How do they Compare? Automatic Identification of Comparable Entities on the Web

Alpa Jain and Patrick Pantel
Yahoo! Labs, Sunnyvale, CA 94089
alpa, ppantel@yahoo-inc.com

***Abstract*—People love comparing things: from home mortgages and digital cameras to travel destinations and political philosophies. Today, we are mostly limited to browsing documents after issuing comparative queries to Web search engines, such as"15-year vs. 30-year mortgage", "Nikon D90 / Canon 40D", "Oahu or Maui", and "communism vs. fascism". There is an opportunity to improve the search experience by automatically offering comparisons to users. In this paper, we propose a first step towards this goal of *comparative analysis* by mining a broad class of comparable entities from search query logs and a large Web crawl. Example *comparables* that we extract include medicines, appliances, electronics, vacation destinations, and many more. We present an extensive empirical analysis showing that our methods generate comparables with high precision and recall, and showing that Web search query logs are a superior source for mining such entities as compared to Web pages, typically used for extraction tasks. We further compare the performance of our methods with "related entities" reported by Google Sets, and show a gain of 39% in average precision and a gain of 30% in NCDG.**

I. Introduction

Consumers frequently compare products or services in order to make an informed selection. For this task, consumers are increasing relying on web search engines. Search engines receive many explicit queries for comparisons, such as *"Nikon D80 vs. Canon Rebel XTi"* and *"Tylenol vs. Advil"*. Several requests for comparisons, however, are implicit. For example, consider the query *"Nikon D80"*, which hints an ambiguous intent: either the searcher is researching cameras (pre-buying stage), or she is ready to buy a camera (buying stage), or she is looking for product support (post-buying stage). If in the pre-buying stage, the searcher is typically interested in reviews, product specifications, and comparisons with other similar models. In this paper, we present the task of detecting *comparable entities* and generating meaningful comparisons, and we propose semi-supervised information extraction methods for extracting comparables from the Web.

Comparable entities can be extracted from various sources, including: (a) comparison websites such as http://www.cnet.com; (b) unstructured documents such as a webcrawl; and (c) search engine query logs. Web page wrapping methods [10] can be used to extract comparisons from comparison websites. Although high in precision, these methods require manual annotations per web host in order to train the model. Higher coverage sources, such as a full webcrawl, contain comparable entities co-occurring in documents in contexts such as lexical patterns (e.g., *compare X and Y*) and HTML tables. Common semi-supervised extraction algorithms from such unstructured text include distributional methods and pattern-based methods. Distributional methods [11] model the distributional hypothesis [7] using word co-occurrence vectors where two words are considered semantically similar if they occur in similar contexts. The word similarities typically consist of a mixed bag of synonyms, siblings, antonyms, and hypernyms. Teasing out the siblings (which often map to comparable entities) has been addressed using clustering techniques such as Google Sets and CBC [20]. Pattern-based methods [21], [2], [17], [24] learn lexical or lexico-syntactic patterns for extracting relations between words. These are most often used since they directly target a semantic relation given by a set of seeds from the user. Thus, to extract comparable entities, we may give as seeds example pairs, *comparable*⟨Nikon D80, Canon Rebel XTi⟩ and *comparable*⟨Tylenol, Advil⟩.

In this paper, we rely on exploiting Web search query logs and a large webcrawl. Our hybrid method applies both a novel pattern-based extraction algorithm to extract candidate comparable entities as well as a distributional filter for ensuring that resulting comparable entities are semantically similar. We present a detailed experimental analysis showing that our extraction method from query logs performs best and greatly outperforms a strong baseline.

II. Extracting Comparables

An ideal comparables framework requires little manually generated data, represents characteristics of comparisons and classes of comparisons, and has high precision and recall. We formulate our task as that of extracting a comparables relation consisting of tuples of the form $\langle x, y \rangle$, where entities x and y are comparable. Our algorithm follows these steps.

(1) Identify candidate comparable pairs from web pages and query logs using information extraction techniques.
(2) Identify a canonical representation for entities in each pair.
(3) Identify and filter out or demote noisy comparables.

A. Pattern-based Information Extraction

To identify instances of comparables in web pages as well as query logs, we learn extraction patterns. We explored two pattern learning methods, namely, bootstrapped learning method based on [17], and an active selection learning method.

Bootstrapped pattern learning: Bootstrapping methods for information extraction start with a small set of seed tuples from a given relation. The extraction system finds occurrences of these seed instances in plain text and learns extraction patterns based on the context between the attributes of these instances. Extraction patterns are, in turn, applied to text to identify new instances of the relation at hand.

At each iteration, both extraction patterns and identified tuples are assigned a confidence score, and patterns and tuples with sufficiently high confidence are retained. This process continues iteratively until a desired termination criteria (e.g., number of tuples or number of iterations) is reached. Several

IEEE IRI 2011, August 3-5, 2011, Las Vegas, Nevada, USA
978-1-4577-0966-1/11/$26.00 ©2011 IEEE

bootstrapping methods have been proposed in the literature, varying mostly in how patterns are formed and unreliable patterns or tuples are identified and filtered out [2], [17]. For our task, we use the bootstrapping algorithm proposed by Pasca et al. [17], which is effective for large-scale extraction tasks. Using this bootstrapping method, example learned patterns are:

p_1:$\langle E_1 \rangle$ vs. $\langle E_2 \rangle$	p_6: $\langle E_1 \rangle$ is better than your $\langle E_2 \rangle$
p_2:$\langle E_1 \rangle$ versus $\langle E_2 \rangle$	p_7: $\langle E_1 \rangle$ compared to the $\langle E_2 \rangle$
p_3:$\langle E_1 \rangle$ instead of $\langle E_2 \rangle$	p_8: $\langle E_1 \rangle$ to $\langle E_2 \rangle$
p_4:$\langle E_1 \rangle$ will beat $\langle E_2 \rangle$	p_9: $\langle E_1 \rangle$ or $\langle E_2 \rangle$
p_5:$\langle E_1 \rangle$ compared to $\langle E_2 \rangle$	p_{10}:$\langle E_1 \rangle$ over $\langle E_2 \rangle$

TABLE I
PATTERNS USING BOOTSTRAPPING; E_1 AND E_2 ARE COMPARABLES.

While these patterns effectively capture the comparison intent, the resulting output can be noisy due to several reasons. First, generic patterns such as p_{10} tend to match a large fraction of sentences in text collections, and thus, generate a large number of incorrect tuples. For example, applying p_{10} to the text *...jumped over the fence ...* would generate an invalid tuple. Second, lack of prior knowledge about what to expect as an entity further exacerbates the problem. Despite the issue of generic patterns, bootstrapping methods have been successfully deployed for tasks such as, extracting *person-born-in*, *company-CEO*, or *company-headquarters* relations. As the attribute values in such relations are homogeneous, noisy tuples can be potentially identified using named-entity taggers that can identify instances of a pre-defined semantic classes (e.g., organizations, people, location). This, in turn, allows for verifying if values for say the company attribute in a company-CEO relation is an organization or not. In contrast, the attribute value in our comparables relation may belong to a variety of target semantic classes: the tuples, ⟨tea, coffee⟩, ⟨DSL, cable⟩, and ⟨magnolia, Tilia⟩, are all valid instances of the comparables relation, but the values tea, DSL, and magnolia belong to different semantic classes. Due to the iterative nature of this learning process, we expect the quality of the output to rapidly deteriorate after a small number of iterations.

To alleviate this problem of noisy tuples, we need a way to early on identify unreliable tuples. For this, we could employ an active learning framework where humans intervene at each iteration and suggest tuples to be eliminated. Unfortunately, manually annotating each candidate tuple can be cumbersome and thus, instead of identifying noisy tuples we focus on pruning out patterns likely to generate many noisy tuples. Next, we discuss our *active selection* pattern learning method.

Active selection pattern learning: The rationale behind this approach is that although humans find it difficult to recommend or generate patterns for a task, they are generally good at identifying good patterns from bad. With this in mind, we present the top-N ranking patterns to a human and manually select a subset of patterns. As humans are requested to choose from extraction patterns already verified to exist in text, they are likely to generate reliable tuples. We picked a small set of extraction patterns from the top-10 patterns generated by the bootstrapping method. Specifically, we used patterns p_1, p_2, p_4, and p_7 from those listed in Table I To summarize, we extend bootstrapping extraction methods using active selection to learn patterns to generate comparables. We run the resulting extraction methods on two sources, i.e., web pages and query logs. Upon generating the candidate comparable pairs, we identify canonical representations for the entities. Textual data is often noisy or contains multiple non-identical references to the same entity, and therefore, text-oriented tasks need to perform data cleaning. Specific to our task of identifying comparables, new data cleaning issues arise as discused next.

```
GeneratePartialCandidates
Input: (A1, A2, .. Am; B) where Ai, B : string
Output: Candidates : set of pairs {X, Y} X, Y : string
Candidates = ∅;
for i = 0 to m do
    for j = i to m do
        /* ICS */
        if i > 0 then
            Candidates.insert(A[1:i] + A[i+1:j], B + A[i+1:j]);
        end
        /* CIS and SIC*/
        if j > i then
            Candidates.insert(A[i+1:j] + A[1:i], B + A[1:i]);
            Candidates.insert(A[i+1:j] + A[j+1:m], B + A[j+1:m]);
        end
        /* SCI */
        forif j < m then
            Candidates.insert(A[j+1,m] + A[i+1:j], B + A[i+1:j]);
        end
    end
end
return Candidates
/* Procedure to generate all candidate representations */ GenerateAllCandidates
Input: (A1, A2, .. Am; B1, B2, ... Bn) where Ai, Bi : string
Output: Candidates : set of pairs {X, Y} X, Y : string
return GeneratePartialCandidates(A1, .. Am; B[1:n]) +
            GeneratePartialCandidates(B1, .., Bn; A[1:m])
```

Fig. 1. Algorithm to generate candidate representations for a comparable.

B. Identifying Canonical Representations

Appropriately identifying entity boundaries is a critical step in automated information extraction. Consider the case of processing the text, *I prefer tea versus coffee* using pattern p_2 in Table I where after matching the pattern we need to identify a correct representation of the entities to be included in the final tuple. Specifically, this text can result in tuples, such as, ⟨tea, coffee⟩, ⟨prefer tea, coffee⟩, or ⟨I prefer tea, coffee⟩. For text documents such as web pages, a common approach to boundary detection is to pre-process the text using a named-entity tagger (e.g., tag instances of pre-defined set of classes such as, organizations, people, location) or using a text chunker (e.g., tag noun, verb, or adverbial phrases) such as Abney chunker [1]. To allow for arbitrary phrases in comparables relation, we use a text chunker. Specifically, we process the web pages using a variant of Abney chunker [1], and use the phrases in a given chunk as an entity when generating a tuple.

Query logs on the other hand do not yield to text chunkers due to their free-form textual format. Also, the terseness of queries where only keywords are provided introduces new challenges. To understand the data cleaning issues when using queries, below are some examples observed in our experiments:

c_1: Nikon d80 vs. d90
c_2: 15 vs. 30 year mortgage calculator

The above examples underscore two important points: (a) generally, phrases that are common to both entities are specified only once (e.g., nikon in c_1); (b) queries may contain extraneous words that need to be eliminated to generate a clean representation (e.g., calculator in c_2).

Consider a comparable pair $P = \{x, y\}$. To construct a canonical representation for P, we first generate a search space of candidate representations for both x and y and pick the most likely representations for *both* entities combined. Specifically, given a candidate representation γ_x, γ_y for P, we assign a score $R(\gamma_x)$ to γ_x and a score $R(\gamma_y)$ to γ_y and pick the values for γ_x, γ_y that maximizes the following:

$$\langle \gamma_x, \gamma_y \rangle = \underset{\{\gamma_x, \gamma_y\}}{argmax} \{R(\gamma_x) \cdot R(\gamma_y)\} \quad (1)$$

To compute $R(\gamma_i)$ for representation γ_i, we note that this score is high for a well-represented entity, e.g., $R(\text{Nikon d90}) > R(\text{d90})$ and similarly $R(15) < R(15 \text{ year mortgage})$ but $R(15 \text{ year mortgage}) > R(15 \text{ year mortgage calculator})$. We derive $R(\gamma_i)$ as the fraction of queries that contain a representation in a stand-alone form, i.e., query is equal to the representation. Intuitively, users are more likely to query for "nikon d90" than "d90."

We now turn to the issue of generating a search space of representations for a pair P. Instead of considering combinations of terms in the query string in a brute-force manner, we hypothesize that the query strings involving comparable pairs consist of three main sets: (a) a class C, (b) an instance I, and (c) a suffix S. For example, for c_2 $I = \{15 \text{ year}\}$, $C = \{\text{mortgage}\}$, $S = \{\text{calculator}\}$; similarly for c_1, $S = \{\}$, $I = \{\text{d90}\}$, $C = \{\}$. Furthermore, of all six (3!) possible permutations of these sets only four permutations are likely to be used to form queries. Specifically, these four cases are ICS, CIS, SIC, SCI; we eliminate cases ISC and CSI where the instance and class are not juxtaposed. As final canonical representations, we want to rewrite both strings x and y in P in the form IC.

Given a candidate pair $P = \{x, y\}$, we explore the space of representations as follows (see Figure 1): holding one of the strings (x or y) constant, we construct all possible strings for C using the four cases listed above. Each value for C is appended (or prefixed) to the other string that has been held constant. This process is repeated viceversa for the other string. As a concrete example, Table II shows examples of representations for c_2. To summarize, we explore a space of candidate representations for a given pair and pick as the canonical representation the case which maximizes the representation scores for both entities.

C. Distributional Similarity Filters

As a final step towards a well-represented comparables database, we need to check if each comparable pair consists of entities that broadly belong to the same semantic classes. For example, while ⟨Ph.D., MBA⟩ is a valid comparables, ⟨Ph.D., Goat⟩ is not. To support our goal of allowing arbitrary semantic classes to be represented in the comparables relation, we employ methods to identify semantically similar phrases on a large scale. Specifically, we use distributional similarity methods [11] that model the *Distributional Hypothesis* [7].

To model the distributional hypothesis, we process a large corpus of text (e.g., web pages in our case) using a text chunker. Terms are all noun phrase chunks with some modifiers removed; their contexts are defined as their rightmost and leftmost stemmed chunks. We weigh each context f using pointwise mutual information [4]. Specifically, we construct a pointwise mutual information vector $PMI(w)$ for each term w as: $PMI(w) = (pmi_{w1}, pmi_{w2}, \cdots, pmi_{wm})$, where pmi_{wf} is the pointwise mutual information between term w and feature f and is derived as: $pmi_{wf} = log\left(\frac{c_{wf} \cdot N}{\sum_{i=1}^{n} c_{if} \cdot \sum_{j=1}^{m} c_{wj}}\right)$, where c_{wf} is the frequency of feature f occurring for term w, n is the number of unique terms, m is the number of contexts, and N is the total number of features for all terms. Finally, similarity scores between two terms are computed by computing a cosine similarity between their *pmi* context vectors [22].

Case	I	C	S	γ_x	γ_y
ICS	30	year	mortgage calculator	15 year	30 year
	30	year mortgage	calculator	15 year mortgage	30 year mortgage
	30	year mortgage calculator		15 year mortgage calculator	30 year mortgage calculator
	30 year	mortgage	calculator	15 mortgage	30 year mortgage
	30 year	mortgage calculator		15 mortgage calculator	30 year mortgage calculator
	30 year mortgage	calculator		15 calculator	30 year mortgage
	30 year mortgage		calculator	15	30 year mortgage
	30 year mortgage calculator			15	30 year mortgage calculator
SIC	30 year mortgage calculator			15	30 year mortgage calculator
	year	mortgage calculator	30	15 mortgage calculator	year mortgage calculator
	year mortgage	calculator	30	15 year mortgage	year mortgage calculator
	mortgage calculator		30 year	15 mortgage calculator	mortgage calculator
	mortgage	calculator	30 year	15 mortgage	mortgage calculator
	calculator		30 year mortgage	15 calculator	calculator

TABLE II
SEARCH SPACE OF REPRESENTATIONS $\{\gamma_x, \gamma_y\}$ FOR PAIR ⟨15, 30 YEAR MORTGAGE CALCULATOR⟩ FOR TWO CASES.

A natural question that arises is whether distributional similarity methods can be used to generate comparables. While distributional similarity methods could generate comparables, their output also consists of a mixed bag of other semantic relations such as synonyms, siblings, antonyms, and hypernyms. For example, for the word *Apple*, the distributional thesaurus generates: *pear, strawberry, Microsoft, Nintendo, company,* ... Only *Microsoft* in this list would be considered a valid comparable entity. It is noteworthy that the output may contain phrases such as *company* which are distributionally similar to Apple, but again invalid comparables.

Most comparable entities fall under a sibling relation, however teasing these out from a distributional similarity output is difficult. Instead, we rely on a distributional thesaurus to filter the output of relation learning methods, in order to generate a comparables relation. In particular, for each comparable pair (x, y), we check if y exists in the list of similar terms for x or viceversa and eliminate all pairs for which the comparable was not found in this list of similar terms. Alternatively, these scores can also be used to demote invalid pairs.

So far, our discussion focused mostly on a flat list of comparables, i.e., we did not consider the relevance score of a comparable. We studied a variety of functions to score a comparable pair, while accounting for scores from the canonical representation and filtering steps. We found that using a simple frequency-based approach where the number of times a comparable pairs was queried works well; intuitively, aggregating over several independently issued queries can effectively capture the relevance of a comparable.

III. EXPERIMENTAL EVALUATION

A. Data collection

Data sources: We used the following data sets as sources for finding comparable entities:

Web documents (`WB`**)** A collection of 500 million web pages crawled by Yahoo! search engine crawl.

Query logs (`QL`**)** A random sample of 100 million, fully anonymized queries collected by Yahoo! search engine in the first five months of 2009. Of these queries, a 5000 query subset was used as a development set to select a diverse collection of popular entities for our evaluation described in Section III.

Extraction methods: For our experiments, we combined the bootstrapped pattern-learning and active selection algorithms presented in Section II with the two datasets introduced above to generate four techniques in all. We denote each of our systems using a two-letter prefix denoting the dataset (`WB`=web documents; `QL`=query logs) and a two-letter suffix

Method	Nr. of comparables
QL-AS	4,591,343
WB-AS	7,146,982
WB-BT	1,243,121
QL-BT	2,657

TABLE III
SIZE OF COMPARABLES RELATION FOR EACH METHOD.

denoting the extraction method (BT=bootstrapped pattern-learning; AS=active selection). We further generated two variants for each method by turning the distributional filtering stage on and off, denoted by FL when on.

Baseline: We are unaware of any existing system for extracting comparables. Arguably the most well known is Google Sets (http://www.sets.google.com), which returns a broad-coverage ranked ordering of terms semantically similar to a set of queried terms. We use Google Sets as our baseline by issuing each entity in our test set and extracting the list of ranked entities output by the system. We denote this technique as GS. This results in the following extraction systems:

- QL-BT: Bootstrapped pattern-learning over query logs;
- QL-BT-FL: QL-BT with distributional filtering;
- QL-AS: Active selection over query logs;
- QL-AS-FL: QL-AS with distributional filtering;
- WB-BT: Bootstrapped pattern-learning over 500-million document Web crawl;
- WB-BT-FL: WB-BT with distributional filtering;
- WB-AS: Active selection over 500-million document Web crawl;
- WB-AS-FL: WB-AS with distributional filtering; and
- GS: Our strong baseline using Google Sets.

Table III lists the sizes of the relations generated by each method without the distributional filter and Table IV lists some example comparables generated using QL-AS.

Distributional similarity filters: We construct our distributional similarity database following [19]: we POS-tag our WB corpus (500-million documents) using Brill's tagger [3] and chunked it using a variant of the Abney chunker [1]. We built a distributional filter from this chunked corpus using the method outlined in Section II-C.

B. Evaluation metrics

We evaluate the performance of each system using set-based measures, i.e., precision and recall, as well as using rank retrieval measures, i.e., normalized discounted cumulative gain (NDCG) and average precision used in information retrieval.

Recall: Given an entity and a list L of comparables for it, we compute recall as $\frac{|L \cap G|}{|G|}$ where G is a list of ideal comparables.

Precision: Given an entity and a list L of comparables for it, we compute precision as $\frac{\textit{Number of correct entries in } L}{|L|}$. Additionally, we also study the precision values at varying ranks in the list.

Average precision (AveP): Average precision is a summary statistic that combines precision, relevance ranking, and recall. $AveP(L) = \frac{\sum_{i=1}^{|L|} P(i) \cdot isrel(i)}{\sum_{i=1}^{|L|} isrel(i)}$, where $P(i)$ is the precision of L at rank i, and *isrel(i)* is 1 if the comparable at rank i is correct, and 0 otherwise.

Normalized Discounted Cumulative Gain (NDCG): NDCG is commonly used to measure the quality of ranked query results. NDCG examines the fact that ideally, we would like to see good results at early rank positions, and poor quality results at lower rank positions. For a given rank R, NDCG is computed as: $NDCG = \lambda \cdot \sum_{i=1}^{R} \frac{2^{g(i)-1}}{log(i+1)}$, where $g(i)$ is the grade (e.g., 10 for a perfect result, 5 for an average result, etc.) assigned to the result at rank i and λ is a normalization constant computed as the $\sum_{i=1} R \frac{2^{g(i)}}{log(i+1)}$ for a list generated by sorting the results in the best grade order.

Entity	Comparables
401k	ira, pension, sep ira, 457 plan, simple ira, money market funds
density	weight, volume, mass, hardness, temperature, specific gravity
plastic bags	paper bags, canvas, cotton bags
sod	grass, seeds, reseeding, artificial grass
solar panels	wind mill, geothermal, fossil fuels, wind turbines, solar shingles
stocks	corporate bonds, etf, small cap stocks, equities, commodities

TABLE IV
SAMPLE COMPARABLES GENERATED FROM QUERY LOGS.

Domain	Entities
ACT	*dental implants, bahamas, swimming, mba, apartment*
APP	*whirlpool, nikon d80, canon eos 450d, ipod, mac*
AUTOS	*honda accord, ford explorer, toyota camry, bmw, honda civic*
ENT	*britney spears, angelina jolie, obama, new york yankees, the simpsons*
MED	*tylenol, ritalin, ibuprofen, vicodin, claritin*

TABLE V
SAMPLE 25 ENTITIES EVALUATED FOR THE TARGET-DOMAIN EVALUATION.

C. Evaluation methodology

We perform *target-domain* and *open-domain* evaluations.

Target-domain evaluation: Our target-domain evaluation focuses on an in-depth evaluation of various methods for a pre-defined set of entity classes. Due to the tedious nature of evaluation of extraction tasks, we restrict ourselves to five generic classes of entities, namely, *Activities* (ACT), *Appliances* (APP), *Autos* (AUTOS), *Entertainment* (ENT), and *Medicine* (MED). For each domain, we picked five frequently queried entities using the training set. Table V shows these 5 categories along with the entities for each domain.

We conducted two user studies, with 7 participants, to evaluate the quality of results for a given method. Our first user study requested a *gold set* of comparables from participants. Given an entity e, participants provided two distinct comparables deemed relevant to e. If the e or its domain was unknown to a participant, we allowed the participant to research on the Web and provide an informed comparable. For example, for *Nikon d80*, users provided *Canon rebel xti, Nikon d200, Fujifilm Finepix z100*.Our second user study requested users to judge the quality of the comparables on a *three-point grades* scale. Starting with an entity, we generated a ranked list of top-5 comparables from each system to be evaluated. We took a union of these lists and presented it to each participant. Participants were asked to rate each comparable in the list as G for good,F for fair, or B for bad. Each user was requested about 350 annotations, and overall, our user study yielded 2,450 annotations (inter-annotator agreement using Fleiss's kappa [13] ranged between 0.41 and 0.54 indicating a moderate agreement). Most of the disagreement could be traced to cases marked F or B, however, for cases marked as G we observed high kappa values indicating substantial agreement. For each entity, we picked a final grade based on the majority opinion of the judgments, and in case of disagreement, we requested an additional judgment. Using these manual annotations, we generated another gold set of *graded* comparables which was used to compute the NDCG values. We also computed precision at varying rank and average precision of each list by assigning a score of 1 to all comparables that were marked G and a score of 0 to the rest. Note that all comparables graded as fair were also assigned a score of 0.

Open-domain evaluation: Our open-domain evaluation moves away from a target domain and examines the quality of comparables using a random sample of the output generated

by each system. Specifically, we draw a sample of pairs of comparables generated by each method, verify them, and study the precision and nature of errors for each method.

D. Target-domain Evaluation Results

Recall: Our first experiment was to measure the extent to which each method identifies comparables desired by our user study participants. For each entity in our test set (see Table V), we generated a ranked list of comparables for each method (i.e., QL−AS, WB−AS, WB−BT, ···) and computed the recall of these lists (see Section III). Table VI compares the recall of all eight methods against that of GS, and the bold-faced numbers mark the techniques with highest recall value for a domain. QL−AS exhibits highest and QL−AS−FL exhibits close to highest values for recall, suggesting query logs as a comprehensive source for generating comparables. As expected, using a filtering step results in a small drop in the recall values across all methods and all domains. While this drop is relatively small, the case for APP needs further discussion. Among the appliances, the query *Canon EOS 450d* was not found in the distributional thesaurus and this resulted in degrading recall for this category. This could be traced to errors by the text chunker. As hypothesized, using bootstrapping methods to learn extraction patterns in both cases, query logs and web documents, results in low coverage of comparables.

We now examine the effect of introducing the filtering step. In our experiments, we observed that the overall quality of the output lists substantially improved when using the distributional thesaurus (Section II-C) as a filter. As a concrete example, for the entity *britney spears* the comparables generated by WB−AS included, *paris hilton* and *bff paris hilton* (bff="best friends forever"). Interestingly, the phrase *bff paris hilton* occurs frequently enough to be ranked higher, and furthermore, our canonical representation generation method also finds enough support for this entity. The filtering method on the other hand, eliminates this entity. To show the improvements by using a filter, we compare the fraction of gold set entities that were returned among the top-10 comparables returned by each method. Intuitively, a good system should return these entities early on. Table VII shows the percentage of gold set comparables found in top-10 results for each method, averaged over all domains. For QL−BT, we observe an increase in the percentage of gold set comparables that are covered when using a filter, with the exception of the case we discussed above. This indicates that the filtering step effectively demotes noisy tuples and, in turn, boosts the ranks for reliable comparables. In case of WB−BT, we observe a relatively small improvement for a few cases. Low performing methods QL−BT and WB−BT are more sensitive to the filter due to the already small values of recall. In what follows, we focus on the competing methods, namely, QL−AS−FL, WB−BT−FL, WB−AS−FL, and GS.

Rank order precision: We now examine the accuracy of each technique using precision. Figure III-D shows the precision for each system at varying rank, for each domain, averaged across all entities in that domain. QL−AS−FL results in a perfect precision (precision = 1.0) or close to perfect precision across domains. The less than perfect precision for APP, can be explained by an example of *nikon d80*: the system returned *canon* as a comparable at rank 1, which was graded as F by our annotators which is treated as incorrect when computing the precision. All other comparables generated for this entity were marked G. Comparing WB−AS−FL and WB−BT−FL we observe that, as hypothesized, using active selection to identify reliable patterns substantially improves the extraction performance. As compared to GS, both QL−AS−FL and WB−AS−FL consistently outperform GS, across all domains.

Method	ACT	APP	AUTOS	ENT	MED
GS	0.37	0.32	0.50	0.62	0.47
QL−AS	**0.77**	**0.90**	**0.87**	**0.95**	**0.90**
WB−AS	0.55	0.37	0.40	0.58	0.52
QL−BT	-	0.22	0.03	0.02	0.10
WB−BT	0.07	0.12	0.03	0.20	0.22
QL−AS−FL	0.62	0.35	0.78	0.72	0.85
WB−AS−FL	0.33	0.22	0.40	0.43	0.52
QL−BT−FL	-	0.13	0.03	-	0.02
WB−BT−FL	0.05	0.05	-	0.07	0.12

TABLE VI
AVERAGE RECALL FOR EACH METHOD, FOR EACH CATEGORY, MEASURED USING A USER-PROVIDED GOLD SET.

Method	ACT	APP	AUTOS	ENT	MED
GS	34	54	60	72	54
QL−AS	56	82	62	64	84
WB−AS	58	56	46	58	58
QL−BT	-	5	4	2	12
WB−BT	4	26	2	18	26
QL−AS−FL	76	48	68	70	94
WB−AS−FL	58	56	66	56	62
QL−BT−FL	-	4	4	-	2
WB−BT−FL	2	26	-	18	14

TABLE VII
AVERAGE PERCENTAGE OF USER-PROVIDED GOLD SETS IDENTIFIED IN TOP-10 RESULTS RETURNED BY EACH SYSTEM.

Table VIII compares NDCG@5 values for each method, across all entities and target domains. † marks NDCG values that are a statistically significant improvement over the baseline of GS. Both QL−AS−FL and WB−AS−FL exhibit a significant improvement of 30% and 20% gain, respectively, over the existing approach of using Google Sets. Table IX shows the NDCG@5 values for each of the five target domains. Interestingly, for the domain of ACT, using an approach based on related words as in the case of GS, proves to be undesirable. This conforms our earlier observations that using distributional similarty-based methods suffer from being too generic for the task of comparables. As a specific example, for the entity *apartment*, GS generates the following comparables, *1 bathroom, washing machine, 2 bathrooms* which were consistently graded as B by all participants in our user studies. In contrast, QL−AS−FL generates comparables, such as, *condominium, house, townhouse* which were graded as G by our participants. We observed similar results for NDCG@10.

Table VIII compares the average precision (AveP) values for each method and † marks statistically significant improvement over GS. (Recall that AveP summarizes the precision, recall, and rank ordering of a ranked list.) Both QL−AS−FL and WB−AS−FL exhibit a significant improvement of 39% and 36% gain, respectively, over GS. As expected, QL−AS−FL exhibits highest values for AveP confirming the choice of active selection over query logs as a promising direction.

E. Open-domain Evaluation Results

The average precision of the relations based on our samples for QL−AS−FL, WB−AS−FL, and WB−BT−FL, were 86%, 67%, and 43%, respectively. Table X shows examples of erroneous facts. A common error observed across all sources is the case where two entities are to be disambiguated: increasingly, people

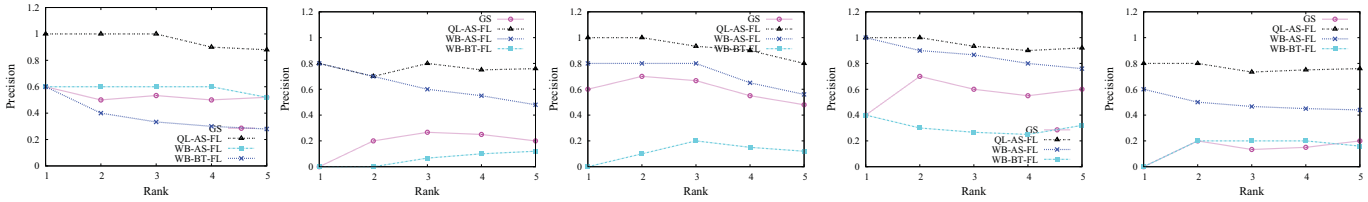

Fig. 2. Precision at varying rank for (a) AUTOS, (b) APP, (c) ENT, (d) MED, and (e) ACT.

Method	NDCG@5	AveP
GS	0.67± 0.11	0.56± 0.13
QL-AS-FL†	0.96± 0.03	0.92± 0.05
WB-AS-FL†	0.86± 0.06	0.86± 0.08
QL-BT-FL	0.54± 0.12	0.38± 0.12

TABLE VIII
AVERAGE NDCG@5 AND AVERAGE PRECISION (AVEP) OVER ALL CATEGORIES. († INDICATES STATISTICAL SIGNIFICANCE OVER GS.)

Category	ACT	APP	AUTOS	ENT	MED
GS	0.35	0.51	0.85	0.85	0.80
QL-AS-FL	**0.93**	**0.91**	**0.99**	**1.00**	**0.99**
WB-AS-FL	0.81	0.77	0.86	0.93	0.97
QL-BT-FL	0.44	0.47	0.72	0.41	0.62

TABLE IX
AVERAGE NDCG@5 FOR EACH CATEGORY USING A 3-POINT GRADE.

make use of search engines as substitutes for a dictionary, thesaurus, spell checker: e.g., 'affect vs effect' or 'cieling vs. ceiling' or 'aptitude vs. ability'. Identifying and correctly addressing such user requests is future work.

Among the valid instances, a large proportion of comparables involve matches between countries or people(e.g., Barcelona vs. Chelsea). In particular, 18%, 23%, 17%, of the sample for QL-AS-FL, WB-AS-FL, WB-BT-FL, respectively, involved a match. Related to this, are instances involving court cases (e.g., Brown vs. Board of Education) which were rarely observed (e.g., 1 among our sample for QL-AS-FL). While these are valid comparables, as an extension of our work, we plan to classify such comparables. To understand this, comparables for *Hong Kong* contain *Shanghai* and *New York*, however, at third position, we observe *Bahrain* due to soccer matches.

IV. RELATED WORK

The problem of automatically extracting structured information from text documents has received significant attention in recent years, in part spurred by the Message Understanding Conferences (MUC). Earlier approaches to building information extraction systems relied on hand-crafted extraction rules [6]. Recent efforts have automated the task of generating extraction rules using bootstrapping methods [2], [17]. Extraction systems based on machine-learning and statistical methods have also been extensively studied [5], [23], [14]. These methods rely on a set of labeled examples of the extraction task and automatically learn extraction rules that maximize the output quality over these examples. Oftentimes, the main difficulty in using such supervised methods to build an information extraction system lies in the tedious task of generating sufficiently many labeled examples. To address this shortcoming, semi-supervised methods have also been studied, which aim to reduce the amount of necessary labeled data [12]. In general, existing solutions have considered the construction of reliable extraction systems for well-defined relations with homogeneous attribute values (e.g., Company-Headquarters, Company-CEO, Person-Born-In.) In this paper, we focused on a bootstrapping method and adapted methods proposed by Pasca et al. [17] for the task of mining comparables. Several specialized extraction tasks have been successfully investigated and our work is similar in spirit to such settings. Examples include building a large scale collection of acronyms and their expansions [15], and identifying sentiments and reviewer opinions [16].

QL-AS-FL	⟨capital, debt⟩,⟨christian, americans⟩, ⟨bias, biased⟩
WB-AS-FL	⟨ignored, hated⟩, ⟨fieling, feiling⟩, ⟨game, company⟩
WB-BT-FL	⟨best case, worse case⟩, ⟨directors, writers⟩, ⟨carmel, caramel⟩

TABLE X
INCORRECT COMPARABLES FROM VARIOUS SOURCES.

Our work heavily relies on using query logs to gather structured information. Increasingly, research efforts are looking into exploiting valuable information available in query logs. For instance, [18] showed how interesting attributes can be derived from user queries. We believe this method is complementary to our approach and can be used in concert with our comparables database generation methods to build descriptions of comparables. A preliminary version of our work [8] (a "poster paper") briefly introduces the generic framework without giving details on how to build such a framework. In this paper, we focus on building an end-to-end system for building a comparables database. [9] uses supervised approach to a related problem of identifying comparitive *sentences* in documents; in contrast our goal is provide comparable entities mined from both query logs and web documents.

V. CONCLUSION

This paper introduced a new web search paradigm that allows users to carry out comparative analysis. Our methods work hand-in-hand with existing information extraction and semantic similarity identification techniques to build a comprehensive yet precise collection of comparable entities. Many interesting research problems such as generating self-explanatory comparables, ranking unreliable comparables, etc. remain open.

REFERENCES

[1] S. Abney. Learning taxonomic relations from heterogeneous sources of evidence. In *Principle-Based Parsing*. Kluwer Academic Publishers, 1991.
[2] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *DL*, 2000.
[3] E. Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4), 1995.
[4] K. Church and P. Hanks. Word association norms, mutual information, and lexicography. In *ACL*, 1989.
[5] W. Cohen and A. McCallum. Information extraction from the World Wide Web (tutorial). In *KDD*, 2003.
[6] O. Etzioni, M. J. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in KnowItAll (preliminary results). In *Proceedings of WWW-04*, 2004.
[7] Z. Harris. Distributional structure. *Word*, 10(23):146–162, 1954.
[8] A. Jain and P. Pantel. Identifying comparable entities (poster paper). In *CIKM*, 2009.
[9] N. Jindal and B. Liu. Identifying comparative sentences in text documents. In *SIGIR*, 2006.
[10] A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. A brief survey of web data extraction tools. *SIGMOD Record*, 31, 2002.
[11] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of ACL/COLING-98*, 1998.
[12] I. Mansuri and S. Sarawagi. A system for integrating unstructured data into relational databases. In *ICDE*, 2006.
[13] J. P. Marques De Sá. *Applied Statistics*. Springer Verlag, 2003.
[14] A. McCallum and D. Jensen. A note on the unification of information extraction and data mining using conditional-probability, relational models. In *IJCAI*, 2003.
[15] Nadeau and P. Turney. A supervised learning approach to acronym identification. In *AI*, 2005.
[16] K. Nigam and M. Hurst. Towards a robust metric of opinion. In *AAAI-EAAT*, 2004.
[17] M. Paşca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Names and similarities on the web: Fact extraction in the fast lane. In *Proceedings of ACL06*, July 2006.
[18] M. Paşca and B. Van Durme. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *ACL-HLT*, 2008.
[19] P. Pantel, E. Crestan, A. Borkovsky, A.-M. Popescu, and V. Vyas. Web-scale distributional similarity and entity set expansion. In *Proceedings of EMNLP-09*, 2009.
[20] P. Pantel and D. Lin. Discovering word senses from text. In *SIGKDD*, 2002.
[21] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of AAAI-99*, 1999.
[22] G. Salton and M. J. Mcgill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.
[23] S. Sarawagi and W. Cohen. Semimarkov conditional random fields for information extraction. In *21st International Conference on Machine Learning (ICML 2004)*, 2004.
[24] Y. Yan, Y. Matsuo, Z. Yang, and M. Ishizuka. Unsupervised relation extraction by mining wikipedia texts with support from web corpus. In *Proceedings of ACL-09*, 2009.