Quality Impact of Value Matching and Scoring in Top-k Entity Attribute Extraction*

Matthew Solomon
Columbia University
solomon@cs.columbia.edu

Luis Gravano
Columbia University
gravano@cs.columbia.edu

Cong Yu
Google Research
congyu@google.com

ABSTRACT

The *entity attribute extraction* problem, or how to extract entities and their attribute values from natural language Web documents, is of critical importance for Web search and information access in general. Unfortunately, because of the noisy nature of the Web and its scale, entity attribute extraction is notoriously challenging in terms of both extraction efficiency and quality. In our earlier work [24], we proposed a top-k extraction processing approach that addressed the efficiency challenge: Our approach leveraged a popularity-based scoring function to rank Web pages according to their entity-specific importance, and focused the extraction effort over the highly ranked pages for each entity of interest. The extraction quality resulting from this efficiency-motivated extraction approach, however, has not been studied and is the focus of this paper. Specifically, we make progress toward addressing the quality challenge through an in-depth analysis of two critical components of the extraction process, namely, *matching* and *scoring* of extracted attribute values. The design choices for these components can substantially impact the quality of the entity attribute extraction process, as we demonstrate with experiments with a state-of-the-art extraction system and entities from two domains of interest.

1. INTRODUCTION

Many sophisticated user search tasks involve searching and reasoning about real-world entities (e.g., people, organizations) and their attributes and properties. For these tasks, the traditional Web search paradigm is sometimes insufficient. As an example, consider a user who is interested in learning about all local politicians, together with attributes such as their party affiliations and positions. Ideally, with help from information extraction techniques, we should provide the user with the desired attributes directly, so that the user is not forced to gather the information manually from a long list of documents, as would be required with traditional Web search. The focus of this paper is on the *entity attribute extraction* problem, or how to extract entities and attribute values from natural language Web documents.

The entity attribute extraction problem requires processing individual documents with an information extraction system (e.g., Snowball [1], MALLET [21], SystemT [5]) and extracting *attributes* (e.g., position or date of birth) of specific *entities* (e.g., athletes or politicians). Information extraction is an expensive process, so processing all available documents should be avoided for efficiency, because (1) domain- and application-specific extraction tasks arise continuously, and (2) existing extraction programs are repeatedly refined and improved over time. Together, these two observations make the exhaustive, one-size-fits-all processing of all available Web documents (e.g., as in [4]) undesirable for our entity attribute extraction problem.

In recent work [24], we posed the entity attribute extraction problem as a *top-k extraction processing task*, where the goal is to return the k attribute values with the highest "score" for each entity of interest, as extracted from the available documents. Because information extraction is an error-prone task, a *scoring function* [7, 8, 11, 26] is typically used to assign a score to each extracted attribute value for an entity, indicating our confidence in the extracted value. With a scoring function, we can return the extracted attribute values that are most likely to be correct, while ignoring the rest. We showed that, if the scoring function assigns a high weight to information from important documents, identifying the top-k attribute values for a given entity will not require processing every available document. Our earlier work hence focused on the *efficiency* of the top-k extraction process, and is summarized in Section 2.

Unfortunately, this earlier work did not analyze the *quality of the output of the top-k extraction process*, which is the subject of this paper. Specifically, we will study the impact on extraction quality of two important factors in the top-k extraction process. First, in Section 3 we discuss how to match extracted information to appropriate entity names and attribute values, an instantiation of the *entity resolution* or *record linkage* problem to our extraction scenario. Second, in Section 4 we discuss the efficiency-motivated, entity-specific importance weighing of the documents for the scoring of the extracted information. As a key contribution of this paper, in Section 5 we report a revealing experimental analysis of these two important factors, involving entities from two different domains of interest, real-world Web documents and statistics derived from a large-scale commercial

*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.

search engine query log. As we will see, the design choices for the above two factors can substantially impact the quality of the entity attribute extraction process.

2. BACKGROUND

In [24], we posed the entity attribute extraction problem as a *top-k extraction processing task*, where the goal is to return the k attribute values with the highest score for each entity of interest, as extracted from the available documents. We showed that, if we use a scoring function that assigns a high weight to information from important documents, identifying the top-k attribute values for a given entity and attribute will not require processing every document in the input set. Intuitively, at some point before all documents are processed, we can safely stop processing documents, as further processing could not change the identity of the top-k values. Specifically, (a slightly modified version of) the top-k extraction processing approach that we used in [24], and which we continue to explore in this paper, is as follows:

Input: set of entities E; information extraction system X for attribute A; document set D; scoring function S.

Output: for each $e \in E$, the top-k values of A that can be extracted from D using X, according to S.

4-Step Process:

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

Step 2: Extraction and Matching. Process each document in the batch with extraction system X and match extractions to appropriate entities and attribute values.

Step 3: Scoring and Top-k Calculation. Update the score of each extracted attribute value.

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

Our earlier work in [24] used a flexible definition of scoring function that can be instantiated to model alternative options from the literature:

DEFINITION 1 (SCORING FUNCTION). *Consider a document set D, an entity e, a value a for an attribute A of e, and an information extraction system X. A scoring function S assigns a* score to value *a for entity e over document set D as: $score(S, e, a, D) = \sum_{d \in D} \gamma_{d,e} \cdot conf_X(e, a, d)$, where $\gamma_{d,e}$ represents the "importance" of d for e, and $conf_X(e, a, d)$ represents the confidence of (e, a) as extracted from document d by extraction system X.*

We showed that, by instantiating $\gamma_{d,e}$ with the "popularity" of document d for entity e, as determined from a large-scale search engine log [24], we can substantially improve the efficiency of the top-k extraction process, in comparison with alternative definitions that do not prioritize certain documents over others for extraction.

Our earlier work focused on *efficiency* considerations and did not analyze the impact on the *quality of the output of the top-k extraction process* of the choice of (1) matching strategy for Step 2 and (2) scoring function instantiations for Step 3. In particular, the matching component of Step 2 was defined in a rudimentary manner, by just matching entities and attribute values whenever the corresponding strings were identical after removing capitalization. We now describe robust, principled matching strategies and analyze their impact—and that of instantiations of the above scoring function—on the quality of the top-k extraction process.

3. MATCHING EXTRACTED VALUES

In Step 2, we process each document from Step 1 using an extraction system to extract entity and attribute value pairs. Given an extracted pair (e, a), we need to integrate it into $\{(e_1, A_1), (e_2, A_2), \ldots, (e_n, A_n)\}$, where each e_i is an entity of interest and each $A_i = \{a_i^1, a_i^2, \ldots\}$ is the (possibly empty) set of already extracted values for e_i. For this, we need to determine (1) if any entity e_i corresponds to e (Section 3.1) and (2) if so, if an already extracted a_i^j value matches a or if a is a new, distinct value (Section 3.2).

3.1 Entity Matching

The quality of the top-k results is strongly affected by how we match extracted entity values. (In the next section, we discuss the complementary problem of matching extracted attribute values.) Once an attribute value a for an entity e has been extracted from a document, identifying the entity—if any—that e matches, among the set of entities of interest in E, is a challenging problem. If we only accept entity names that match the name of an entity in E exactly, then we may miss a large number of valuable extractions (e.g., the same soccer player might be referred to as "Steven Gerrard," "Stevie Gerrard," or even "Stevie G.") [9, 12].

Fortunately, we can exploit the extensive literature on *entity resolution* (also known as record linkage or synonym resolution; see [9] for a survey). For example, we can use approximate string matching to verify if, say, the edit distance between two names surpasses some matching threshold [9]. Unfortunately, it is costly to calculate edit distance for large sets of entities and extractions, and this approach misses nicknames that differ substantially from the input name (e.g., "Stevie G." vs. "Steven Gerrard"). An alternative approach leverages large databases of entities and multiple attributes to "learn" synonymous entity names based on co-reference (e.g., both "Iker Casillas" and "Casillas" have the "goalkeeper" attribute) [27, 6]. However, this general method requires a large number of extractions for each entity and attribute of interest to obtain substantial co-reference statistics. In our entity attribute extraction scenario, this requirement is problematic for all but the most popular entities of interest; not-so-popular entities generally do not have a sufficiently large mass of associated extractions to enable such statistical approaches effectively.

An alternative approach, which we use in our experiments, is to leverage existing Web resources (e.g., `DBpedia.org`) and compile a list of acceptable candidate names for each entity. For each extraction (e, a), we compare the text string for e, normalized for case and punctuation, against all of the candidate name variations for each entity, and produce a match if the e string exactly (or, alternatively, approximately) matches a candidate name for any of the entities of interest. This technique is efficient, only requiring some pre-processing (namely, compiling the list of variations for the entities of interest, in an offline step) and a small number of exact (or, alternatively, approximate) string matching comparisons for each extracted entity name, during the extraction processing. Furthermore, this technique greatly improves the robustness of the extraction process.

3.2 Value Matching through Clustering

Once an extraction (e, a) has been matched to a specific entity e_i in E, we check if the attribute value a, or an equivalent variation thereof, already appears among the extracted

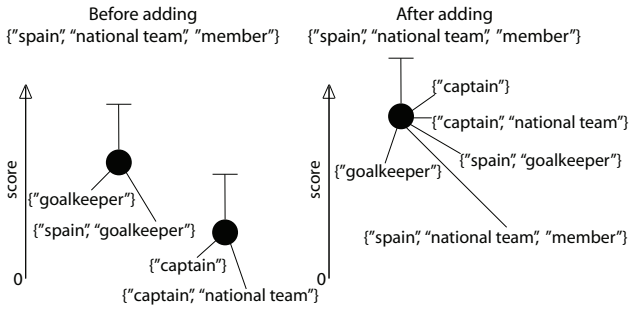


Figure 1: Problematic "merging" of two clusters for entity "Iker Casillas" and attribute *position*.

values A_i. If we do not resolve synonymous attribute values, we will produce redundant top-k entries (e.g., a "soccer player" may equivalently be referred to as a "professional soccer player" or a "footballer"). Additionally, we have observed that splitting an attribute value's "true" score among different versions has a significant impact on the efficiency of the processing. By matching these corresponding attribute values we avoid returning duplicate attribute values, resulting in diverse and useful results as well as in more reliable value scoring. Proposed approaches for matching attribute values are similar to those for entity resolution (see above), using string matching and learning approaches. For domain- and application-specific extraction systems, which may target specific or unique information, there will generally not be a large number of mentions of any particular value; in particular, our entities often have relatively rare attribute values (e.g., *position* instance "captain of the West Indies cricket team") that are not suitable for statistical approaches that rely on millions of extracted values to function. An alternative is to use existing semantic databases (e.g., WordNet [3]) to identify equivalent attribute values. However, these databases are unable to cover the lexical variability in domain-specific, noisy extracted attribute values (e.g., "good goalie," "first-rate goalkeeper").

In our work, we synthesize an alternate two-step method to *normalize* and *cluster* attribute values, by leveraging relevant research results in novel ways. Specifically, the first step identifies the most critical components, or "concepts," in each attribute value. Many noisy attribute values contain incorrectly extracted terms due to segmentation errors (e.g., "good goalkeeper," "hot goalkeeper"), and we must distinguish these noisy cases from correct extractions (e.g., "Spain goalkeeper"). To efficiently identify the most critical concepts in each attribute value, we develop a *filter* to eliminate noisy terms. This filter takes an attribute value string as input, and outputs a *concept set*, where each concept is an n-gram (for $n \leq 4$) that matches a concept from a domain-specific list compiled offline in a pre-processing step (e.g., see [23]), excluding subsets of other n-gram concepts. We then use this concept set as a representation of the original input attribute value (e.g., "last few years Real Madrid goalkeeper" becomes {"real madrid", "goalkeeper"}).

The second step makes the decision to either cluster an extracted attribute with an existing cluster of one or more concept sets extracted for the entity (i.e., the new value corresponds to already extracted values), or start a new cluster (i.e., the new value introduces a new, unseen attribute value for the entity in question). We make the clustering decision based on a similarity function for concept sets, to compare an extracted attribute value concept set and a concept set cluster. The similarity function can be defined in different ways; we choose to explore a set of functions that measure the overlap in concepts between two sets, such as the familiar Jaccard coefficient [13], as well as a binary function that measures whether a concept set is a subset of, a superset of, or equivalent to another concept set. Using set overlap guarantees that the concept sets share similar information; other functions (e.g., based on term similarity) result in a large number of "false positive" matches.

Beyond the choice of similarity function, another key decision is the choice of clustering algorithm, to group together extracted values that are, conceptually, just variations of each other. This choice is restricted to clustering algorithms that are *compatible with the top-k extraction processing approach*. At its core, the top-k approach processes documents incrementally—in an online manner—so that the processing can stop early, for efficiency, once the "current" top-k clusters for each entity can be shown to correspond to its actual top-k attribute values [24]. Intuitively, we need to focus on "stable" clustering algorithms that do not arbitrarily merge or substantially alter existing clusters. Otherwise, attribute values (i.e., clusters) with arbitrarily low scores might be promoted into the top-k values, through a series of merges or substantial alterations, preventing the early stopping of the top-k extraction processing. See Figure 1 for an example of a step in the execution of such "unstable" clustering (a single-link hierarchical agglomerative algorithm [19]), where two position values (i.e., clusters) for entity "Iker Casillas" get (incorrectly) merged together after a new value that "connects" the two clusters is extracted from a new document. Based on this important observation, we focus on stable clustering algorithms, such as a complete-link hierarchical agglomerative algorithm [13] or an online single-pass clustering algorithm [22], which have been proved effective and robust for many text-processing tasks [14, 22, 13] (see Section 5.1). In these stable algorithms, new extracted values can be added to existing clusters or, alternatively, they can start new clusters, without collapsing existing clusters. Hence these algorithms are a natural fit for the top-k extraction processing approach.

4. SCORING EXTRACTED VALUES

Once each extraction has been matched in Step 2 to an entity, and to an associated attribute value cluster, we proceed to score the extractions with a scoring function (see Section 2) and update the top-k rankings for each entity (Step 3). As we discussed in [24], and as reflected in Definition 1, different instantiations of the scoring function vary on how they incorporate (or do not incorporate) three main factors: (1) an *extraction confidence* score provided by the extraction system, which indicates the system's internal confidence in the correctness of each individual extraction; (2) the *(entity-specific) document importance* of the documents where an attribute value originates; and (3) the *redundancy* of an attribute value across documents.

With a first instantiation of the scoring function that we will study in this paper, we will capture state-of-the-art, redundancy-based functions that assign higher estimates of correctness to extractions that appear more frequently [7, 8, 11, 26]. Since these functions treat every document equally, we can instantiate the general scoring function to model these existing functions by setting $\gamma_{d,e} = 1$ for all $d \in D$ and $e \in E$. In our experiments, we refer to this generic

function as *Uniform-All*. As we will see, this function leads to high-quality top-k results. Unfortunately, this is achieved at a prohibitively high execution cost, given the absence of any form of document selection or prioritization.

As an alternative to *Uniform-All*, we could direct the extraction effort for each entity exclusively to documents that are strongly associated with the entity. For example, we can determine these associations by analyzing actual entity-specific user-access patterns from search engine logs [24], entity-specific resource directories (e.g., `http://www.daylife.com/topic/Iker_Casillas` for "Iker Casillas"), or via carefully chosen queries (e.g., [2]). Such an instantiation of the scoring function, which we refer to as *Uniform-Ind* in our experiments, is modeled by setting $\gamma_{d,e} = 1$ for documents associated with entity e, and $\gamma_{d,e} = 0$ for the remaining documents. This function has the advantage of focusing the extraction effort on a reduced set of documents for each entity, hence leading to more efficient—but potentially lower-quality—top-k extraction executions than for the *Uniform-All* function. *Uniform-Ind* intuitively classifies each document as relevant or not for an entity, but does not prioritize the extraction among the relevant documents, hence missing a key opportunity towards efficiency.

Our third, and final, alternative scoring function instantiation moves beyond the binary document classification of *Uniform-Ind* and attempts to capture the importance of each document for the top-k extraction process in an *entity-specific* manner. This function, which we refer to as *User-Guided*, allows for arbitrary real values for $\gamma_{d,e}$ between 0 and 1, and is based on the observation that not all documents for an entity are equally reliable, trustworthy, or useful for the extraction of information for the entity. For example, consider two athlete entities, e_1 = "Iker Casillas" and e_2 = "David Beckham." The Wikipedia page for Casillas might have a value $\gamma_{d,e_1} = 0.2$, a relatively high score. In contrast, $\gamma_{d,e_2} = 0$, given that this document is not promising for the extraction of data about Beckham. The exact value for document importance can be determined in a number of different ways. For example, we can consider the "authoritativeness" of a document for an entity [18]. One possible instantiation of importance, whose efficiency implications we studied in [24] and whose effects on extraction quality we explore in this paper (Section 5.1), leverages search engine statistics to examine 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 for an entity, the more likely it contains important information for the entity in question.

Once we have determined the scores of each extraction based on one of the above scoring functions, we then update the top-k values for each entity accordingly. As we discussed in [24], we can define a stopping condition (Step 4) so that the top-k extraction processing sometimes finishes early, for efficiency, once the top-k extracted attribute values for each entity of interest can be guaranteed not to change, even if all unseen documents were to be processed [24].

5. EXPERIMENTS

We now turn to evaluating the impact of the two main extraction quality aspects discussed above, namely, the matching and scoring of extracted values. We first report our experimental settings (Section 5.1) and then summarize our evaluation results (Section 5.2).

5.1 Experimental Settings

Entity sets: We use two entity sets: *politicians* and *athletes*. The politicians set contains 547 United States politicians, with senators, representatives, governors, and a few prominent political figures such as Sarah Palin. The athletes set contains 5128 athletes, extracted from Wikipedia and Yahoo! Sports. We focus on one attribute, *position*, as it is both multi-valued (i.e., each entity typically has multiple positions) and applicable to both entity sets.

Data set: We collected 24,244 Web pages for politicians and 208,637 for athletes. We identified and retrieved these Web pages based on entity-specific access data from a large-scale, three-month, real-user search log from Yahoo! We described the data collection process in detail in [24]. We have made this data set *available to the academic community*, as part of the Yahoo! Labs WebScope program[1].

Extraction system: We use OpenCalais[2], a high-accuracy, state-of-the-art extraction system that is suitable to our entity domains. OpenCalais does not return a confidence value for extractions. Therefore, without impacting the relative performance of the various algorithms, we set $conf(e, a, d) = 1$ (Definition 1).

Training set: We randomly removed 12 politicians and 114 athletes—and their associated Web documents—to use them as a *training set* to tune our techniques.

Scoring functions: We compare the quality associated with the three scoring functions discussed in Section 4, namely, *Uniform-All* (which, as discussed, represents state-of-the-art scoring functions based on extraction redundancy), *Uniform-Ind*, and *User-Guided*. *Uniform-Ind* focuses on a set of entity-specific documents for each entity, while *User-Guided* goes a step further and exploits the entity-specific "importance" $\gamma_{d,e}$ (Definition 1) of each document d for an entity e. For our experiments, we derived this entity-specific information for *Uniform-Ind* and *User-Guided* from the Yahoo! search log [24].

Attribute-value normalization: After extracting an attribute value, we *normalize* it to facilitate the matching against variations of the same value. This "concept filtering" step (Section 3.2) [23] requires a domain-specific training corpus, which we extract from `en.wikipedia.org` category pages. We start in two root category pages, namely, "category:Sportspeople" and "category:Politicians," and collect all descendants, without duplicates. As a result of this process, we extracted 19.5 million unique n-grams for politicians and 81.2 million for athletes, with $n = [1, 4]$.

Attribute-value clustering: We cluster extracted attribute values for an entity in order to match variations of the same values (Section 3.2). As mentioned above, we focus on "stable" algorithms that are compatible with the top-k extraction approach. Specifically, we evaluate three alternatives:

Single-Pass Centroid (Centroid): When a new value is extracted, we compare it against the "centroid" (defined as a weighted average of concepts) of each existing cluster, using the Jaccard coefficient [19]. The new value is added to the closest cluster, if a sufficiently similar cluster—according to an empirically determined threshold θ (see below)—exists (ties are broken in favor of the highest-ranked cluster); otherwise, a new cluster with the new value is created.

[1] `http://webscope.sandbox.yahoo.com/` (data set: ydata-ysearch-location-entity-sources-v1_0).
[2] `http://www.opencalais.com/`

Complete-Link Hierarchical Agglomerative Clustering (Complete-Link): When a new value is extracted, we assign it to a cluster such that *all* of its members are sufficiently close to the new value, according to the Jaccard coefficient and a similarity threshold θ (see below).

Set Containment (Subset): When a new value a is extracted, we assign it to a cluster C such that either a is a subset of all values in C or, alternatively, a superset thereof. If no such cluster exists, we start a new cluster with a. Intuitively, this clustering algorithm ensures that all values in a cluster are represented by one "largest," most informative value.

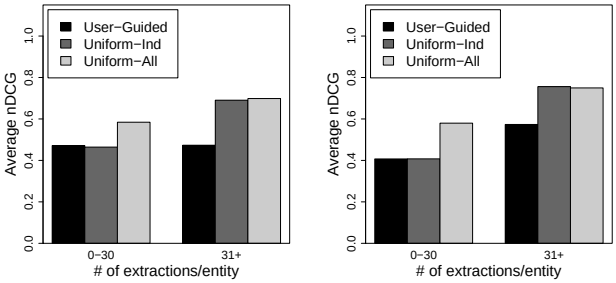
We compared *Centroid*, *Complete-Link*, and *Subset* over the training set, with different values of threshold θ for *Centroid* and *Complete-Link*. Over our training set, we observed that *Complete-Link* with the Jaccard coefficient and a threshold $\theta = 0.5$ resulted in top-k values with the highest average nDCG (see below), or marginally close to the highest, for both domains and across several alternative scoring functions. Therefore, we use only *Complete-Link* with these settings for the rest of the experiments.

Evaluation metrics: We evaluate the *quality* of each entity's top-k values according to the *normalized Discounted Cumulative Gain (nDCG)* metric [17]. This order-sensitive metric considers the rank of correct elements, according to some "ground truth" or *gold standard*, among the top-k values. We built the gold standard value set for each politician and athlete entity from `DBpedia.org`, with additional results for politicians from an online directory[3]. Additionally, we account for the fact that multiple attribute values may refer to the same real-world "fact" (e.g., "goalkeeper" and "goalkeeper, Spain"), so we calculate the number of distinct facts that appear in the top-k values for each entity, giving half credit to "partial facts" (e.g., "goalkeeper" without "Spain"). Therefore, in addition to nDCG, we measure the *fact recall* for each entity, as the fraction of *distinct* extractable facts that appear among the top-k extracted values. Finally, we supplement the DBpedia-based gold standard with a *manual annotation*. We observed that many of the (more obscure) entities were not represented properly in DBpedia. To obtain meaningful results, we then randomly selected a sample of 50 politicians and 250 athletes, and supplemented the gold standard with each correct attribute value—with their correctness determined manually by one of the authors—extracted from the full data set.

5.2 Experimental Results

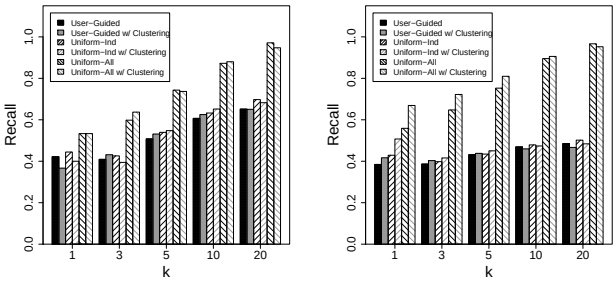
We start by looking at the impact of scoring in terms of the way we account for document importance. Figure 2 presents the average nDCG of top-k results (we focus on $k = 10$ for brevity; different values of k have similar results) over the manually checked extractions (Section 5.1). We split entities into "popular" (with over 30 matched extractions) and non-popular sets, where the former accounts for 20% of the distinct entities but close to 80% of the query volume. Not surprisingly, *Uniform-All* achieves better nDCG than *User-Guided* for both entity sets and regardless of the popularity. This is mainly due to the fact that *User-Guided* directs the extraction effort solely toward highly influential pages such as those on Wikipedia and ignores the rest of the pages, even when they are relevant to the entity of interest. Still,

[3] `http://www.gpoaccess.gov/cdirectory/browse-cd-09.html`.



(a) politicians (b) athletes

Figure 2: nDCG of top-10 results, with entities split by extraction frequency, for (a) politicians and (b) athletes.



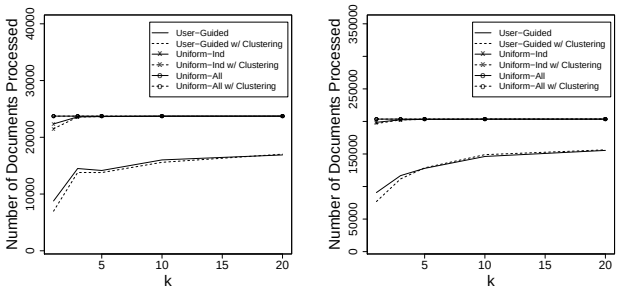
(a) politicians (b) athletes

Figure 3: Fact recall as a function of k for (a) politicians and (b) athletes.

User-Guided manages to recover close to 80% of the nDCG values. More interestingly, *Uniform-Ind* is able to achieve roughly the same level of extraction quality as *Uniform-All* for popular entities but not for non-popular entities. The reason is that, for popular entities, most true values can be extracted from the set of entity-specific documents, which is not the case for non-popular entities.

For the impact of value matching (i.e., clustering), we turn to the fact recall metric as described in Section 5.1. We expect clustering to improve recall by reducing cases where multiple values of the same fact dominate the top-k results, preventing other values from being returned. As shown in Figure 3, the results here are mixed. For the politician entity set, clustering does not provide any significant boost to the recall and can occasionally make things worse by wrongly grouping values for different facts. For the athletes entity set, however, clustering provides a noticeable improvement in recall especially for small values of k, which are (arguably) the most important values. We speculate that the positions of politicians are, in general, referred to consistently across documents, which limits the role of clustering. In contrast, the positions of athletes can be referred to in many different ways (e.g., with emotional descriptors like "amazing"), so the role of clustering is important. We again note that *Uniform-All* achieves the highest recall, which is expected since it processes all the documents, and the other two alternatives perform reasonably well.

Finally, Figure 4 confirms our previous conclusion from [24] that leveraging document importance can significantly speed up the extraction process by reducing the number of documents being processed, at a reasonably cost regarding extraction quality.



(a) politicians (b) athletes

Figure 4: Number of documents processed as a function of k for (a) politicians and (b) athletes.

6. RELATED WORK

Our work builds upon various previous studies on extraction quality [11, 28, 7, 8, 11, 26], which leverage pattern frequency and value redundancy to estimate extraction correctness, and on extraction scalability [4], which focuses on "one-size-fits-all" super-efficient extraction from all documents. The approach that we take in this study and in [24] focuses on domain- and application-specific extraction systems that might co-exist and evolve (e.g., to improve accuracy) over time, so it is critical to carefully select a relatively small set of documents for extraction processing.

Our top-k extraction processing approach, which we introduced in [24], also leverages top-k query processing over structured data (e.g., TA [10] and Upper [20]). For information extraction, Jain and Srivastava [16] showed that, when using state-of-the-art, redundancy-based scoring functions, it is necessary to process most of the available documents to return the true top-k extraction results. They then propose to relax the problem definition and return k values from a proportion of the top results, for efficiency [16]. We showed in [24] that, by defining our scoring function appropriately using entity-specific document weights, we can leverage efficient top-k processing algorithms for efficient executions.

Recent work considers both quality and efficiency in extraction processing [15, 25], and integrates the document processing and extraction management steps, with model parameters to exchange quality for processing efficiency. In [24], we showed that our document-importance scoring functions enabled efficient top-k executions. This paper complements our earlier efficiency study and shows that these top-k executions maintain high-quality results. Our approach is similar in spirit to work by Wu and Marian [26], who query the Web and analyze extraction redundancy from Web documents for the narrower problem of resolving single-valued numerical attribute conflicts efficiently.

7. CONCLUSION

In this paper, we focused on the entity attribute extraction problem, or extracting entities and their attribute values from Web documents. This problem presents both efficiency- and quality-related challenges. Our earlier work [24] studied efficiency challenges, in the context of a top-k extraction approach guided by a measure of entity-specific document "importance." Our current work complements this earlier research by examining the quality of the top-k extraction output. Specifically, we evaluated the impact of two critical extraction steps, namely, the matching of the extracted values as well as the scoring of the extracted information based on document importance. Our experiments show that the design choices for these important extraction components impact the quality of the overall top-k extraction process, and that using clustering and adopting document importance for scoring extracted values can achieve a good balance of extraction quality and extraction efficiency.

8. REFERENCES

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

[2] E. Agichtein and L. Gravano. Querying text databases for efficient information extraction. In *ICDE*, 2003.

[3] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Pasca, and A. Soroa. A study on similarity and relatedness using distributional and WordNet-based approaches. In *NAACL-HLT*, 2009.

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

[5] L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, and S. Vaithyanathan. SystemT: an algebraic approach to declarative information extraction. In *ACL*, 2010.

[6] W. W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Trans. Inf. Syst.*, 2000.

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

[8] D. Downey, O. Etzioni, and S. Soderland. Analysis of a probabilistic model of redundancy in unsupervised information extraction. In *AI*, 2010.

[9] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. In *TKDE*, 2007.

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

[11] Y. Fang and K. C.-C. Chang. Searching patterns for relation extraction over the web: rediscovering the pattern-relation duality. In *WSDM*, 2011.

[12] L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. Text joins in an RDBMS for web data integration. In *WWW*, 2003.

[13] A. Griffiths, L. A. Robinson, and P. Willett. Hierarchic agglomerative clustering methods for automatic document classification. *Journal of Documentation*, 40, 1984.

[14] V. Hatzivassiloglou, L. Gravano, and A. Maganti. An investigation of linguistic features and clustering algorithms for topical document clustering. In *SIGIR*, 2000.

[15] P. G. Ipeirotis, E. Agichtein, P. Jain, and L. Gravano. Towards a query optimizer for text-centric tasks. *TODS*, 2007.

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

[17] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR*, 2000.

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

[19] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

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

[21] A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[22] R. Papka and J. Allen. On-line new event detection using single pass clustering. *UMASS Computer Science Technical Report*, 1998.

[23] A. Parameswaran, H. Garcia-Molina, and A. Rajaraman. Towards the web of concepts: extracting concepts from large datasets. In *PVLDB*, 2010.

[24] M. Solomon, C. Yu, and L. Gravano. Popularity-guided top-k extraction of entity attributes. In *WebDB*, 2010.

[25] D. Z. Wang, M. J. Franklin, M. Garofalakis, and J. M. Hellerstein. Querying probabilistic information extraction. In *PVLDB*, 2010.

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

[27] A. Yates and O. Etzioni. Unsupervised resolution of objects and relations on the web. In *HLT-NAACL*, 2007.

[28] J. Zhu, Z. Nie, X. Liu, B. Zhang, and J.-R. Wen. StatSnowball: A statistical approach to extracting entity relationships. In *WWW*, 2009.