Popularity-Guided Top-k Extraction of Entity Attributes*

Matthew Solomon
Columbia University
solomon@cs.columbia.edu

Cong Yu
Yahoo! Research
congyu@yahoo-inc.com

Luis Gravano
Columbia University
gravano@cs.columbia.edu

ABSTRACT

Recent progress in information extraction technology has enabled a vast array of applications that rely on structured data that is embedded in natural-language text. In particular, the extraction of *concepts* from the Web—with their desired *attributes*—is important to provide applications with rich, structured access to information. In this paper, we focus on an important family of concepts, namely, *entities* (e.g., people or organizations) and their attributes, and study how to efficiently and effectively extract them from Web-accessible text documents. Unfortunately, information extraction over the Web is challenging for both quality and efficiency reasons. Regarding quality, many sources on the Web contain misleading or invalid information; furthermore, extraction systems often return incorrect data. Regarding efficiency, information extraction is a time-consuming process, often involving expensive text-processing steps. We present a top-k extraction processing approach that addresses both the quality and efficiency challenges: for each entity and attribute of interest, we return the top-k values of the attribute for the entity according to a scoring function for extracted attribute values. This scoring function weighs the extraction confidence from individual documents, as well as the "importance" of the documents where the information originates. We define the document importance in terms of entity-specific document "popularity" statistics from a major search engine. Overall, our top-k extraction processing approach manages to identify the top attribute values for the entities of interest efficiently, as we demonstrate with a large-scale experimental evaluation over real-life data.

1. INTRODUCTION

There is an unprecedented wealth of information on the Web, and search engines are the dominant interface to access much of this information. As the traditional Web search paradigm (i.e., keyword search with 10 blue links as results) becomes increasingly mature, identifying *concepts* from the Web and extracting desired *attributes* about them are critical to enable rich, structured access to information. For example, a user may be interested in compiling information about a group of politicians, such as their party affiliations and positions in government. Rather than directing the user to potentially relevant documents that contain the information, a better system should automatically present to the user the exact values corresponding to these attributes of interest. In this paper, we focus on an important subset of the concepts, namely, *entities* (e.g., people or organizations) and study how to efficiently and effectively extract attribute information for them from the Web.

Research in information extraction has designed techniques to automatically extract structured relational data from unstructured text. For example, an information extraction system would parse the sentence "Barack Obama is the president of the United States," and produce a tuple (Barack Obama, President). Extraction management systems can then be used to construct comprehensive lists of entities and corresponding attribute values from large document sets. Applying these techniques to the Web at large introduces many research challenges, some of which we will summarize and address in this paper.

One key challenge for extracting information from unstructured text sources on the Web is extraction quality. Sources on the Web include contradictory information, and extraction systems occasionally make extraction errors. It is, therefore, critical for the extraction management system to consider the "importance" of the sources of information when estimating the "value" of the potentially conflicting extraction results. In this paper, we do not attempt to determine what facts are "true," an impossible proposition in its general form; instead, we focus on resolving extraction conflicts by analyzing the source importance.

Another challenge is efficiency: information extraction is often an expensive process that uses complex text processing methods (e.g., part-of-speech tagging, named-entity recognition, segmentation) to identify the entities and attribute values within natural-language text. Also, a large number of domain- and application-specific extraction systems coexist on the Web. (One-size-fits-all systems such as [2] might not be appropriate for applications that require specialized extraction systems.) Furthermore, extraction systems are constantly being refined to improve accuracy, as well as to handle changes in the extraction tasks and in the underlying data sources. Therefore, a brute-force extraction approach

*This material is based upon work supported by a Yahoo! Faculty Research and Engagement Gift, and by the National Science Foundation under Grant IIS-08-11038.

WebDB '10 Indianapolis, IN USA

that processes all documents on the Web exhaustively for every system in the evolving set of available extraction systems would be prohibitively expensive. It is then critical to carefully choose which documents to consider for each extraction task, for scalability and efficiency.

To address the above quality and efficiency challenges, we adopt a *top-k extraction processing approach*: for each entity and attribute of interest, we aim to return the top-k values of the attribute for the entity according to a scoring function for extracted attribute values. This scoring function will depend on a number of key factors, including extraction confidence and the importance (e.g., popularity, as determined by a search engine's access logs) and number of sources where the extraction originated. In turn, the scoring function provides opportunities for algorithms to extract the top-k values of the entity attributes efficiently, by avoiding processing a large number of the available documents.

Several quality metrics for scoring extractions, some including extraction redundancy as a factor (e.g., [4, 12]), have been proposed. Our scoring function, as discussed above, will rely on a measure of "importance" of Web sources for the entities of interest. If most users are looking at information about certain entities in a small group of Web sites, then these sites are likely to contain important information about those entities. Interestingly, search engines have access to large repositories of user data that can be leveraged to identify important sources of information. For example, by analyzing query click-through statistics, search engines can identify the Web documents that people refer to for information about politicians in general or an individual politician specifically.

With access to such rich and dynamic data, search engines could consistently prioritize extraction on the important sites, and therefore reduce the amount of work necessary to extract the best attribute values, as we will see. In short, the contributions of this paper are as follows:

- We define a robust scoring function for entity attributes extracted from text documents (Section 2).
- We cast and address our problem as a top-k extraction processing task (Section 3).
- We evaluate our techniques experimentally using real-world entities in two separate domains, and leverage large-scale, real-world user data from a major search engine (Section 4).

Finally, we discuss related work and summarize our findings in Sections 5 and 6, respectively.

2. PROBLEM DEFINITION

We consider the problem of efficiently extracting attributes (e.g., position or date of birth) of specific entities (e.g., athletes or politicians) from a set of text documents. To score the extracted attribute values, we could directly adopt the extraction confidence produced by the extraction system of choice. However, we should also consider information on the extraction sources and their importance. We propose a robust scoring function that combines both the *extraction confidence* and the *cumulative document importance* to better capture the ranking of the extracted attribute values. In the rest of this section, we formally describe extraction confidence and document importance in Definitions 1 and 2, respectively, and introduce our comprehensive scoring function in Definition 3.

Definition 1 (Extraction confidence). *Given an attribute A and a text document d, an information extraction system extracts zero or more pairs (e, a) from d, where e is an entity and a is a value of A for e. Furthermore, the extraction system returns an* extraction confidence score *$conf(e, a, d)$ associated with the (e, a) pair and document d, indicating the expected quality of the extracted information ($0 \leq conf(e, a, d) \leq 1$).*

As an example, consider extracting attribute *position* for politicians. The OpenCalais extraction system[1] extracts the pair (Barack Obama, President) from document d = `http://news.yahoo.com/topics/barack-obama`, indicating that entity "Barack Obama" has value "President" for attribute position. Furthermore, OpenCalais associates this extraction with *conf(Barack Obama, President, d)* = 0.595, indicating a relatively high confidence that this information is correct. Another piece of extracted information (e.g., (Barack Obama, Candidate)) may, on the other hand, get a lower confidence score (e.g., 0.205).

Definition 2 (Document importance). *Given an entity e and a document d, we use $\gamma_{d,e}$ to represent the* entity-specific importance of d for e.

The document importance $\gamma_{d,e}$ can be instantiated in different ways. For example, we can consider the "authoritativeness" of d for e [9]. In this paper, we leverage search engine statistics by examining how often each document is accessed by users when searching for information related to an entity. The rationale is that the more frequently a document is requested by users, the more likely it contains important information. (See Section 4 for details.) Consider an example where we are processing two political entities, e_1 = "Barack Obama" and e_2 = "Sarah Palin." The Facebook page for Obama has $\gamma_{d,e_1} = 197.053$, a relatively high score. In contrast, $\gamma_{d,e_2} = 0$, which is not surprising, given that this document is not promising for the extraction of data about Palin.

We are now ready to define the "score" that we assign to each extracted attribute value for an entity.

Definition 3 (Attribute Value Score). *Consider a set D of documents, an entity e, and a value a for an attribute A of e. For a given information extraction system, the* score of attribute value *a for entity e over document set D, $score(e, a, D)$, is defined as $\sum_{d \in D} \gamma_{d,e} \cdot conf(e, a, d)$.*

Intuitively, the score of an attribute value a derived from a document set D for an entity e will be high if the information extraction system derives (e, a) with high confidence from a large number of important documents for e. This definition combines previous proposals in the literature to characterize extraction quality (e.g., [8, 12]).

Given the scoring function, we propose a top-k entity attribute extraction task, which is formally defined as follows. **Problem Statement**: Consider a set E of entities of interest, an entity attribute A, and an extraction system for A, as well as a set D of documents. We assume that we are given the entity-specific importance, $\gamma_{d,e}$, for each $d \in D$ and $e \in E$. We aim to retrieve the top-k values of attribute A for each $e \in E$, according to the *score* function from Definition 3, while minimizing the number of documents in D that are processed with the extraction system.

In the next section, we describe our preliminary approach for addressing this top-k extraction processing problem.

[1] `http://www.opencalais.com/`.

3. EXTRACTING TOP-K VALUES

Our goal is to extract the top-k values of a given attribute for a given set of entities, using an information extraction system over a set of documents D. The algorithms that we consider share the following general structure:

Top-k Extraction Processing Algorithm:

Input: set E of entities; information extraction system X for attribute A; set D of documents

Output: for each $e \in E$, the top-k values of A that can be extracted from D using X, according to our scoring function (see Definition 3)

Step 1: Document Selection. Select a batch of unprocessed documents from D.

Step 2: Extraction. Process each document in batch with extraction system X.

Step 3: Top-k Calculation. Update rank of extracted attribute values for each $e \in E$, using Definition 3.

Step 4: Stopping Condition. If top-k values for each $e \in E$ have been identified, stop; otherwise, go to Step 1.

To define the stopping condition (Step 4), consider a point in the execution where we have processed document set D_p, and let $D_U = D - D_P$ be the set of unprocessed documents. For each entity e, let $a_1, \ldots, a_k$ be the top-k attribute values for e extracted up to that point in the execution, with $score(e, a_i, D_P) \geq score(e, a_j, D_P)$ if $i < j$. (If at least one entity in E has fewer than k extracted values, then the extraction must continue[2].) Intuitively, the execution can stop as long as, for every $e \in E$, $score(e, a_i, D) \geq score(e, a_u, D)$, for $i = 1, \ldots, k$ and $\forall a_u \in V_e - \{a_1, \ldots, a_k\}$, where V_e is the set of all attribute values that can be extracted for e from the full document set D. In other words, we can stop as long as the current top-k values for each entity can be shown to be the overall top-k values for the entity.

Interestingly, from Definition 3 it follows that $score(e, a, D) \leq score(e, a, D_P) + b(D_U, e)$, where $b(D_U, e) = \sum_{d \in D_U} \gamma_{d,e}$. So $b(D_U, e)$ is a (conservative) upper bound on the maximum gain in score that any attribute value a can receive for e by processing all of the documents in D_U. (This bound will be reached if (e, a) is extracted from every document in D_U with perfect confidence.) We can then state the following proposition, which specifies when we have found the top-k attribute values for an entity.

PROPOSITION 1. *Consider a set D of documents, an entity e, and an information extraction system X for an attribute A of e. Let $a_1, \ldots, a_k$ be the top-k values for A and e that can be extracted from $D_P \subseteq D$ by X, such that $score(e, a_i, D_P) \geq score(e, a_j, D_P)$ if $i < j$. Furthermore, let $D_U = D - D_P$. Then $a_1, \ldots, a_k$ are the overall top-k values for A and e if $score(e, a_k, D_P) \geq s_u + b(D_U, e)$, where s_u is either (1) $score(e, a_{k+1}, D_P)$, where a_{k+1} is the top-$(k+1)$ attribute value extracted from D_P for e, if such value exists, or (2) 0, if only k values are extracted for e from D_P.*

This proposition indicates that the top-k values for e have been identified when no other values could possibly exceed the current top-k values. This stopping condition is analogous to that of traditional top-k query processing algorithms (e.g., TA [5], Upper [10]).

[2]Note that we can define an alternative formulation of the problem where some entities might receive fewer than k extracted values (e.g., if no such values can be extracted with a "sufficiently high" score); see Section 6.

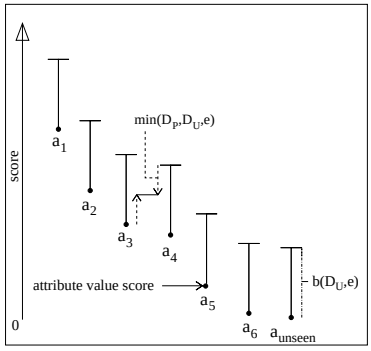


Figure 1: A snapshot of a top-3 processing scenario.

Figure 1 shows a snapshot of a top-3 execution for an entity. Six attribute values, namely, a_1 through a_6, have been extracted. The dark dots represent the current score of each value, while the segment above each dot represents the (uniform) $b(D_U, e)$ value. (a_{unseen} represents values not yet extracted, with a current score of zero.) The execution cannot stop at this point because a_4, for example, might become a top-3 value as we process more documents, since its score upper bound exceeds the current score of a_3.

We now turn to the critical document selection step (Step 1). Ideally this step should select a set of documents that is as small as possible among the sets that result in the extraction of the top-k values for every entity. We will consider a single-entity scenario first. After processing a document set D_P, we have a set of extractions (e, a_i) with corresponding scores $score(e, a_i, D_P)$. We want to select the next document $d \in D_U$ to process that will bring us closer to the stopping condition than any other document (see Proposition 1). The document with the largest $\gamma_{d,e}$ in D_U is naturally the best candidate: it will decrease $b(D_U, e)$ by the largest amount, and its extractions have the highest potential score of any $d \in D_U$ (see Definition 3). Then, in absence of additional information about the unprocessed documents, the optimal document selection strategy is to select the unprocessed document with the largest $\gamma_{d,e}$ until we can stop.

Interestingly, checking the stopping condition (Step 4) in this one-entity scenario does not need to be done after processing each document. Instead, at any point in the execution, we can calculate the minimum cumulative $\gamma_{d,e}$ value that the next batch of documents to process must have before the stopping condition can be reached. Specifically, if we have processed the documents in D_P and, correspondingly, we have not yet processed the documents in $D_U = D - D_P$, such minimum cumulative value is $min(D_P, D_U, e) = (s_u + b(D_U, e) - score(e, a_k, D_P))/2$, where s_u is defined as in Proposition 1 and a_k is the current k-th value of A extracted from D_P. (If fewer than k values have been extracted, then s_u and $score(e, a_k, D_P)$ are both assigned values of 0.) To understand why we need to process documents whose cumulative $\gamma_{d,e}$ value is at least $min(D_P, D_U, e)$ before stopping, consider once again the top-3 example in Figure 1. The figure shows that, in this case, $min(D_P, D_U, e) = (score(e, a_4, D_P) + b(D_U, e) - score(e, a_3, D_P))/2$, which is marked in the figure with a dashed line. To reach the stopping condition, there are two cases: (1) If a_3 is indeed among the top-3 values, then we need to prove it by reducing a_4's upper bound so that it does not exceed a_3's score. The current difference between these two scores is $2 \cdot min(D_P, D_U, e)$: in the most efficient scenario, after process-

ing documents with a cumulative $\gamma_{d,e}$ value of $min(D_P, D_U, e)$, a_3's score and a_4's upper bound coincide in "the middle point." (2) If a_3 is not among the top-3 values, then we need to lower its current score upper bound (i.e., $score(e, a_3, D_P) + b(D_U, e)$) so that it does not exceed the score of one value outside the current top-3 values. So we must lower a_3's score by at least $(score(e, a_3, D_P) + b(D_U, e) - score(e, a_4, D_P))/2 \geq min(D_P, D_U, e)$.

Overall, note that processing a batch of documents with a cumulative $\gamma_{d,e}$ of $min(D_P, D_U, e)$ does not guarantee that the stopping condition will be met; however, without processing such a batch of documents, the execution cannot complete. After processing a batch of documents, the $min(D_P, D_U, e)$ value is recalculated, and the next batch of unprocessed documents can be selected accordingly.

Now we can address the general, multiple-entity scenario. Intuitively, the algorithm should identify documents that have high importance values (Definition 2) for as many entities as possible: we process each of these documents just once with the information extraction system, and the result contributes to multiple entities. Unfortunately, we can show that the problem of picking the smallest set of documents D_S such that $\sum_{d \in D_S} \gamma_{d,e} \geq min(D_P, D_U, e)\ \forall e \in E$ is NP-Complete, by reducing the minimum set cover problem to our scenario [6]. So our algorithms will necessarily produce approximations to this optimal document set.

We now introduce four techniques that differ in how they select the batch of documents in Step 1 of the algorithm. The descriptions below assume that we have processed document set D_P, and $D_U = D - D_P$ is, as usual, the set of unprocessed documents.

Ind-Highest-n: Continue to add the document d with the highest $\gamma_{d,e}$ value for each individual entity e, until the batch contains n documents for every uncompleted entity (i.e., for each entity for which we have not yet identified the top-k values).

All-Highest-n: Continue to add the document d with the highest value of $\sum_{e \in E_U} \gamma_{d,e}$, where E_U is the set of uncompleted entities, until the overall batch has n documents.

Ind-Gamma: For each entity e, continue to add the document d with the highest $\gamma_{d,e}$ score until the batch of documents for the entity reaches $min(D_P, D_U, e)$.

All-Gamma: Select the document batch as follows: (1) batch $= \phi$; (2) $\forall e \in E, remaining_e = min(D_P, D_U, e) - \sum_{d \in batch} \gamma_{d,e}$; (3) If $\forall e \in E, remaining_e = 0$, then stop; (4) Choose d such that $\sum_{e \in E} min\{\gamma_{d,e}, remaining_e\}$ is maximum and add it to batch; (5) Go to Step 2.

The above techniques vary in their choice of documents and, importantly, in the efficiency of their document selection strategy. Specifically, Ind-Highest-n and Ind-Gamma are the most efficient strategies because they focus on each entity in isolation. In contrast, All-Highest-n and All-Gamma are less efficient since the document set must be reordered multiple times as entities complete and, for All-Gamma, as the $min(D_P, D_U, e)$ and $remaining_e$ values change.

4. EXPERIMENTAL RESULTS

We evaluate the performance of our algorithms using two real-life entity sets and a corpus of real Web documents. In particular, the document importance scores are derived from the Yahoo! search logs based on the behavior of Yahoo! users (see below). To foster future research, we have made the document importance data available as part of the Yahoo! Labs Webscope program[3].

4.1 Experimental Settings

Entity sets: We use two entity sets: *politicians* and *athletes*. The politicians dataset contains 547 politicians, which include senators, representatives, state governors, and a few prominent political figures such as Sarah Palin. The athletes dataset contains 5128 athletes, extracted from both Wikipedia and Yahoo! Sports. For our experiments, we focus on one attribute, *position*, as it is both multi-valued (i.e., each entity typically has multiple positions) and widely applicable to the entities that we consider. For politicians, this attribute corresponds to the various posts that the entities held over their career (e.g., author, lawyer, representative); for athletes, in contrast, this attribute corresponds to the various positions that each entity had on his or her team (e.g., quarterback, attacker, coach).

Document importance: We estimate the importance of a document (Definition 2) using the collective behavior of search engine users. We consider documents at two levels: *host* and *pattern*. Host is simply the host portion of the URL. Pattern, on the other hand, can take several forms: (1) an actual URL (e.g., `whitehouse.com/contact/`); (2) a simple regular expression for dates (e.g., `www.huffingtonpost.com/####/##/##/`, matching URLs with a prefix such as `www.huffingtonpost.com/2008/11/04/`); and (3) an expression with placeholders for an entity name or a long sequence of terms (e.g., `news.aol.com/main/{last_name}-pres-idency/article/{description}/`, matching URLs with a prefix such as `news.aol.com/main/obama-presidency/article/barack-obama-health-care/`).

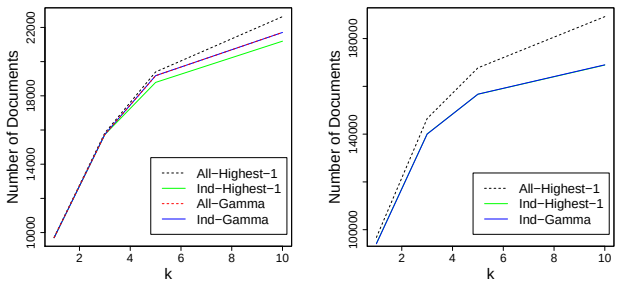
For each entity in the dataset, we compute a frequency measure based on how many users have searched for the entity and how many times pages matching a particular host or pattern have been clicked as a result of the search. We collect the normalized click rate for each (entity, host) and (entity, pattern) pair over a three-month Yahoo! search log. We also collect the click rate for each host and pattern over each entity in the dataset to estimate their importance to the entire domain (e.g., politicians or athletes). As mentioned earlier, this normalized aggregated frequency data is made available to the general research community through the Yahoo! Labs Webscope program.

To instantiate specific Web pages for each entity and host, we use the entity name to form a host-restricted query and issue it to the Yahoo! search engine through its BOSS API[4]. The top-10 returned documents for each (entity, host) pair are combined to form the full corpus of documents for that dataset. The corpus for politicians contains 28,546 documents, and the one for athletes contains 219,873 documents.

Given an entity e and a document d in the corpus, we estimate the *entity-specific document importance* $\delta_{d,e}$ as follows. If d matches a pattern, then $\delta_{d,e}$ is defined as the click rate for the (entity, pattern) pair divided by the number of documents that match the (entity, pattern) pair in the full corpus. Otherwise, if d does not match any pattern, then it is labeled as "miscellaneous." Miscellaneous documents are grouped together and receive any portion of the host click rate that is not covered by specific patterns. For example, for entity e,

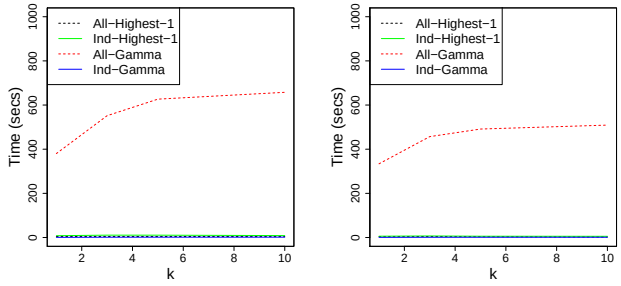
[3] `http://webscope.sandbox.yahoo.com/` (data set: ydata-ysearch-location-entity-sources-v1_0; contact: Cong Yu).

[4] `http://developer.yahoo.com/search/boss/`.



(a) politicians (b) athletes

Figure 2: Number of documents processed as a function of k for (a) politicians and (b) athletes.



(a) politicians (b) athletes

Figure 3: Scheduling overhead as a function of k for (a) politicians and (b) 11% athletes sample.

if the click rate for `en.wikipedia.org` is 100, and all patterns with that host (e.g., `en.wikipedia.org/{first_name}_{last_name}`) collectively have a click rate of 75, then the remaining, miscellaneous pages collected for e will share the remaining 25.

We also compute the importance β_d of each document d for the general entity *domain* (e.g., politicians or athletes) in a similar fashion, by considering a host or pattern click rate over the entire entity set. The entity- and domain-specific values for an entity e and document d are combined into the overall importance $\gamma_{d,e}$ (Definition 2) as follows: $\gamma_{d,e} = \alpha_e \cdot (\beta_d + \delta_{d,e}) \cdot r(d)$. In this formula, α_e[5] represents the entity importance (e.g., clicks for entity e documents divided by total clicks on any entity document) and $r(d)$ is the result of partitioning the total click rate for a host or pattern to its corresponding pages based on the order of the results returned by the search engine.

Extraction system and extraction confidence: Once a document is chosen for extraction, the actual extraction is conducted using OpenCalais, which returns extraction confidence values as well.

Metrics: We evaluate the efficiency of our techniques based on the *number of documents extracted* to obtain the top-k attribute values for all entities. (The actual extraction from the chosen documents dominates the cost of the top-k extraction process.) We also measure the *scheduling time* for our techniques, to evaluate the cost of the document selection step (Step 1 of the algorithm in Section 3). The four alternative techniques that we consider differ on this document selection step. An ideal scheduling algorithm is expected to reduce the number of documents extracted while incurring little scheduling overhead. We implemented our algorithms in Python, on a Linux machine with 16 GB of RAM and two quad-core Intel Xeon 2 GHz processors.

4.2 Experimental Results

Number of documents: Figure 2(a) shows the number of documents processed for various k values for politicians. (For All-Highest-n and Ind-Highest-n we only report results for $n = 1$, as larger values of n consistently perform worse due to unnecessary document processing in the final batch.) The savings from prioritization are significant: for example, at $k = 1$, we only need to process 34% of the documents. As expected, a larger k value means more documents must be processed to ensure that the top-k attribute values for each entity are obtained. We notice that the increase is steeper at lower ranges. At $k = 1$, the four methods perform almost equivalently because for most entities one value dominates all other candidates (e.g., senator). As k increases, the difference between the frequency and quality of the top-k and top-$(k + 1)$ extractions is smaller, and lower-ranked extractions appear less frequently in general. Therefore increasingly more documents must be processed to identify the top-$(k + 1)$ value. Figure 2(b) shows analogous results for athletes but without All-Gamma, which is prohibitively expensive over large datasets such as athletes (see below).

Overall, the most complex document selection technique, namely, All-Gamma, which considers the "global" contribution of each document across entities, is matched in performance by the other, simpler techniques. We believe this is mainly due to extraction sparsity, i.e., the extraction system can only extract high-quality results from a small percentage of documents. As a result, the fastest way to reach the stopping condition is to shrink the bounds (from the top) rather than improve the scores of already extracted attribute values (from the bottom). Picking the documents with the largest $\gamma_{d,e}$ scores for each entity individually, therefore, becomes the most effective way to decrease the bound. Furthermore, in our datasets, documents with non-zero $\gamma_{d,e}$ scores for multiple entities are uncommon (1% in the politicians corpus, 5% in the athletes corpus), and the most important documents for each entity are often specific to that entity (e.g., Wikipedia articles, Twitter pages), hence the good performance of the techniques that consider individual entities in isolation for document selection. So we can rely on the less expensive techniques (namely, Ind-Gamma and Ind-Highest-1) and finish the top-k extraction process with a similar number of processed documents.

Document selection overhead: Figure 3 shows the document selection overhead of the four techniques for both politicians and an 11%-sample of the athletes dataset[6]. The scheduling task determines which document(s) to process for each batch, with the number of overall batches varying by method. Ind-Highest-1 produces a new batch each time a document is processed for each uncompleted entity, which results in significantly more batches than with Ind-Gamma or All-Gamma (e.g., for $k = 1$ on the politicians dataset, Ind-Highest-1 needs 616 document batches, while

[5]The current definition of $\gamma_{d,e}$ considers the "popularity" of entity e in factor α_e; in future work, we will investigate the impact of this factor in the top-k extraction process.

[6]The execution of All-Gamma was prohibitively expensive over the full athletes dataset, because of its size. An 11% sample of athletes matches the size of the politicians dataset.

Ind-Gamma needs just 16). For every batch calculation, all algorithms must at least check the stopping condition for each entity, to determine if execution has completed for any or all entities. All-Gamma and Ind-Gamma additionally must calculate new values of $min(D_P, D_U, e)$ for each entity to determine the next batch size. We measure the total cost by the cumulative CPU time used to produce all of the batches. As shown in Figure 3, the overall scheduling overhead of All-Highest-1, Ind-Highest-1, and Ind-Gamma is small, at around 10 seconds; this value is negligible relative to the actual extraction times: the time to process a single document using OpenCalais is, on average, 5 seconds, and ranges between 0.5 and 12 seconds depending on the document length. We also explored the overhead for increasingly larger fractions of the athletes dataset. Again, all algorithms except All-Gamma scale up nicely with no significant increase in scheduling overhead.

5. RELATED WORK

Our work is perhaps closest to a recent paper by Jain and Srivastava [8]. The authors investigate efficient algorithms for answering $good(k, \ell)$ queries, which return k extracted tuples among the top-ℓ fraction of all tuples ranked by their extraction confidence. Jain and Srivastava resort to this relaxation of the top-k extraction problem because, in the absence of a document prioritization scheme, solving the top-k version of the problem would require processing most documents. In our work, we exploit real user data to define the importance of each document for each entity, which leads to efficient solutions for the top-k extraction task.

Scalable information extraction from a large corpus of documents has recently received significant attention [3]. One important approach is OpenIE [2], which focuses on super-efficient extraction from all documents. Another approach is prioritization [7], which processes the most promising documents first in order to conserve resources. Our work adopts the general prioritization approach but differs from [7] by incorporating document importance into the prioritization decisions, without examining the documents themselves.

The stopping condition adopted in our scheduling algorithms is inspired by the bounding conditions for top-k query processing by Fagin et al. [5] and used in numerous other projects (e.g., [10]). Also, we assume that the extraction confidence is provided by the underlying information extraction system, which is a problem tackled in several previous studies [1, 4, 12]. Similar to our work, information from search logs has also been leveraged to make prioritization decisions for different tasks, such as Web crawling [11].

6. CONCLUSION AND DISCUSSION

In this paper, we addressed the problem of extracting the top-k attribute values for a set of entities of interest. Our approach relies on information extraction technology to process natural-language text documents and extract entity information. We proposed a scoring function for the extracted entity attributes that considers both the extraction confidence from each individual document, as well as the "importance" and number of the documents where the information originates. For our experiments, over two entity domains and real-life data, we relied on entity-specific document "popularity" statistics from a major search engine to define document importance. (The data that we used for our experiments is available as part of Yahoo!'s Webscope program.) Our experimental evaluation shows that our top-k extraction processing approach is promising and manages to produce the desired extraction results efficiently.

Many open issues and challenges remain, which will be the subject of future work. Specifically, our work so far has used a scoring function for extracted information that, as discussed, is inspired in multiple proposals in the literature to characterize extraction quality (e.g., [4, 8, 12]). The scoring function is intuitively appealing, including its choice of the specific document "importance" measure. But a careful evaluation of the merits of this function, as well as of variants thereof, remains the subject of important future work. As another direction of future work, we will investigate variations of the top-k model in this paper. Our current model insists that we extract k attribute values for each entity of interest. But often using a uniform value of k across entities is far from ideal (e.g., for the position attribute of politicians, consider a young politician who has held only one public position, and for whom $k = 1$ might be appropriate, against a much older politician who has been in the public eye for decades, and for whom a much larger value of k might be preferable). We will consider variations of the query model that perhaps combine the current top-k model with some threshold-based scheme, so that we can decide to declare an entity "completed" even if fewer than k attribute values for it have been identified and, alternatively, return more than k values for other entities, if appropriate. Finally, our current model assumes a predetermined set of entities of interest. In many domains, the entities of interest are slow-changing (e.g., football players, diseases), so a list of popular entities can be easily maintained. However, occasionally, a new or previously inconspicuous entity might become prominent, with a resulting surge in queries and relevant documents. As future work, we will study the impact on our general extraction approach of such emerging entities, which have a short-lived but high-volume life cycle of importance.

7. REFERENCES

[1] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *DL*, 2000.

[2] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[3] W. Cohen and A. McCallum. Information extraction from the world wide web. In *KDD*, 2003.

[4] D. Downey, O. Etzioni, and S. Soderland. A probabilistic model of redundancy in information extraction. In *IJCAI*, 2005.

[5] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, 2001.

[6] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[7] J. Huang and C. Yu. Prioritization of domain-specific web information extraction. In *AAAI*, 2010.

[8] A. Jain and D. Srivastava. Exploring a few good tuples from text databases. In *ICDE*, 2009.

[9] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[10] A. Marian, N. Bruno, and L. Gravano. Evaluating top-k queries over web-accessible databases. *ACM Trans. Database Syst.*, 29(2):319–362, 2004.

[11] S. Pandey and C. Olston. Crawl ordering by search impact. In *WSDM*, 2008.

[12] M. Wu and A. Marian. Corroborating answers from multiple web sources. In *WebDB*, 2007.