Modeling Query-Based Access to Text Databases

Eugene Agichtein   Panagiotis Ipeirotis   Luis Gravano

Columbia University

{eugene,pirot,gravano}@cs.columbia.edu

ABSTRACT

Searchable text databases abound on the web. Applications that require access to such databases often resort to querying to extract relevant documents because of two main reasons. First, some text databases on the web are not "crawlable," and hence the only way to retrieve their documents is via querying. Second, applications often require only a small fraction of a database's contents, so retrieving relevant documents via querying is an attractive choice from an efficiency viewpoint, even for crawlable databases. Often an application's query-based strategy starts with a small number of user-provided queries. Then, new queries are extracted –in an application-dependent way– from the documents in the initial query results, and the process iterates. The success of this common type of strategy relies on retrieved documents "contributing" new queries. If new documents fail to produce new queries, then the process might stall before all relevant documents are retrieved. In this paper, we develop a graph-based "reachability" metric that allows to characterize when an application's query-based strategy will successfully "reach" all documents that the application needs. We complement our metric with an efficient sampling-based technique that accurately estimates the reachability associated with a text database and an application's query-based strategy. We report preliminary experiments backing the usefulness of our metric and the accuracy of the associated estimation technique over real text databases and for two applications.

1. INTRODUCTION

Searchable text databases abound on the web. Applications that require access to such databases often resort to querying to extract relevant documents because of two main reasons. First, some text databases on the web are not "crawlable," and hence the only way to retrieve their documents is via querying. Second, applications often require only a small fraction of a database's contents, so retrieving relevant documents via querying is an attractive choice from an efficiency viewpoint, even for crawlable databases. Various query-based methods (e.g., [4, 1]) have been proposed in the past for retrieving and extracting the information stored in databases via querying. These algorithms share the general approach of starting with a small set of queries, retrieving some documents from the database, extracting some information from them, and potentially augmenting the set of queries using the newly extracted information.

More specifically, there are two tasks that have been successfully addressed through querying: extracting information from text databases, and building text database summaries.

Task 1: Information Extraction: This is the task of extracting *structured* relations from *unstructured* (i.e., natural language) text. The structured relations can be used for answering queries or for data mining tasks. For example, a user may be interested in automatically building a structured table of reported disease outbreaks *DiseaseOutbreaks(diseaseName, country, date)* from news articles. Unfortunately, exhaustively scanning every news article for potential events of interest is (unnecessarily) slow. It has been shown that querying can be used to significantly improve the efficiency of information extraction [1]. The goal is to extract all tuples for a target relation from a database, starting by using a few seed tuples as queries and then successively querying for each newly extracted tuple. Intuitively, the documents that contain a query tuple are expected to contain other previously unseen tuples as well:

```
(0)  while seed has an unprocessed tuple $t$
(1)      retrieve up to MaxResults documents matching $t$
(2)      extract new tuples $t_e$ from these documents
(3)      augment seed with $t_e$
```

Thus, new tuples are discovered by querying for the known tuples, and when the algorithm terminates, all of the tuples "reachable" from the *seed* tuples will be discovered. In [1] it was observed that while in some cases this simple "bootstrapping" strategy –referred to as *Tuples* in [1][1]– succeeded in reaching most of the useful documents in the database (and thereby in extracting most of the tuples in the target relation), in other cases this algorithm only discovered a small fraction of the available tuples. As another example, Shah [10] uses a query strategy related to *Tuples* to extract people's names and build networks of "experts." In this work we identify and analyze the intrinsic property of text databases with respect to an extraction task that determines the success of this general approach. But first, we present a related task that can also be modeled using our techniques.

Task 2: Text Database Summary Construction: Many valuable text databases on the web have non-crawlable contents that are "hidden" behind search interfaces. Metasearchers are helpful tools for searching over many such databases at once through a unified query interface. A critical task for a metasearcher to process a query efficiently and effectively is the selection of the most promising databases for a query, a task that typically relies on statistical summaries of the database contents. Unfortunately, web-accessible text databases do not generally export summaries of their contents. In the past, query-based algorithms have been proposed to automatically build such summaries for web-accessible databases [4, 8]. The goal is to construct an augmented dictionary of all words that appear in the database, and their frequency. One of the algorithms described in [4] automatically discovers the content of a text database by first querying the database with some seed words, and then extracting new words from the retrieved documents to construct new queries. A somewhat simplified version of this algorithm (e.g., the stopping condition in [4] is different, for execution efficiency) is as follows:

Copyright is held by the author/owner.
International Workshop on the Web and Databases (WebDB).
June 12–13, 2003, San Diego, California.

[1]Reference [1] introduces alternative querying strategies –notably *QXtract*– that do not follow this general structure.

```
(0) while seed has an unprocessed word $t$
(1)     retrieve up to MaxResults documents matching $t$
(2)     extract new words $t_e$ from these documents
(3)     augment seed with $t_e$
```

The output of this algorithm is a set of words and their approximate frequency in the database. If the extracted summary is incomplete, the accuracy of the database selection step might suffer, and so will the overall effectiveness of the metasearching process.

We observe that the querying algorithms for both *Task 1* and *Task 2* share a key characteristic: only the documents that are "reachable" from the initial queries will be discovered by these algorithms. In this paper, we present a querying model that describes the general querying approaches used for *Tasks 1* and *2*. In Section 2 we model the two tasks using our "reachability" graph formalism, and derive a single *reachability* metric based on the connectivity of this graph, to predict the success or failure of a general class of algorithms for *Tasks 1* and *2*. We illustrate the construction of the reachability graph for real text databases in Section 3. Then, in Section 4, we provide an efficient sampling-based technique for estimating the reachability of a given database. We evaluate our estimation techniques in Section 5 over real text databases for the tasks described above, showing that we can successfully estimate the reachability of a database by examining only a small document sample from the database. We conclude the paper in Section 6 with discussion of future work and potential applications of our techniques.

2. MODEL

In this section we present our model of query-based access to text databases. We first provide the intuition behind our model using *Tasks 1* and *2* (Section 2.1), and develop this intuition into a model using the "reachability" graph formalism (Section 2.2). We then present a key observation about the general structure of the reachability graphs that emerge when querying text databases (Section 2.3), which will lay a foundation for our estimation techniques of Section 4.

2.1 Querying Text Databases Revisited

The common characteristic of the "bootstrapping" approaches that have been applied to *Tasks 1* and *2* is that they start with a small set of *seed* queries, and use the information extracted from the retrieved documents for additional querying.

For example, consider the *Task 1* scenario, illustrated in Figure 1, and assume that the initial set of seed tuples consists of one tuple, t_1. The database is queried using t_1 as described in Section 1. As a result, the document d_1 is retrieved. From this document, we extract the new tuple t_2. After adding t_2 to the *seed*, and sending it as a query to the database, we extract the new tuple t_3, which in turn retrieves the document d_4 that contains the tuple t_4. Note that t_3 also retrieves the document d_2, which "rediscovers" the tuple t_2. As we can see, the tuples t_2, t_3, and t_4 are all reachable from t_1, as shown in the resulting "reachability" graph on the right side of Figure 1. In contrast, tuple t_5 is not reachable from any of these tuples. As a result, t_5 will not be discovered by the algorithm if t_1 is the only "seed" tuple.

The procedure described above can be summarized using the graphs in Figure 1. By analyzing the structure of these graphs we can get a better understanding of the process and predict whether a particular query-based access method can succeed in reaching all (or a significant fraction of) the content of interest stored in the database. With this goal in mind, we now formally present our query-based reachability model.

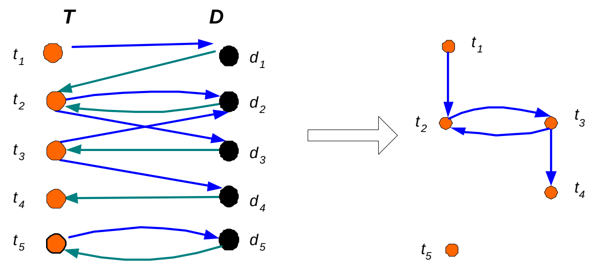

Figure 1: Portion of the querying and reachability graphs of a database.

2.2 Querying and Reachability Graphs

First, we define the "queries" more formally and then we define our graph-based representation of querying. Recall that for *Task 1* the queries consisted of the conjunctions of the extracted *tuple attributes*, while for *Task 2* the queries were the *words* extracted from the retrieved documents. Both types of queries can be modeled as instances of a general unit of information, a *token*, which is extracted from a document –in an application-specific way– and can be used for querying.

DEFINITION 1.: *We define a* token *as a unit of information that can be extracted from a document and can be converted into a Boolean query, perhaps involving phrases.* $\diamondsuit$

The actual choice of what is considered a *token* is application-specific. The tokens might be the *words*, or the *named entities* (e.g., "*Microsoft Corp.*") that appear in the documents, or even tuples (such as $\langle$*flu, aspirin*$\rangle$, which can be transformed to the query "*flu*" *AND* "*aspirin*") for an automatically extracted relation (such as *Treats(Disease, Drug)*). Using this definition of a token, we can now define the *querying graph* more formally.

DEFINITION 2.: *We define the* querying graph $QG(T, D, E)$ *of a database with respect to some task's querying strategy as a bipartite graph containing tokens* T *and documents* D *as nodes, and a set of edges* E *between* T *and* D *nodes. A directed edge from a document node* d *to a token node* t *means that* t *occurs in* d*. A directed edge from a token node* t *to document node* d *means that* d *is returned from the database as a result to a query that consists of the token* t*.* $\diamondsuit$

For example, suppose the token t_1=$\langle$*flu, aspirin*$\rangle$ retrieves a document d that also contains another token t_2=$\langle$*cold, tylenol*$\rangle$. Then, we insert an edge into QG from t_1 to d, and also an edge from d to t_2. We consider an edge $d \longrightarrow t$, originating from a document node d and pointing to a token node t, as a "*contains*" edge and an edge $t \longrightarrow d$, originating from a token node t and pointing to a document node d, as a "*retrieves*" edge. Notice that the existence of the edge $t \longrightarrow d$ in the graph does not imply the existence of the edge $d \longrightarrow t$, and the existence of the edge $d \longrightarrow t$ in the graph does not imply the existence of the edge $t \longrightarrow d$.

The *querying graph* representation can accommodate constraints that appear while querying real text databases. For example, a database might have an upper bound *MaxResults* on the number of documents returned as a result to a query. This is modeled by constraining any token vertex in the querying graph to have out-degree no larger than *MaxResults*. Another real-life constraint for a query-based algorithm might be an upper limit *MaxDocs* on the total number of documents retrieved. In this case, the algorithm to find all tokens that are reachable from an initial seed set (which can be considered as a walk on the graph) should not cross more than *MaxDocs* edges of the type $t \longrightarrow d$.

While the querying graph thoroughly describes the querying process, what we are really interested in is the *reachability graph* of the database, which is derived directly from the querying graph.

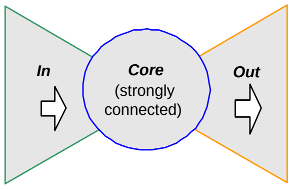

Figure 2: Structure of connected components in directed graphs.

DEFINITION 3.: *We define the* reachability graph $RG(T, E)$ *of a database with respect to some task's querying strategy as a graph whose nodes are the tokens T that occur in the database, and whose edge set E is such that a directed edge $t_i \rightarrow t_j$ means that t_j occurs in a document that is retrieved by t_i.* $\diamond$

In Figure 1(b) we show the reachability graph derived from the underlying querying graph, illustrating how its edges are added. Since token t_2 retrieves document d_3 that contains token t_3, the reachability graph contains the edge $t_2 \rightarrow t_3$. Intuitively, a *path* in the reachability graph from a token t_i to a token t_j means that there is a set of queries that start with t_i and lead to the retrieval of a document that contains the token t_j. In the example in Figure 1, there is a path from t_2 to t_4, through t_3. This means that query t_2 can help discover token t_3, which in turn helps discover token t_4. The absence of a path from a token t_i to a token t_j in the complete reachability graph means that we cannot discover t_j starting from t_i. This is the case, for example, for t_2 and t_5 in Figure 1.

The reachability graph is a directed graph, and its connected components can be described using the "bowtie" structure in [3]. Consider a strongly connected component *Core* in a reachability graph. By definition, every token in the *Core* is reachable via a directed path from every other token in the *Core*. Additionally, the tokens in the *Core* are reachable via a directed path from other tokens not in the *Core*, to which we refer as the *In* component (Figure 2). Finally, other nodes not in the *Core* or the *In* components are reachable via a directed path from the *Core* tokens. We refer to these nodes as the *Out* component (Figure 2). For example, consider the strongly connected component that consists of nodes t_2 and t_3 in Figure 1(b). The *In* component for this *Core* consists of the node t_1, while the *Out* component contains the token t_4.[2]

Having described the general shape of connected components in the reachability graph, we now turn to a quantitative analysis of the relative sizes of the different parts of the graph. We conjecture that the reachability graph of a database for tasks such as *Tasks 1* and *2* tends to belong to the well-studied family of power-law graphs. Power-law distributions have been known to arise in text domains [12]; additionally, power-law graphs have recently been observed to be a good model for graphs in related domains such as the web [3] and the Internet [7] graphs. One property of interest of power-law graphs is that the size of their connected components can be estimated using only a small number of parameters, as we describe next.

2.3 Reachability of Power-Law Graphs

A power-law graph [2] is a graph that has vertices with degrees that follow a *power-law* distribution. The power-law distribution states that the expected number of vertices y with degree k is:

$$y = e^{\alpha} \cdot k^{-\beta} \quad \Rightarrow \quad ln(y) = \alpha - \beta \cdot ln(k) \tag{1}$$

where the parameters α and β are the intercept and the slope of the line with best fit to the degree distribution plotted on the log-log scale.

Power-law graphs are actively studied in the graph theory community. Recent results [2, 5] allow to efficiently estimate properties of a given power-law graph. Specifically, Aiello, Chung, and Lu [2, 5] show that by using the average degree of the vertices in a random undirected power-law graph and the parameters α and β of the power-law distribution, it is possible to predict the size of the biggest connected component C_G of a graph, also called the "*giant component*". If the giant component emerges, then the remaining connected components are expected to be *small*, with a size distribution that also follows the power law.

We conjecture (and we study this experimentally in Section 3) that the reachability graphs in real text databases for our retrieval tasks can be modeled as *directed* power-law graphs. We use the giant *strongly connected* component *Core* to define the giant component C_{RG} in the reachability graph RG. More specifically, we define C_{RG} as the biggest strongly connected component *Core*, with its associated *In* and *Out* components in the "bowtie" structure described above. According to the power-law, if a giant strongly connected *Core* component exists, then the remaining strongly connected components (and their associated *In* and *Out* components) are expected to be small.

We can now define the *reachability* of a text database with respect to some task's querying strategy. As we discussed, in order for tasks such as *Tasks 1* and *2* to succeed, the extracted tokens must help discover other new tokens, which by definition are reachable from the previously discovered tokens. As we discussed above, if a large component C_{RG} exists in the reachability graph RG, then a token not in C_{RG} necessarily belongs in a *small* component and can help discover only a small number of new tokens. On the other hand, any token in the *In* or *Core* portions of C_{RG} will allow the querying strategy to discover all of the tokens in the *Core* and *Out* portions of C_{RG}. In other words, the relative size of the *Core* and the *Out* portions of C_{RG} can be used to predict the performance of query-based algorithms for tasks such as *Tasks 1* and *2* for executions where at least one initial seed token happens to be part of the *In* or *Core* portions of C_{RG}. Thus, we define the *reachability* of a text database as the fraction of the nodes T of the reachability graph that belong to the *Core* and *Out* portions of the giant component C_{RG}:

$$reachability = \frac{|Core(C_{RG})| + |Out(C_{RG})|}{|T|} \tag{2}$$

In the rest of the paper, we turn to the problem of how to efficiently approximate the reachability of a database for some task's querying strategy.

3. REACHABILITY OF REAL DATABASES

In this section we show that the reachability graphs constructed over real text databases for *Tasks 1* and *2* have an approximate power-law degree distribution. Hence, we conjecture that power-law graphs can model the structure of reachability graphs for these retrieval tasks. We illustrate our observations using two real text databases, one for each of our retrieval tasks.

NYT: This database is a collection of 135,000 newspaper articles from The New York Times, published in 1995. We use this database for an instance of *Task 1*: to retrieve all of the *tuples* describing disease outbreaks (e.g., ⟨*Typhus, Belize, June 1995*⟩), extracted from the NYT database using *Proteus* [11], a sophisticated information extraction system developed at New York University. A total of 8,859 tuples were extracted from the collection using an exhaustive scan of the database (which required over two weeks to complete). In this case, the *tokens* correspond to the *tuples* of the target relation, and the queries are constructed using the conjunction of the

[2]Additionally, other nodes not in the *In*, *Out*, or *Core* might still be connected to these components (e.g., the nodes in the "tendrils" of the component [3]). These additional nodes do not help in our reachability analysis, and hence we do not consider them further.

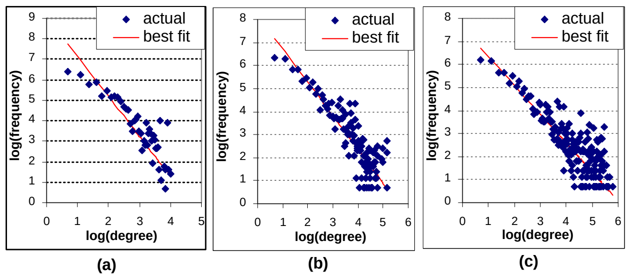

Figure 3: The outdegree distribution of the NYT reachability graph for *Task 1* when (a) *MaxResults* = *10*, (b) *MaxResults* = *50*, and (c) *MaxResults* = *200*.

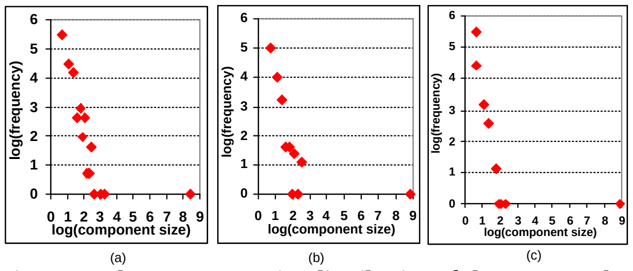

Figure 4: The component size distribution of the NYT reachability graph for *Task 1* when (a) *MaxResults* = *10*, (b) *MaxResults* = *50*, and (c) *MaxResults* = *200*.

first two attributes of each tuple (e.g., *"Typhus" AND "Belize"*). The complete reachability graph RG is computed by querying the database with all the 8,859 tuples extracted beforehand from the collection. To simulate constraints that search engines might impose, we limit the maximum number of documents, *MaxResults*, retrieved for each query.

We report the degree distribution of the resulting reachability graphs of the NYT database with respect to *Task 1* in Figures 3(a)-(c) when *MaxResults* is set to 10, 50, and 200 respectively. We show the power-law distribution that best fits the data. As we can see, the outdegree distribution is closely related to the fitted distribution. We report the size distribution of *connected components* in Figure 4, which also agrees with the size distribution expected for power-law graphs.

20NG: This database is the "20 newsgroups" collection of approximately 20,000 Usenet articles, from the UCI Machine Learning Repository. We now describe the construction of the reachability graph with respect to *Task 2* (building a comprehensive summary of word document frequencies in the database). In this task, the tokens are the words in the documents. Such summaries typically ignore stopwords, and therefore we do not include stopwords in our reachability graph. Figures 5 (a) and (b) report the outdegree distribution of the reachability graph constructed for *Task 2*, for *MaxResults* equal to 1 and 10, respectively. The outdegree distribution follows a power-law form for a large part of the distribution. We can model the distribution more accurately using extensions of the pure power-law model (e.g., see [9]), but a full discussion of other candidate distributions is beyond the scope of this paper. For *MaxResults*=1 and 10 we have just one connected component, which includes all the tokens of the reachability graph.

We conjecture that these observations will hold for the reachability graphs constructed over other text databases as well. We will explore this further experimentally in our future work. For the text databases where our conjecture holds, we will be able to predict the reachability of the databases (and consequently the performance of algorithms for *Tasks 1* and *2*) without constructing the complete reachability graph. Instead, we can simply estimate the parame-

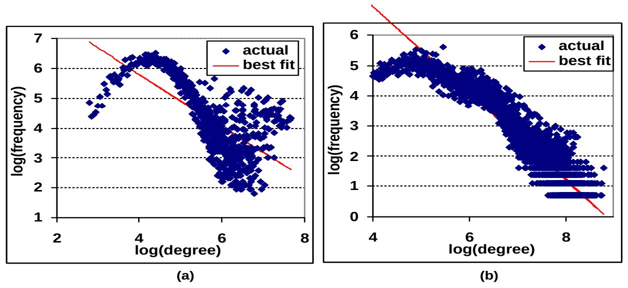

Figure 5: The outdegree distribution of the 20NG reachability graph for *Task 2* when (a) *MaxResults* = *1*, and (b) *MaxResults* = *10*.

ters of the outdegree distribution of the corresponding reachability graphs, as discussed next.

4. ESTIMATING GRAPH PROPERTIES

In this section we describe how to estimate the reachability of a database for a query strategy. Specifically, Section 4.1 shows that we can base our estimates on the average node outdegree d in the reachability graph, which in turn we can estimate using a small document sample from the database. Section 4.2 describes our document sampling method.

4.1 Estimating Reachability

As we discussed, in a power-law graph at most one giant connected component is expected to emerge, and the rest of the connected components are expected to be small. Thus, the reachability estimation for power-law random graphs reduces to estimating the size of the giant component, if any.

Chung and Lu [5] showed that for an undirected power-law graph G with values of $\beta < \beta_0$ $(\beta_0 \approx 3.475)^3$, a giant component emerges, while for smaller values of β all components are expected to be small. More specifically, [5] estimates the relative size of the giant component C_G when $\beta < \beta_0$ as follows:

$$\frac{|C_G|}{|T|} \geq \begin{cases} 1/d \cdot (1 - \frac{2}{\sqrt{de}}) & if \;\; d \geq e \\ 1/d \cdot (1 - \frac{1+\log d}{d}) & if \;\; 1 < d < e \\ 0 & if \;\; 0 < d \leq 1 \end{cases} \quad (3)$$

where d is the average degree of G. Note that the estimate *depends only on d*.

We believe that Equation 2 can be applied to estimate the size of the giant component C_{RG} in our *directed* reachability graph RG, and consequently to estimate the *reachability* of a database with respect to a given extraction task, where d in the equation above is the average *outdegree* of RG. Note that while *reachability* (Equation 2) is defined in terms of the *Core* and *Out* portions of C_{RG}, the equation above predicts the relative size of the *complete* giant component in the undirected graph. Therefore, the value predicted by Equation 3 will overestimate the *reachability*. Unfortunately, we are not aware of any similarly compact theoretical results to estimate the sizes of the specific portions of the giant component in directed graphs. As such results are developed, we could use them to improve the quality of our estimation.

Recall that by applying the results in [5] we can estimate the *reachability* of a database with respect to a given task by estimating the average outdegree d. We now describe an efficient document sampling technique that can be used to estimate d based on an observed (small) sample of the reachability graph.

[3] β is the absolute value of the slope of the linear fit to the degree distribution. See Section 2.3.

4.2 Sampling Text Databases for Estimating Reachability

In order to estimate d, we retrieve a small document sample D_S and construct a reachability graph RG_S for D_S. By definition, RG_S is a subgraph of the complete reachability graph RG. We construct RG_S as follows. We start with a small number (e.g., 50) of seed tokens $T_{seed} \subset T$. We then query the database for each token $t_i \in T_{seed}$, retrieving up to *MaxResults* documents for each query. For each document d retrieved by a token t_i, we extract each token t_j in d. For each t_j, we insert an edge $t_i \rightarrow t_j$ into RG_S.

Having obtained our sample subgraph RG_S, we can use it to estimate parameters of interest for the complete reachability graph RG. Specifically, we estimate d, the average outdegree of RG, as the average outdegree of the nodes in T_{seed}. Note that the outdegree of each node $t_i \in T_{seed}$ in RG_S is equal to the outdegree of t_i in RG. Since we draw T_{seed} randomly, we expect that the average outdegree of the corresponding vertices in the partial reachability graph will reflect the average outdegree of the complete RG.

The estimate of the average outdegree is used to predict the relative size $|C_{RG}|/|T|$ of the giant component as described in Equation 3, which approximates the reachability of the text database in question (Equation 2).

5. EXPERIMENTS

We now report experimental results for the technique that we described in Section 4. In Section 5.1 we describe the experimental setup, and in Section 5.2 we report the results of our preliminary experiments.

5.1 Experimental Setup

To evaluate the accuracy of our estimation technique, we constructed the reachability graph for the NYT and 20NG databases (Section 3). The reachability graph for NYT was constructed for *Task 1* and the reachability graph for 20NG was constructed for *Task 2*, for different values of *MaxResults*, one of the constraints that may be imposed by the text database. For each reachability graph, we compute the size $|C_{RG}|$ of its giant component using the STRONGLY-CONNECTED-COMPONENTS algorithm from [6] to compute the *Core*, and subsequently the *In* and *Out* components. Using $|Core(C_{RG})|$ and $|Out(C_{RG})|$ we compute the reachability values (Equation 2), which are reported in Figure 6, and serve as the "gold standard" for evaluating the estimation accuracy of our method.

Observe that almost all of the tokens for *Task 2* over 20NG are reachable for all values of *MaxResults* (i.e., a complete database summary can be constructed with exhaustive querying)[4]. In contrast, a large fraction of the tokens for *Task 1* over the NYT database are not reachable for values of *MaxResults* below 200 (i.e., by following the approach of *Task 1* we will not be able to reach a large fraction of the tuples in the target relation no matter how many queries we issue to the database).

For each database and each value of *MaxResults*, we apply the sampling technique of Section 4.2 by starting with a different number of seed tokens. Specifically, we sample the corresponding database using S randomly chosen *seed* tokens[5] from T to construct RG_S. We experimented with $S = 10, 50, 100,$ and 200, which means that we would send $10, 50, 100,$ and 200 token queries to the database to estimate its reachability. The maximum number of documents retrieved during sampling has a strict upper bound equal to $MaxResults \cdot S$.

[4]The C_{RG} for *MaxResults* $= 1$ consists of *Core*, which contains 95.5% of all tokens in C_{RG}, and *In*, which contains the rest.

[5]We assume that we can somehow obtain (e.g., as user input) an initial seed set of the appropriate size S. When this is not possible, we can repeatedly obtain random *document* samples until S seed tokens are extracted.

	NYT				*20NG*
MaxResults	*Core*	*In*	*Out*	*reachability*	*reachability*
1	0.001	0.003	0	0.001	0.955
10	0.078	0.156	0.063	0.141	1
50	0.260	0.159	0.200	0.460	1
100	0.334	0.132	0.274	0.608	1
200	0.388	0.102	0.334	0.722	1
1000	0.477	0.036	0.429	0.906	1

Figure 6: The relative size of the subcomponents of C_{RG} for the NYT and 20NG databases and for *Tasks 1* and *2* respectively, for different values of *MaxResults*.

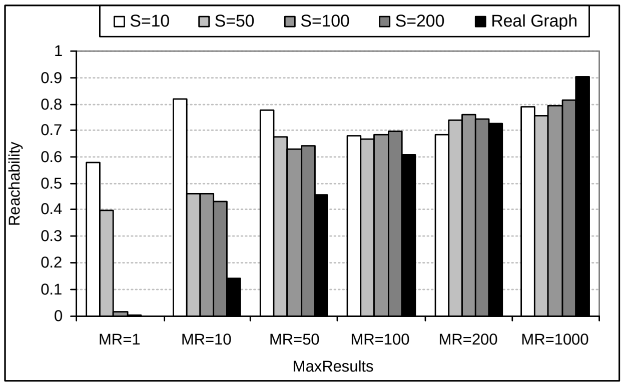

Figure 7: The reachability estimates for the NYT database for *Task 1*, for different values of *MaxResults* and seed sample size *S*.

5.2 Experimental Results

We report the estimation results for the NYT database for *Task 1* in Figure 7. As we can see, for *Task 1* our technique is able to estimate the reachability of the database with varying success. For example, when $MR = 1$ the actual reachability is 0.001. For sample size S of 100 and 200 queries, we estimated the reachability to be 0.015 and 0.002, respectively. While this estimate has high relative error, it is *good enough*, as it indicates that an algorithm for *Task 1* will not succeed in retrieving all necessary documents if *MaxResults = 1*. For the remaining values of *MaxResults* we still generally overestimate the reachability of the database, but our estimates are closer to the real value. As we discussed above, the overestimates are partly caused by using the prediction of Equation 3 to estimate the relative size of only the *Core* and *Out* portions of the giant component. Another possible cause of error is the divergence of real reachability graphs from the idealized, pure power-law model, and may be remedied by extending our work to include refinements of the power-law model. While not exact, our estimates are still indicative of the general structure of the reachability graph, and consequently can be used as a rough predictor of expected performance of algorithms for *Task 1*, as we will discuss in the next section.

Figure 8 reports the results for estimating reachability of the 20NG database with respect to *Task 2*. The results show that our estimation technique was able to accurately detect that the 20NG database is completely reachable for *Task 2*. For example, while the *Core* for $MR = 10$ includes the complete graph (i.e., the reachability is 1), we estimated the reachability to be 0.95 by submitting only 10 queries, which allows us to predict that general approaches for *Task 2* will be successful for this database.

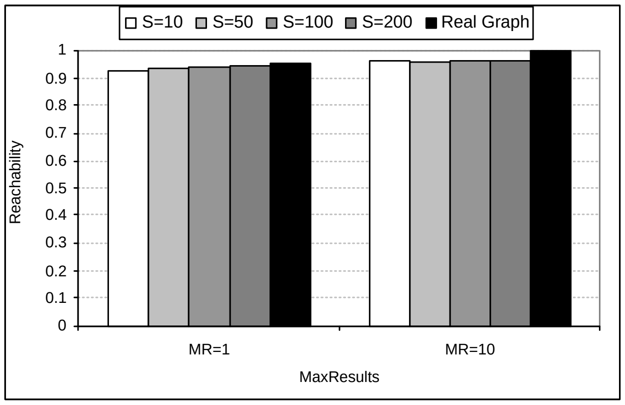

Figure 8: The reachability estimates for the 20NG database for *Task 2*, for different values of *MaxResults* and seed sample size *S*.

6. DISCUSSION

We presented a model for query-based access to text databases for general information extraction and database summary construction tasks. To the best of our knowledge, this is the first attempt to model query-based access to text databases, and our work parallels the efforts of modeling the web [3] and Internet [7] topologies. Based on this model, we developed the *reachability* metric, which indicates the expected performance of a general class of algorithms for these tasks. To complement our model, we presented an efficient technique for computing approximate values of reachability for a database with respect to the desired task.

We believe that our graph-based abstraction of the querying process can be used to model a variety of query-based access methods. Currently, most query-based algorithms are only empirically tested, with limited theoretical justification. We plan to model other query-based algorithms and use results from graph theory to provide theoretical justification for the observed experimental results.

In this work, our predictions for *Tasks 1* and *2* are corroborated by empirical studies presented in [1] and [4] respectively. Reference [1] reports using *Tuples*, an implementation of the strategy for *Task 1* for extracting the *DiseaseOutbreaks* relation over the NYT database with *MaxResults=50*. We found that all but a handful of the tuples retrieved by *Tuples* in [1] are in the *Core* $\cup$ *Out* portions of the giant component of the corresponding reachability graph, and the resulting recall of the strategy is therefore correctly predicted by the reachability value of 0.46 (Figure 6). If higher recall is desired, our reachability predictions could be used to either increase –if possible– the maximum number of documents, *MaxResults*, returned for each query (at the expense of precision), or choose an alternative querying strategy such as *QXtract* [1], which generates queries automatically by following a different approach. On the other hand, our model predicted that the algorithm proposed by Callan et al. [4] for *Task 2* can successfully discover all the words that appear in the text database, as long as no limitation on the number of queries issued is imposed.

Finally, other properties of the reachability graph are also of interest and can be estimated using a small number of parameters. The edge density of the reachability graph is an indication of the rate at which an algorithm can obtain new information by querying a text database. The diameter of the graph shows the minimum required effort to retrieve all the information stored in the database.

Additionally, the *querying* graph is also a useful tool for studying the efficiency of different algorithms. In this paper we have seen that the *reachability* graph can be used to predict whether a method can succeed in retrieving all the tokens stored in a database. However, the reachability graph cannot reveal how many queries are required to retrieve these tokens. In contrast, this information could be derived from the querying graph. For example, the reachability metric predicts that the algorithm in [4] for *Task 2* can retrieve all the tokens from the 20NG database. However, to achieve the goal of retrieving all the tokens, we have to issue thousands of queries and retrieve thousands of documents from the database, which is modeled by crossing thousands of edges in the corresponding querying graph. By issuing only a small number of queries and retrieving up to a total of 300-500 documents, as suggested in [4], it is possible to retrieve only (arguably the most "important") 15%-20% of the tokens in 20NG, which can be predicted by analyzing the querying graph for this task. In the future, we would like to further study the properties of the querying graph to determine how effective an algorithm can be if there is a limit on the number of queries or on the number of documents that can be retrieved from the database.

We believe that our model can serve as inspiration to develop even more comprehensive models and provides useful abstraction tools for the study of a variety of query-based algorithms.

ACKNOWLEDGEMENTS : This material is based upon work supported by the National Science Foundation under Grants No. IIS-9733880 and No. IIS-9817434. We thank Michael E. Agishtein for many fruitful discussions, and Regina Barzilay for her helpful comments.

7. REFERENCES

[1] E. Agichtein and L. Gravano. Querying text databases for efficient information extraction. In *Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE 2003)*, 2003.

[2] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *Proceedings of the ACM Symposium on Theory of Computing (STOC 2000)*, 2000.

[3] A. Z. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. L. Wiener. Graph structure in the web. In *Proceedings of the Ninth International World Wide Web Conference (WWW9)*, pages 309–320, 2000.

[4] J. Callan and M. Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.

[5] F. Chung and L. Lu. Connected components in random graphs with given degree sequences. *Annals of Combinatorics*, 2002.

[6] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill Science, Mar. 1990.

[7] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *SIGCOMM*, 1999.

[8] P. G. Ipeirotis and L. Gravano. Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB 2002)*, 2002.

[9] M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. In *First Workshop on Algorithms and Models for the Web-Graph*, 2001.

[10] M. A. Shah. ReferralWeb: A resource location system guided by personal relations. Master's thesis, M.I.T., May 1997.

[11] R. Yangarber and R. Grishman. NYU: Description of the Proteus/PET system as used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998.

[12] G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.