just detects the first item in the list above (for example, to detect k-grams like 'Where can I find'). Additionally, since most queries do not form a grammatically correct sentence (or portion of it), some NLP techniques cannot be used. However, for the Google k-gram case, a complete POS check as part of the stop word procedure is an integral part of the non-concept pruning.

2.7 The Algorithm: Observations

In this section we state some observations about the algorithm and how the thresholds work together to ensure that the correct set of concepts are extracted. We then state a theorem about the correctness of our concept extraction algorithm. Intuitively, the theorem states that if only the concepts in the k-gram set satisfy the indicators in Sec. 2.5, then *conceptExtraction* will extract precisely the concepts. All proofs can be found in Appendix D.

OBSERVATION 1. *Whether a k-gram a is extracted as a concept or not does not depend on the order of processing of k-grams, assuming that the k-grams are processed only when processing of k–1-grams is complete.*

OBSERVATION 2. *If all other thresholds are fixed, and only the support threshold of k-grams for a certain k is reduced, then the number of k–1-gram concepts extracted can only decrease in number, and their precision can only increase.*

OBSERVATION 3. *If all other thresholds are fixed, and only the support threshold of k-grams for a certain k is reduced, then the number of k+1-gram concepts can only increase in number.*

OBSERVATION 4. *If all other thresholds are fixed, and only the support threshold of k-grams for a certain k is changed, then only the number of k–1, k, k+1, . . .-gram candidate concepts can change.*

THEOREM 1 (CORRECTNESS). *Suppose we are provided with a dataset of k-grams, $k = 1 \ldots r$, such that $c_1, c_2, \ldots, c_n$ are concepts and $s_1, s_2, \ldots, s_m$ are not concepts. If there exist functions f and g such that:*

- *1-gram concepts have frequency $\geq f(1)$, and 1-gram non-concepts have frequency less than $f(1)$.*
- *For $k \geq 2$, k-gram concepts c_i have frequency greater than or equal to the maximum of $f(k)$ and $g(k+1)\times$ frequency of any k+1-gram that contains c_i*

Algorithm 2 $candidateConceptCheck$: Check if a k-gram is a candidate concept

Require: $a \leftarrow k$-gram
Require: $C \leftarrow$ set of concepts
Require: $D \leftarrow$ discard set
$k \Leftarrow$ k-gram-size(a);
$b \Leftarrow$ k-gram-prefix(a)
$c \Leftarrow$ k-gram-suffix(a)
if [$S_a >$ *support-threshold*(k)] **then**
  if [$k = 1$] **then**
    return$(D, true)$
  end if
  if [*min-conf*$(a) >$ *min-conf-threshold*$(k) \wedge$
    rel-conf$(a) >$ *rel-conf-threshold*(k)] **then**
    if [$k > 2$] **then**
      $D \Leftarrow D \cup \{b\}; D \Leftarrow D \cup \{c\}$
    end if
    return$(D, true)$
  end if
  if [$b \in C \wedge$
    post-conf$(a) >$ *dominance-thres* $\times$ *pre-conf*$(a) \wedge$
    post-conf$(a) >$ *post-conf-threshold*(k)] **then**
    $D \Leftarrow D \cup \{c\}$
    return$(D, true)$
  end if
  if [$c \in C \wedge$
    pre-conf$(a) >$ *dominance-thres* $\times$ *post-conf*$(a) \wedge$
    pre-conf$(a) >$ *pre-conf-threshold*(k)] **then**
    $D \Leftarrow D \cup \{b\}$
    return$(D, true)$
  end if
end if
return$(D, false)$

- *For $k \geq 2$, k-grams s_i that are not concepts have frequency less than $f(k)$* or *less than $g(k+1)\times$ frequency of all k+1-grams that (a) contain s_i and (b) have frequency greater than $f(k+1)$, if they exist*

Then there exist threshold values such that the algorithm conceptExtraction will output precisely the concepts c_i.

2.8 Complexity of Algorithm

Let the total number of k-grams be n. If we sort all sets lexicographically, then the algorithm does a constant number of lookups (each of $O(\log n)$) per k-gram, giving a complexity of $O(n \log n)$. Performance can drastically improve using hashing.

3. EXPERIMENTAL RESULTS

We only describe our experimental analysis on the AOL query log [10]. Appendix F contains details of our experiments on the Google KGram [5] and the `del.icio.us` Tag Dataset [13] (also outlined in Sec. 5).

We do not evaluate our program on the time taken for extraction. This is because, in practice, the algorithm is not required to respond on a real-time basis; and can be used to enrich the back-end concept set whenever there is a new dataset available, for example, the day's query log. But we also note that the program took less than half an hour to run on the AOL query log dataset on a laptop running Ubuntu on Intel Centrino Pro 1.60 GHz, and is therefore reasonably fast. We also note that extraction of concepts of a certain size k can happen in parallel, thus the algorithm is parallelizable.

The goal of our experiments is to study how well our algorithm performs in terms of "precision" and "recall" on varying various parameters, and in comparison with other algorithms. Precision is measured by seeing how many of the concepts returned by the algorithm are actually concepts. We make use of Mechanical Turk [16], a human evaluation engine, and the Wikipedia concept list for this purpose. However, we cannot compute recall precisely since we do not know how many concepts exist in the world at any point of time. Instead, we use the number of concepts returned by our algorithm as a measure of recall. We call this quantity the *volume*.

We first describe the results of our algorithm on the AOL query log dataset. We then compare our results to two other algorithms when extracting the same number of concepts (i.e., same volume). Then, we study the precision of our algorithm when compared to other algorithms when we vary the volume of concepts extracted. We also study the performance of our algorithm on changing various thresholds. Finally, we study how our algorithm performs on changing the size of the input dataset.

Methodology

For the algorithms, we set the thresholds empirically using a small random sample of 30 k-grams for each k. We then used the thresholds to extract a certain number of concept k-grams. Some of these k-grams were already present in Wikipedia and are, therefore, correct concepts. For the remaining k-grams, to rate them against human intuition, we evaluated each independently by 3 unknown reviewers (with a high approval rate) on Mechanical Turk [16]. The reviewers were given the definition of concepts as in Sec. 2.1 and asked to identify k-grams that are concepts. The reviewers were also provided the search results from a popular search engine for the k-gram (thus helping reviewers judge concepts that they were not familiar with). We took the majority of the opinions of the reviewers — if two or more (out of 3) reviewers said that a k-gram is not a concept, we assumed that it is not a concept. (Note that the results are not very different if we assumed a k-gram is a concept iff all three reviewers said so.) We used these results to evaluate the algorithms on how precise the returned concepts are — measured in terms of absolute precision, i.e., (total 'correct' concepts/total concepts) and non-wiki precision, i.e., (total 'correct' non-Wikipedia concepts/total non-Wikipedia concepts). Non-wiki precision is also important because we already have the list of Wikipedia concepts at our disposal, and we would like to know how well the algorithms extract new concepts not found in Wikipedia.

Later on, when we evaluate the performance on varying parameters, we measure precision using the number of Wikipedia concepts extracted, i.e., precision is (total wikipedia concepts/total concepts). This makes it easier to evaluate our results automatically (for several result sets) and is a *lower-bound* on actual precision.

Parameter	k = 1	k = 2	k = 3	k = 4
support-threshold	100	35	60	60
min-conf-threshold	-	0.035	0.1	0.15
rel-conf-threshold	-	-	0.2	0.25
post-conf-threshold	-	0.1	0.15	0.2
pre-conf-threshold	-	0.1	0.15	0.2
dominance-thres	-	1.3	1.3	1.3

Table 2: Threshold Parameter Values.

Note that we did not use NLP as a filtering technique for any of the algorithms, given that the query log dataset is a refined data source, devoid of auxiliary words (which are necessary to make it a grammatically correct sentence). We did, however, use a simple stop word pruner which removes all concepts that begin with words such as 'who','that' and 'of', for all the 3 algorithms, as described in Sec. 2.6. Note that we do not evaluate 1-grams because all of them are likely to be concepts (if we eliminate stop-words).

Results

(1)*We ran our experiments on the AOL query log dataset, which contains 36M queries from different users and 1.5M unique terms.*

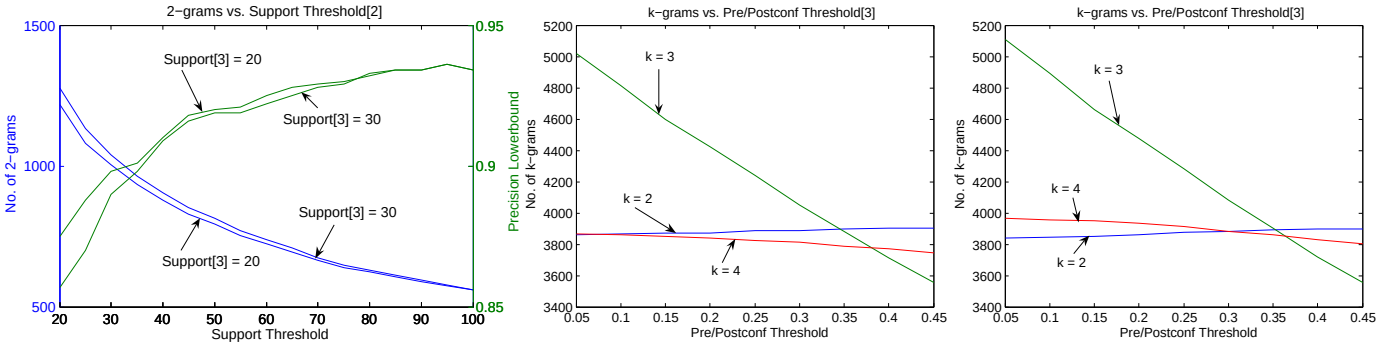

Figure 1: (a) Variation of Number of 2-grams and their precision, vs. Support threshold for 2-grams. (b) Variation of Number of k-grams extracted for various values of the *pre-/post-conf* threshold and *dominance-thres* = 1.5(c) Same as before with *dominance-thres* = 1 for all k: More 3-gram and 4-gram concepts

With parameters as listed in Table 2, a total of 25882 concepts were extracted by our algorithm, as in Table 3, with an absolute precision of 0.95 rated against Wikipedia and Mechanical Turk.

The thresholds in Table 2 were used to extract 25882 concepts. Out of these, 21711 concepts were already present in Wikipedia. The statistics are displayed in Table 3. The first column describes the total statistics for 2,3 and 4-grams, while the subsequent columns give the breakup independently for 2, 3 and 4-grams, and then 1-grams. The first row gives the total number of concepts extracted by the algorithm, the second row gives the number of Wikipedia concepts found in the concepts obtained in the first row. The third row gives the number of unevaluated concepts (i.e., those that are not Wikipedia concepts). The fourth row gives the number of concepts from the previous row found correct by the Mechanical Turk reviewers. The next two rows give the absolute precision (total 'correct' concepts/total concepts) and the precision on excluding concepts found in Wikipedia ((total 'correct' concepts - Wikipedia concepts)/(total concepts - Wikipedia concepts)). As is clear from the first column of Table 3, out of the 2303 unevaluated 2-, 3- and 4-grams, 1934 were found to be concepts by the reviewers. The absolute precision for extraction of 2-,3- and 4-grams is $(1934 + 4741)/(2303 + 4741) = 0.95$. If we exclude Wikipedia concepts, the precision is 0.84 (for 2-,3- and 4-grams). The mistakes made by our algorithm are 369 for 2-,3- and 4-grams out of a total of 7044 concepts. Note that the precision for each k is high as well.

Quantity	Total	k = 2	k = 3	k = 4	k = 1
Concepts	7044	4393	2251	400	18838
Wiki Concepts	4741	3401	1147	193	16970
Remaining	2303	992	1104	207	-
M.Turk Correct	1934	837	947	150	-
Total Precision	0.95	0.97	0.94	0.86	-
Non-Wiki Precision	0.84	0.84	0.86	0.72	-

Table 3: Our Algorithm.

Here are examples of extracted concepts not in Wikipedia (and verified by Mechanical Turk):"Who's On My Page" (a Myspace plugin), "Shih tzu puppies" (puppies of a certain breed), "Oriental trading company", "Weichert real estate", etc. Examples of extracted concepts rejected by Mechanical Turk include: "Find out if someone", "Whats left of me", "Sunshine of the spotless" (subconcept), "Get rid of ants".

(2)*Comparison with two other strategies (naive and C-Value) for the same volume of 2,3 and 4-grams (7000): our algorithm gave fewer errors (369) as compared to the Naive algorithm (997) and C-Value algorithm (557). Also, the absolute precision for our algorithm was 0.95, when compared to 0.86 for the Naive algorithm and 0.92 for the C-Value algorithm. The non-wikipedia precision was 0.84 for our algorithm, as compared to 0.66 for the Naive algorithm and 0.75 for the C-value algorithm.*

Quantity	Total	k = 2	k = 3	k = 4	k = 1
Concepts	7070	3669	2842	559	18838
Wiki Concepts	4131	2615	1261	255	16970
Remaining	2939	1054	1581	304	-
M.Turk Correct	1942	626	1101	215	-
Total Precision	0.86	0.88	0.83	0.84	-
Non-Wiki Precision	0.66	0.59	0.70	0.71	-

Table 4: Naive Algorithm.

To examine how a naive approach would work as compared to Algorithm 1, we implemented a concept extraction procedure that selects concepts based simply on the frequency of occurrence of the k-gram. We tuned the frequency thresholds to achieve high precision (on a random set of 30 k-grams) and ensure that approximately the *same* total number of 2-gram, 3-gram and 4-gram concepts were extracted. The results are in Table 4.

The extracted concepts were evaluated by reviewers on Mechanical Turk. The fraction of non-Wikipedia k-grams found to be concepts by Mechanical Turk is less: 1942/2939 = 0.67. By comparison, the precision for our algorithm for non-Wikipedia concepts is 1934/2303 = 0.84. Overall the mistakes made (ignoring 1-grams), is $2939 - 1942 = 997$, almost 2.7 times the number of mistakes made by our algorithm (369) for the same number of concepts.

Quantity	Total	k = 2	k = 3	k = 4
Concepts	7029	5091	1567	371
Wiki Concepts	4790	3730	872	188
Remaining	2239	1361	695	183
M.Turk Correct	1682	985	570	127
Total Precision	0.92	0.93	0.92	0.85
Non-Wiki Precision	0.75	0.72	0.82	0.69

Table 5: C-Value Algorithm.

To contrast our algorithm to a term recognition strategy that also uses frequency information as a metric, we picked the C-Value Method [15], and the results are in Table 5. The method does not extract 1-gram concepts. The threshold was set so that a similar total number of k-grams were extracted (sum of 2-, 3- and 4-grams) as concepts as in our algorithm above. The fraction of non-Wikipedia k-grams found by the algorithm was 1682/2239 = 0.75, marginally better than the naive algorithm given above. The number of mistakes made in total was 2239-1682=557, 1.5 times the number of mistakes made in our algorithm.

The poor results of the C-Value Algorithm are probably because the rules used in [15] are geared towards term recognition, and fail to recognize some Wikipedia-like concepts. Additionally, while our algorithm allows k-gram frequencies to affect whether k–n-grams and k+n-grams are extracted as concepts, the C-Value algorithm has a single top-down strategy, with k+n-grams affecting k-grams. Our algorithm, being based on statistics of Wikipedia concepts, reflecting true concepts, is therefore more sophisticated.

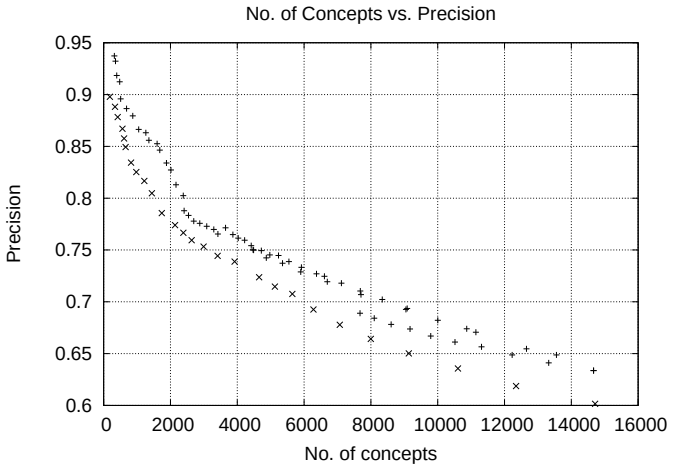

Figure 2: Variation of precision versus the volume of concepts extracted for the C-Value algorithm ($\times$) and our algorithm ($+$)

(3)*Variation of precision versus volume of concepts extracted using a scatterplot for the C-Value algorithm and our algorithm: Our algorithm gets better precision for all volumes.*
Note that the performance of our algorithm and the C-Value algorithm may vary as the total number of k-grams extracted as concepts changes. To check this variation, we plotted the precision (measured using the number of Wikipedia concepts) of each of the two algorithms versus the volume of concepts extracted in Fig. 2. Each point plotted on the figure corresponds to execution of one of the two algorithms with various threshold values, for instance, there is an execution of the C-Value algorithm that extracts around 15000 concepts and has a precision of 0.6, while there is a setting of threshold parameters for our algorithm which extracts the same number of concepts, but has a precision of around 0.64. It is clear from the figure that we can achieve higher precision for our algorithm than for the C-value algorithm for any number of concepts extracted. So for any value of the volume of concepts required, we can adjust threshold values in order to extract more precise concepts with our algorithm. This plot also demonstrates that setting the thresholds is relatively easy in our algorithm, because the data points in the scatter-plot were obtained by setting threshold values at 10 different starting configurations, and varying one threshold while keeping the rest fixed.

Now that we have established that our algorithm gives us more flexibility in getting more concepts for a fixed precision, we now try to study how the various thresholds and the size of the query log affect the precision and volume of concepts extracted for our *conceptExtraction* algorithm.

(4)*For our algorithm, Precision-Volume Tradeoff On Varying Support Threshold: Precision increases as Volume decreases on increasing support threshold.*
In Figure 1(a) we kept all other thresholds constant and varied the 2-gram support threshold and examined the number of 2-gram concepts extracted and the precision of those 2-gram concepts. We repeated the experiment for two values of the support threshold of the 3-grams, for 20 and for 30. As expected, the volume of 2-grams extracted reduces as the support threshold increases. Also, the precision increases as the threshold increases since the concepts extracted with a high threshold are likely to be correct. Note that the graph for volume with a lower support threshold for 3-grams is below the graph with the higher support threshold. Similarly, the graph for the precision for the lower support threshold for 3-grams is above that for the higher one. This effect is because of Obsvn. 2.

(5)*For our algorithm, Dependence of Volume (for various k) on Post-conf / Pre-conf / Min-conf Threshold: Thresholds for each k can be tuned independently of the others.*
Figure 1(b) and 1(c) depicts the variation of the number of various k-gram concepts with 3-gram *pre-/post-conf* threshold for fixed values of other parameters. The 3-grams are directly affected, and the graph falls sharply as the *pre-/post-conf* threshold is increased. The number of 2-gram concepts increases slightly because the number of 3-gram concepts has fallen, and the 2-gram concepts are less likely to be 'overridden' by a smaller number of 3-gram concepts. The number of 4-gram concepts, on the other hand, decreases slightly. This decrease is because the number of 3-gram candidate concepts has reduced in number, and by the second and third case of Algorithm 2, this would screen out more 4-gram concepts (since the 3-gram concepts would not pass the check). Note that the number of 2-gram and 4-gram concepts increase or decrease only slightly. Therefore, in practice, we can tune the *pre/post-conf* parameter independently for each k.

We repeated the experiment on varying the *min-conf* threshold and found similar behavior (i.e., that one could tune *min-conf* independently for each k). These figures can be found in Appendix E.

(6)*For our algorithm, Volume-Precision tradeoff with differing size of query log: Precision increases as Volume increases with increasing size of query log*: Intuitively, larger datasets give us more and better concepts. Details are given in Appendix E.

Setting the Thresholds

The performance of *conceptExtraction* hinges on whether or not the thresholds are set appropriately. In the experiments above we manually tuned the parameters to obtain good results. In this paper, we focus on using the metrics rather than on finding the optimal thresholds. We show how to set thresholds in Appendix G.

4. RELATED WORK

In this section, we briefly discuss work closely related to ours; additional work is surveyed in Appendix H.

Our work is closely related to the field of association rule mining [2], also called the market baskets problem. There are two steps to the solution of this problem: (a) extracting sets of items with sufficient support and (b) testing whether the items actually depend on each other. Agrawal et. al. [2] only deals with the first step, noting that the second step is straightforward in the item-sets case.

Our problem is related in the sense that our goal is to extract concepts that are groups of words that occur more together than with other words. We use metrics similar to support and confidence. But the ordering of the words imposes additional restrictions on our problem. Also, we wish to extract the "best" concept, and delete some sub- and super-concepts. In our case, the downward-closure property does not hold. Also unlike [2], we focus on the equivalent of the second step of the market baskets problem, given that we know the frequencies of all k-grams that are above a certain small threshold. The A-priori method may be used to determine 'frequent' word-sequences that have minimum support (the smallest support threshold over all k), i.e., step (a) above.

Term Recognition or Terminology Extraction [8, 19, 17, 14, 26] is a well studied problem in the field of Information Extraction. Superficially, the aim is similar: that of extracting the most meaningful and concise k-grams, i.e., terms from a document. These terms aid in indexing for textbooks, creation of thesauri, etc. However, we wish to extract popular terms, i.e., concepts, and our metrics are tuned towards extracting high precision concepts. Also, since our aim is to extract concepts that are Wikipedia-like, our algorithm has been selected in order to make use of Claim 1 (which is based on statistics of Wikipedia concepts).

The statistical metrics used for term extraction include Mutual

Information, log likelihood, cost criteria [17, 23] etc. For example, Mutual Information is a single quantity which measures the probability of two terms occurring together as against independently. This measure is similar to our metrics of *pre-* and *post-conf*. However, the term extraction metrics suffer due to the fact that they give an absolute score to a k-gram, which is not sufficient when we are concerned about extraction of concepts that are "good" relative to super- and sub-concepts. Claim 1 illustrates why we need both *pre-* and *post-conf* independently when looking for concepts. Also note that Mutual Information and the log likelihood measure would rank highly for a 'term' that has not been seen before in the dataset (and is therefore not popular). The cost criteria metric does not allow a concept to be part of a larger concept. It therefore does not apply for our purpose (see Sec. 2.3).

However, we found that the C-Value metric [15] incorporates frequency information in such a way that could be used to extract popular terms using a top-down scheme. But in our analysis in Sec. 3 we find that the C-Value metric does not perform as well as our algorithm in extracting concepts from data sources that are 'dense' in concepts.

We could also consider semantic methods: the extraction of concepts from text based on semantic relationships, stemming, part-of-speech tagging and other natural language processing tools. While our statistical approach can also be adapted seamlessly to free text (with the help of an NLP parser in *containsStopWords*), a semantic approach of determining concepts from text would not work in the case of query logs or tags. Thus the standard linguistic techniques [14, 19, 26] used in term extraction are not applicable.

5. CONCLUSIONS

In this paper, we considered the problem of extracting concepts from a large set of k-grams annotated with frequencies. We demonstrated that it is necessary to look at not just the frequency of the k-gram concerned, but also at the k–1-gram and k+1-gram frequencies when determining if a k-gram is a concept. We defined metrics to capture the indicators that we expect concepts to possess. We designed a one-pass algorithm to extract concepts using these metrics and showed that the algorithm is *correct* and possesses some desirable properties. We performed experimental analysis on the AOL dataset, measuring the volume-precision tradeoff on varying various parameters. The results were compared with those of a naive algorithm and a term recognition algorithm, displaying that our algorithm performs better than other techniques for the problem of concept extraction from large datasets. We also provided some intuition as to how thresholds may be tuned to get high precision and volume on a new dataset.

The experiments were also carried out on the Google k-gram dataset [5] (k-grams extracted from web-pages along with their frequency) and the `del.icio.us` tag dataset for one year. The Google k-gram dataset required an extensive POS tagger and evaluator to remove free text context and spam from the k-grams representing portions of concepts, but extracted as many as 0.25M concepts. The tag dataset gave rise to around 21500 concepts for the same thresholds in Table 2. However, most of the concepts were 1-grams, indicating that users tend to use short concepts to label URLs. Additionally, the precision values were less for the two datasets. For example, for the tag dataset, while 1-grams had a precision lower-bound of 0.85, 2-grams had 0.6 precision.

There are several avenues for future work in extracting useful information from query logs. Once concepts have been identified, the web query logs could prove to be helpful not only in finding metadata about concepts (for example, the search query "Red bull taurine" might help us identify that the "Red bull" energy drink contains taurine), but also finding relationships between concepts (for example, "Martin Scorcese" and "The Departed" are likely to appear together in many search queries). In particular, if our web of concepts is arranged as a taxonomy (as in Wikipedia), then we can use cues from search queries that the concept is present in to find the "nearest" parent of a concept, or examine queries similar to the one the concept is in to find siblings. We can then add our concept to the taxonomy at its appropriate location.

Also, it would also be interesting to see if the precision of our results can be improved even further, automatically, using information from the Web.

Acknowledgments: We would like to thank Digvijay Lamba for useful discussions, Paul Heymann for the tag dataset, and the reviewers for their suggestions.

6. REFERENCES

[1] http://en.wikipedia.org/wiki/wikipedia:criteria_for_speedy_deletion.
[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB 1994*.
[3] B. Gelfand et. al. Discovering concepts in raw texts: Building semantic relationship graphs. Technical report, 1998.
[4] I. Bichindaritz and S. Akkineni. Concept mining for indexing medical literature. *LNCS*, 3587, 2005.
[5] T. Brants and A. Franz. Web 1T 5-gram V1, 2006.
[6] E. Brill. A simple rule-based part of speech tagger. In *Ap. NLP 1992*.
[7] S. Brin. Extracting patterns and relations from the world wide web. In *WebDB, EDBT 1998*.
[8] D. Evans and C. Zhai. Noun-phrase analysis in unrestricted text for information retrieval. In *ACL 1996*.
[9] F. De Comit et. al. Positive and unlabeled examples help learning. In *Conf. on Alg. Learning Theory 1999*.
[10] G. Pass et. al. A picture of search. In *InfoScale 2006*.
[11] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation. *Data Min. Knowl. Discov.*, 2004.
[12] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992*.
[13] P. Heymann, G. Koutrika, and H. Garcia-Molina. Can social bookmarking improve web search? In *WSDM 2008*.
[14] C. Jacquemin and D. Bourigault. Term extraction and automatic indexing. *Handbook Of Comp. Linguistics*, 2003.
[15] K. Frantzi et. al. Automatic recognition of multi-word terms: the c-value/nc-value method. *Int. Journal on Digital Libraries 2000*.
[16] A. Kittur, E. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *CHI 2008*.
[17] K.Kageura and B. Umino. Methods of automatic term recognition: a review. *Terminology*, 3, 1996.
[18] M. Looks et. al. Streaming hierarchical clustering for concept mining. *Aerospace Conference, IEEE*, 2007.
[19] M. T. Castellvi et. al. Automatic term detection: A review of current systems. In *Recent Adv. in Comp. Terminology 2001*.
[20] A. Maedche and S. Staab. Mining ontologies from text. In *EKAW 2000*, London, UK.
[21] C. D. Manning and H. Schutze. *Foundations of Statistical NLP*. MIT Press, June 1999.
[22] N. Dalvi et. al. A web of concepts. In *PODS 2009*.
[23] P. Pantel et. al. A statistical corpus-based term extractor. *AI 2001*.
[24] R. Agrawal et. al. Mining sequential patterns. In *ICDE 1995*.
[25] R. Jones et. al. Generating query substitutions. In *WWW 2006*.
[26] S. Loh et. al. Concept-based knowledge discovery in texts extracted from the web. *SIGKDD Explor. Newsl.*, 2(1), 2000.
[27] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *SIGIR 1999*.
[28] M. Seno and G. Karypis. Finding frequent patterns using length-decreasing support constraints, 2001.
[29] K. Wang, Y. He, D. W. Cheung, and Y. L. Chin. Mining confident rules without support requirement. In *CIKM 2001*.
[30] Q. Yang and H. H. Zhang. Web-log mining for predictive web caching. *TKDE.*, 15(4), 2003.

APPENDIX

In the appendix, we give details for several items that were not considered in the main body of the paper, including

- Justification for evaluating sub-concepts in Appendix A
- Details of the Algorithm in Appendix B
- Complexity in Appendix C
- Proofs of Theorems/Observations in Appendix D
- Additional Experiments for Query Logs in Appendix F
- Experiments on other Datasets in Appendix F
- Setting the Thresholds in Appendix G
- Additional Related Work in Appendix H

A. JUSTIFICATION FOR APPROACH

In this section, we provide justification for why our technique of evaluating bottom up, and comparing k-grams with sub-concepts is sufficient in order to evaluate indicator 2 given in Sec. 2.5. We provide this justification by proving the correctness of our approach while making some assumptions.

Given a dataset, we will define concepts to be all the k-grams in the dataset that are "better" than k–1-grams (sub-concepts) and k+1-grams (super-concepts). Thus, all k-grams in our setting have sufficient support to be a concept, and do not contain portions of sentences (i.e., they satisfy both indicator 1 and 3). Therefore, all 1-grams are concepts.

Let c be a k-gram ($k > 2$). Let b_1 and b_2 be the two k–1-grams that c contains, and d_1, d_2, ..., d_m be all the k+1-grams that contain c. Let function $\gg$ take two word sequences and return either true or false. (Intuitively, $a \gg b$ says that a is a "better concept" than b.) We define c to be a concept iff

- $c \gg b_1$ or $c \gg b_2$ (i.e., c is better than sub-concepts) AND
- $\forall i \in \{1, \ldots, m\}$ it is not true that $d_i \gg c$ (i.e., c is better than super-concepts)

For 2-gram concepts, only the second item is true. (Recall that Claim 1 states that 2-gram concepts can contain two 1-gram concepts.)

We now suggest a bottom-up procedure that at every iteration evaluates k-grams relative to k–1-grams. Consider the following procedure *Simple*:

- **0:** Let set S contain all 1-grams and 2-grams
- **1:** For k from 3 to m
  - **2:** For all k-grams: (Let c be a k-gram, containing two k–1-grams b_1 and b_2.)
  - **3:** If $c \gg b_1$, add c to S, remove b_1 from S.
  - **4:** If $c \gg b_2$, add c to S, remove b_2 from S.

We now prove that *Simple* extracts precisely the concepts from the dataset.

THEOREM 2 (CORRECTNESS OF SIMPLE). *If we define concepts as given above, at the end of iteration k, the i-grams ($\forall i : 1 \leq i \leq k-1$) still present in S are precisely the concept i-grams.*

PROOF. We use an induction argument. First, note that the theorem is trivially true at the end of iteration 1 and iteration 2. Consider the end of iteration 3. We first prove that all 2-gram concepts a are in S. Since a is a concept, there exists no 3-gram c that contains a such that $c \gg a$. Therefore, the condition in line 3 and 4 would never be true, and a is never removed from S. We also need to prove that no extraneous 2-grams are in S. If the 2-gram a is not a concept, then there is a c in the 3-gram set for which $c \gg a$, in which case a would be removed from S.

Now let us assume that the theorem holds at the end of iteration m, we wish to prove that it holds at the end of iteration m+1. By the induction hypothesis, all $1, 2, \ldots, m$–1-grams still in S are precisely the concepts among the $1, 2, \ldots, m$–1-grams. Now we need to prove the same for m-grams. We first prove that all m-gram concepts a are in S. Since a is a concept, (i) there is a m–1-gram b that a contains such that $a \gg b$ (ii) there exists no m+1-gram c that contains a such that $c \gg a$. Due to (i), a is added to S in iteration m via line 3 or 4. Additionally, due to (ii), at iteration m+1, the condition in line 3 and 4 never evaluates to true, and a is never removed from S. On the other hand, if a is not a concept, then either a was not added to S at iteration m, or there is a c for which $c \gg a$, in which case a would be removed from S at iteration $m+1$. Thus, at the end of iteration m+1, the $1, 2, \ldots, m$-grams still in S are precisely the concepts among the $1, 2, \ldots, m$-grams. □

B. ALGORITHMIC DETAILS

In this section, we provide additional details for procedures *candidateConceptCheck* and *containsStopWords* from Sec. 2.6 that we did not cover in the main body of the paper due to space constraints.

As per the procedure *candidateConceptCheck*, a k-gram ($k > 2$) is a candidate concept if it falls into one of three cases. These three cases (lines $8-13, 14-17, 18-21$ in Algorithm 2) correspond to various values of *pre-conf* and the *post-conf* of the candidate concept. Either the candidate concept on the whole is "good" (case 1) or is formed by attaching words to the beginning or end of "good" concepts (case 2/3). The first case ($8-13$) corresponds to both *pre-* and *post-conf* being very close to 1. This case corresponds to the case when neither of the k–1-grams are extracted as concepts. The second case (14-17) corresponds to the *post-conf* very close to 1, and the *pre-conf* small. This corresponds to the case where the prefix k–1-gram is extracted as a concept. The third case is symmetric to the second case (replacing b with c, *pre-* with *post-*, and so on) and we do not discuss it further.

In the first case, the k-gram is evaluated using two checks: the first of which says the least amount of confidence that we have in the candidate concept (i.e., the *min-conf*) should surpass some threshold (which is a function of k). The second check says that the k-gram should be "better" than its sub-concepts - defined by the fact that *rel-conf* surpasses a threshold. If a k-gram passes these two checks, then we are better off keeping the given k-gram in our set of extracted concepts than any of the 2 sub-concepts; and thus we discard both the sub-concepts. Consider, for example, 'Who Framed Roger Rabbit', which has a high *min-conf* and a high *rel-conf*. We discard 'Who Framed Roger' and 'Framed Roger Rabbit', and retain 'Who framed Roger Rabbit'.

The second case involves concepts of the form <name> <topic>, where topic refers to a concept that forms a kind of 'category' (example: Intel / Microsoft Research Labs) or <topic> <name> (example: Home Alone I/II/III (These are children's movies.)). Here, the *pre-conf* is likely to be much higher than the *post-conf* (in the algorithm, we would want *pre-conf* to be $>$ *post-conf* $\times$ *dominance-thres*, where *dominance-thres* > 1) - because the <name> field can be changed. We also expect the *pre-conf* to be very close to 1. Since the *pre-conf* is very high, we can effectively discard the prefix k-gram and keep the k-gram as a candidate concept. However, we retain the suffix, because that could still be extracted as a concept. The suffix must already be a candidate concept, since it has been examined before. <topic> <name> is dealt with in the same way. These cases are reflected in the algorithm.

Now, a few remarks about the *containsStopWords* procedure.

While the *containsStopWords* check may eliminate some actual concepts, we prefer to have higher precision (which is the case when we discard k-grams of the above form) rather than higher recall (more concepts, including those with stop-words). The trade-off is strongly in favor of precision, because the loss in precision is drastic if we allow k-grams of the above kind to be concepts. Not having this check essentially allows portions of sentences to be concepts simply because the words involved occur often together — yet may not represent a real-world entity.

Note that we only run the *containsStopWords* check (and prune candidate concepts that do not pass the check) after the first phase of extraction (lines 2-10 of *conceptExtraction*). This is because the candidate concepts could still be part of a larger concept, and may be required for correct screening of the larger concepts. However, we could pass the candidate concept set of size until k through to the *containsStopWords* check when we have moved to the processing of the k+2-grams in the first phase.

C. COMPLEXITY DETAILS

If we sort the k-gram set lexicographically, searching is $O(\log n)$. Per k-gram, we do a lookup of the frequencies of two k–1-grams and three k–2-grams, which amounts to 5 lookups. In addition, we may need to check if the two k–1 grams are candidate concepts in the candidate concept set (or discard them), which may incur an additional $O(\log n)$ each. Thus the worst case number of lookups that we may need to perform per concept are $7 \log n$, and the overall complexity of the one-pass algorithm is $O(n \log n)$. Our performance can drastically improve if we use hashing to do the lookups.

D. PROOFS

In this section, we give proofs for the results listed in Sec. 2.7.

PROOF OF OBSERVATION 1. Whether or not a k-gram is added as a candidate concept depends on the set of k–1-grams. As per algorithm 1, we do not discard the k–1-grams that get 'overridden' until we are done processing all the k-grams. Thus the order of processing of the k-grams is unimportant. □

PROOF OF OBSERVATION 2. Lower the k-gram support threshold, more the number of k-gram candidate concepts will pass the support condition, and the number of k–1-grams that get overridden and discarded will increase (because they are sub-concepts of the new k-gram candidate concepts). The k–1-grams that remain are only the exceptionally good ones (implying an increase in precision). Hence the result. □

PROOF OF OBSERVATION 3. Lower the k-gram support threshold, more the number of k-gram sub-concepts, more the number of k+1-gram candidate concepts that will be under consideration via lines 14-21, and more of them will get accepted as concepts. Hence the result. □

PROOF OF OBSERVATION 4. When we are analyzing k-grams, the k–2-gram candidate concept set (and below) has already been fixed, and hence cannot be changed. k–1-grams are affected because they may be discarded. k+1-grams are affected because the k-gram set is consulted to see if a k+1-gram is a candidate concept or not. Similarly, k+2-grams are affected because the k+1-gram set is consulted to see if a k+2-gram is a candidate concept or not, and so on. □

Note that in the theorem of correctness, whose proof we give below, we do not need to specify the frequency of concepts relative to sub-concepts, just relative to the context (i.e., the k+1-grams that the k-gram appears in).

PROOF OF THEOREM 1. Consider *rel-conf-threshold* to be 0 for all k. Let *min-conf-threshold(k)* = *post/pre-conf-threshold(k)* = $1/g(k)$. Let the *support-threshold* = $f(k)$. We let *dominance-thres* be 1. We also let the *containsStopWords* check returns false for all k-grams, since we can make out if a k-gram is a concept or not by looking at just the statistical properties. We use induction for the proof. We start with k = 1, and argue that the *conceptExtraction* algorithm precisely outputs the concepts and discards the non-concepts.

For $k = 1$, as per the algorithm, if the support threshold is met, it is a concept, else not. This set is the same as the 1-gram concept set matching the specification above. Since none of the 1-grams are discarded (while analyzing 2-grams) in the algorithm, the same 1-grams are output as concepts.

Now consider analysis of k-grams ($k \geq 2$) by the algorithm. By the end of this phase, the k–1-gram concept set will no longer be modified, so we analyze the k–1-grams that are output as concepts. We assume, as the induction hypothesis, that all up-to k–2-gram concepts have been correctly extracted.

If a k–1-gram a is a concept, then we prove the following:

- It is not discarded at the k–1-gram stage
- It is not discarded at the k-gram stage

At the k–1-gram stage, the k–1-gram passes the support threshold, either contains one k–2-gram that is a concept, or no k–2-grams that are concepts (from Claim 1). In the first case, the *candidateConceptCheck* procedure proceeds to lines 14-17 (or 18-21, equivalently). The prefix k–2-gram is a candidate concept, the *post-conf* $>$ the *pre-conf* (since the threshold is 1) — because the concept k–2-gram occurs more frequently than $g(k-1) \times S_a$, while the other k–2-gram occurs less frequently than $g(k-1) \times S_a$. Also, the *post-conf* is greater than the threshold $1/g(k-1)$. Thus a will not be discarded at the k–1-gram stage. In the second case, no k–2-gram is a candidate concept. In this case the *min-conf* will be greater than threshold $1/g(k-1)$, and hence a will not be discarded.

At the k-gram stage, any k-gram b that the k–1-gram concept a appears in has a *pre/post-conf* (as the case may be) of S_b/S_a, which is definitely less than $1/g(k)$, and hence the k–1-gram is not discarded.

Thus all the concepts are definitely not discarded.

If a k–1-gram is not a concept and is not contained in any k-gram, then it does not pass the support threshold, and hence is discarded by the algorithm. If the k–1-gram is not a concept but is contained in some k-gram, then:

- Either it does not pass the support threshold (in which case the algorithm discards it at the k–1-gram phase),
- Or, the algorithm accepts the k–1-gram in the k–1-gram phase. But the frequency of the k–1-gram is smaller than $g(k) \times R$, where R is the smallest frequency of all k-grams containing the k–1-gram having frequency $> f(k)$. In this case, one of two things can happen for any given k-gram b (with frequency $> f(k)$ — since only those k-grams can pass the support threshold):
  - The other k–1-gram contained in b is also a non-concept
  - The other k–1-gram contained in b is a concept

  In the first sub-case, the two k–1-grams in b give rise to a *min-conf* for $b > 1/g(k)$, thus the algorithm (correctly) discards both the k–1-grams (lines 8-13). In the second case, we look at lines 14-17 or 18-21. Let us assume that the second k–1-gram (the one that is a concept) is a prefix k–1-gram. In this case the k-gram has a high post-confidence $> 1/g(k)$ (higher than the pre-confidence, which is $< 1/g(k)$), and the non-concept k–1-gram is discarded.

Thus all the non-concepts are definitely discarded by the algorithm either at the k–1-gram phase or the k-gram phase.

Hence the algorithm correctly extracts the concepts and discards the non-concepts. □

E. QUERY LOG EXPERIMENTS (CONTD.)

We now describe in detail additional experiments that we performed on the AOL query log set.

(5)*For our algorithm, Dependence of Volume (for various k) on Min-conf Threshold: Thresholds for each k can be tuned independently of the others*.

Figure 3(a) is identical to Figures 1(b) and 1(c) except that we vary the *min-conf* thresholds. This figure shows that we can tune the *min-conf* thresholds independently for each k. We changed the 4-gram *min-conf* threshold and plotted the results on the same graph. Also note that while the number of 2-gram concepts do not change on changing the 4-gram threshold, the number of 4-gram concepts reduce on increasing the threshold, and 3-gram concepts increase, as expected. Thus the *min-conf* thresholds can be tuned independently as well.

(6)*For our algorithm, Volume-Precision Tradeoff with differing size of query log: Precision Increases as Volume Increases with increasing size of query log*.

In Figure 3(b) and 3(c) we ran the algorithm (having fixed the thresholds) on segments of the query log of increasing size for 3- and 4-grams respectively. We found that while the number of concepts (i.e., the volume) increased as the query log size increased (because more and more k-grams cross the thresholds), the precision does not increase after a point, because all the Wikipedia concepts have been found.

F. ADDITIONAL EXPERIMENTS

We now briefly outline our experiments on the Delicious tag dataset and the Google K-gram dataset.

F.1 Experiments on Web KGrams (Summary)

In addition to looking at the AOL query logs, we also ran our algorithm on Google's Web KGram Dataset [5]. KGrams from the Web are more prone to spam and errors from portions of sentences in free text and thus special rules to eliminate such candidate concepts were required. A powerful POS tagger was used in our *containsStopWords* procedure to eliminate portions of sentences. After this, (and after eliminating those found in Wikipedia) around 0.05M 2-gram concepts and around 0.2M 3-gram concepts were extracted. The dataset had around 300M 2-grams and around 900M 3-grams in total.

Since it is not feasible (both practically and monetarily) to evaluate all the concepts extracted by our algorithm, we instead evaluated a random sample of concepts, wherein the accuracy was found to be around 70%.

F.2 Experiments on Tag Dataset (Summary)

User submitted tags for delicious (`http://del.icio.us`) are another useful source for k-grams. We repeated our experiment on the Delicious Dataset [13], for the period from June 2007 to March 2008. Similar thresholds were used as in Sec. 2. The experiment extracted 20302 1-gram concepts, with 85% found in Wikipedia. But the 2-, 3- and 4- gram concepts extracted were fewer: 396, 334 and 480 respectively. 60% of the 2-grams were Wikipedia topics, while 15% of the rest were Wikipedia topics. The reason that a smaller number of concepts were extrated is probably that the del.icio.us interface encourages users to type in all the tags for a URL together on one line, not separated by punctuation. Hence the smaller concepts are harder to extract when placed next to other frequently occurring concepts.

Clearly, tagging sets are less useful than query logs for long concepts because people tend to tag items with as few 'disjoint' words as possible. Most useful tags are therefore, 1-grams.

G. SETTING THE THRESHOLDS

The thresholds that we use in our algorithm *candidateConceptCheck* needs to be tuned inorder to extract concepts with a high level of precision. To select good thresholds, one can use the following techniques:

•*Machine Learning Approach:* We could use machine classification to 'learn' optimal thresholds for the *candidateConceptCheck* algorithm, in order to be able to classify better what is a concept and what is not. In our case, the problem is that of learning from positive and unlabeled examples (also called PU-learning [9]). The positive examples include k-grams that we know to be concepts from sources such as Wikipedia article titles and encyclopedias. The machine learning algorithm should try to extract or approximate a set of reliable negative training examples and then apply a classification algorithm to approximate the optimal thresholds. We plan to explore this direction in subsequent work.

•*Manual Tuning Approach:* We can also tune the thresholds manually. On a new dataset, we could do the following: Figures 1(b) 1(c) and 3(a) indicate that parameters may be tuned separately, independent of each other. Thus we could fix all parameters at some starting value, change thresholds for one k, note the response (either by looking at the Wikipedia set, or by random sampling and evaluation). We then use that as the starting point for tuning thresholds for the next k. We set all thresholds in this manner in increasing order of k. However, note that the threshold *dominance-threshold* will need to be tuned globally.

For a given k, to set the *pre-/post-conf*, *rel-conf*, *min-conf* and support thresholds, we do the following: Assume that d% of the k-grams are concepts (We can quickly determine the value d by looking at a random sample of the k-grams). Set the support threshold to the value such that for a random sample of the log, $(d + \Delta)$% of that sample has frequency above this value. Then consider all the k-gram Wikipedia concepts that are above the support threshold, and choose the smallest values for the *pre-/post-conf*, *rel-conf* and *min-conf* thresholds such that the same Wikipedia concepts are selected as concepts. For those Wikipedia concepts that have sub-concepts (since we have tuned the k–1-gram thresholds beforehand), depending on the ratio *r* = *pre-conf/post-conf*, we select thresholds such that the Wikipedia concepts pass the check in line 8 (if $r \approx dominance\text{-}threshold$), line 14 ($r$ is smaller) or 18 (r is larger) in Algorithm 2. We retain one of the k–1-grams as sub-concepts if r or $1/r >$ *dominance-threshold*. Else we discard both k–1-grams.

We note that a lower *min-conf* threshold needs to be accorded to the 2-grams relative to the 3-grams because a given word may be followed by many other words giving rise to a number of concepts. Basically, seeing the 1-gram gives you less indication as to what the 2-gram might be.

H. ADDITIONAL RELATED WORK

We now describe in detail some other prior work which is not as closely related as those in Sec. 4 and details for some that were briefly outlined in Sec. 4.

[28] describes an approach to extract item-sets with length decreasing support-thresholds, which is true in our case (as k in-

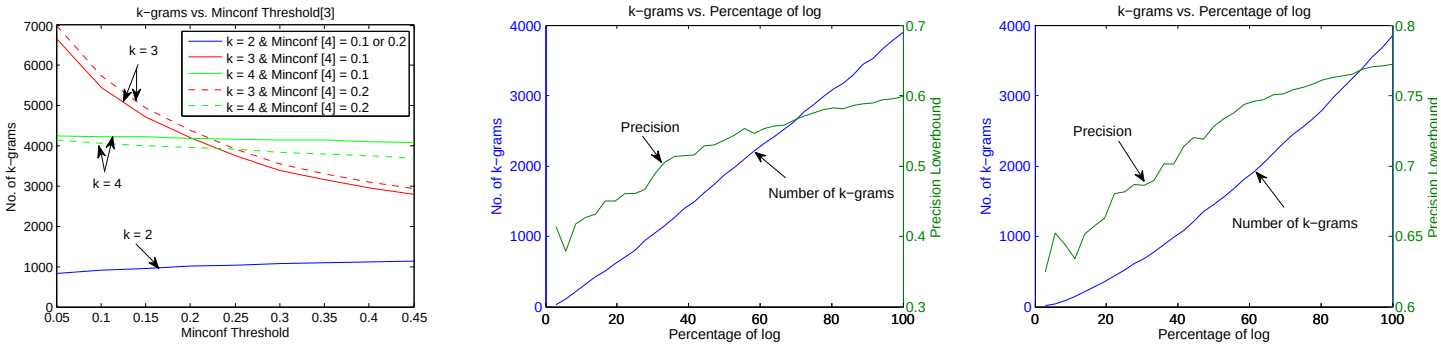

Figure 3: (a) Variation of Number of k-grams extracted for various values of the *min-conf* threshold for 3-grams (b) Variation of number of 3-grams extracted and precision vs. percentage of log exposed (c) Same as before for 4-grams

creases, the k-gram support threshold decreases), however, the algorithm involves use of the FP-tree method [11], which cannot be immediately applied to find all k-grams (which are word sequences, not sets) with a support greater than some $f(k)$, since the subset property does not hold. Adapting the FP-tree method for our purposes remains an open problem. Additionally, we focus on the second part of the frequent-itemset problem, i.e., how the confidence in an k-gram relative to other k'-grams affects our choice of the k-gram as a concept.

[29] describes a method to mine confident association rules without a support requirement. While we could use this method to prune k-grams that we do not have sufficient confidence in, even this work is not immediately applicable to our case, because the universal-existence upward closure property as in [29] does not hold. Additionally we have multiple confidence metrics and values of thresholds for those metrics, as against one *min-conf* threshold in [29].

Another related topic of research is sequence mining [24] which deals with mining sequences of items to find items that occur one after the other in sequence, in order to isolate temporal dependencies between items, which is very different from our goal, which is to find sequence of words that form concepts, and the procedures used are vastly different.

Generating hierarchies from documents has been explored in the past [27, 20], however, these works are limited by the fact that they use natural language processing to extract terms from grammatically correct sentences, and therefore cannot be applied to generic datasets like tags and query logs.

In [3], the authors use SRGs (Semantic Relationship Graphs) to represent semantic relationships between words extracted from text. Popularity is not used as a criteria to extract these words. [18, 12] deal with determining relationships, groupings etc. between terms extracted from text. We do not discuss relationships between concepts - or even the conditions of hypernymy or metronymy in this paper.

[7, 4] talk of extracting items (and then relationships between those items) based on bootstrapping with pre-defined (or subsequently discovered) patterns appearing in text. We, on the other hand, do not have a seed set of patterns or items to begin with, and our domain is not restricted to select types, the set of book-author pairs or symptom-cause pairs, respectively in the two references above. Our approach is general and does not require grammatically correct sentences as input.

The problem of named entity recognition [21] has as its aim detection and identification of 'named entities' — cars, persons, institutions, dates etc. Recognition is done via part-of-speech tagging and other natural language processing techniques. While we extract concepts of all kinds, named entity recognition can detect and extract concepts of specific types. However, named entity recognition cannot be performed on datasets such as query logs and tag data.

Building association rules on text k-grams is not new; [30] uses association rules between text k-grams and web pages that are clicked on for those k-grams.

Our work is also related to the field of word-level k-gram language modeling [21]: the *post-conf* metric we use is nothing but the marginal probability of seeing the last word given the previous k–1 words. However, we measure both forward and backward probabilities, i.e., that of seeing the first word given the last k–1 words as well. Additionally, *rel-conf* metric has no counterpart in the language modeling field.