Identifying Comparable Entities on the Web

Alpa Jain
Yahoo! Labs
701 First Avenue
Sunnyvale, CA 94089
alpa@yahoo-inc.com

Patrick Pantel
Yahoo! Labs
701 First Avenue
Sunnyvale, CA 94089
me@patrickpantel.com

ABSTRACT

Web search engines are often presented with user queries that involve comparisons of real-world entities. Thus far, this interaction has typically been captured by users submitting appropriately designed keyword queries for which they are presented a list of relevant documents. Richer interactions that explicitly allow for a *comparative analysis* of entities represent a new potential direction to improve the search experience. With this in mind, we present an initial step of mining comparable entities from sources of information available to a large-scale Web search engine, namely, search query logs and documents from a Web crawl. Our mining methods generate a diverse set of *comparables* consisting of entities from a broad class of categories, such as medicines, appliances, electronics, and vacation destinations.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*dictionaries*

General Terms

Algorithms, Design, Experimentation.

Keywords

Information extraction, comparables, query logs.

1. INTRODUCTION

Which should I choose? Consumers frequently compare products or services in order to make an informed selection. For this task, consumers are increasing relying on the Internet and 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"*, for which the user intent is ambiguous: 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 large-scale semi-supervised information extraction methods for extracting comparables from the Web.

Web search engines can greatly benefit from learning comparable entities. Knowing the comparable cameras to *"Nikon D80"*, a search engine could propose appropriate recommendations via query suggestions (e.g., by suggesting the query *"Nikon D80 vs. Canon Rebel XTi"*). From an advertisement perspective, knowing the comparables to *"Nikon D80"* would allow generating a diverse set of advertisements including both, for example, sellers of *"Nikon D80"* and sellers of *"Canon Rebel XTi"*. Access to a large database of comparable entities would allow a search engine to better interpret the intent behind queries consisting of multiple entities. For example, consider the query *"Tilia magnolia"*[1]. Finding these two entities in the comparable database would be a strong indicator of *comparison intent*. If we further knew how to generate a meaningful comparison between the two, a search engine could trigger a direct display illustrating a comparison chart between them.

In this paper, we propose a general framework for comparative analysis and take a first step towards automatically mining a large-scale knowledge base of comparable entities by exploiting several resources available to a Web search engine, namely query logs and a large webcrawl.

2. ENABLING COMPARATIVE ANALYSIS

In order to support the applications described in the previous section, a comparables framework must satisfy several criteria. First, we need automated methods to identify and extract comparable real-world entities with as little human effort as possible. Manually generating each comparable tuple is, of course, tedious and prohibitively time consuming. Second, the framework must be capable of representing not only comparable entities but also interesting relationships between entities, such as, characteristics of comparison, classes of comparison, etc. Third, the information used by the framework must appropriately capture a variety of entities as well as a variety of textual resources.

The overall architecture of the query processing method that we envision is as shown in Figure 1. Search engine

[1]Tilia and magnolia are large deciduous flowering trees.

CIKM'09, November 2–6, 2009, Hong Kong, China.

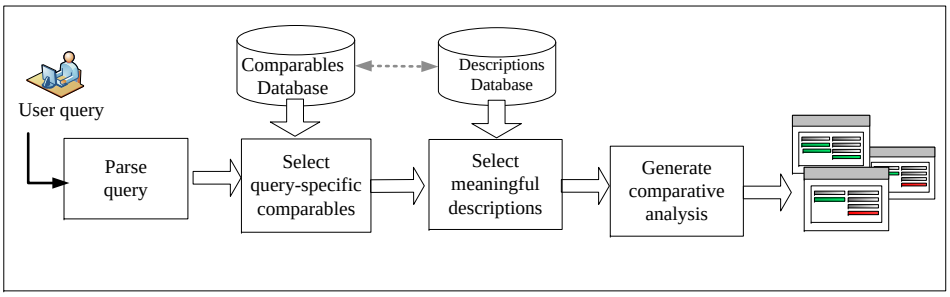

Figure 1: Architecture of a query processing method that enables users to perform comparative analysis.

users interact with the search interface by presenting keyword queries intended to (implicitly or explicitly) compare entities. Starting with a user-specified keyword query, the query execution consists of four main stages:

Step (1), Parse query: Our first step is to classify whether the primary intent of the query is comparison. For this, we can follow one of the several efforts proposed in the past. Specifically, we follow a dictionary-based approach that uses a large collection of sets of comparables to "lookup" terms in the user query.

Step (2), Select comparables: Upon identifying an entity or list of entities mentioned in the query, the next step is to generate a list of comparables relevant to these entities. For this, we can either use an *offline* approach, where comparables are mined, cleaned, and well-represented in a database, or use an *online* approach, where we process only the web pages that match the user query at query execution time. In our prototype, we follow an offline approach of materializing an entire relation of comparables. The rationale behind this choice is that information regarding comparables often spans a variety of sources, such as, web pages, forum discussions, query logs and tapping into such a variety of resources at query execution time could be expensive and unfeasible. Furthermore, focusing only on the information buried in the search results may be too restrictive and result in incomplete information[2]. To the best of our knowledge, we are not aware of any "off-the-shelf" resources that build or provide a diverse set of comparables. Thus, we rely on using *information extraction* methods which focus on automatically identifying information embedded in unstructured text (e.g., web pages, news articles, emails). As we will see, information extraction methods are often noisy and require source-specific and source-independent post processing. As an additional requirement, instead of providing a flat set of comparables, the database must return a ranked list of comparables; oftentimes, an entity is associated with multiple comparables (e.g., in our experiments, we identified more than 50 comparables for honda civic), and not all comparables may be highly relevant. Therefore, a well represented comparables database must include a *relevance score* attached to each comparable tuple.

Step (3), Select descriptions: Output from extraction systems, unfortunately, rarely contains sufficient information that allow consumers to fully understand the content. In the context of serving comparables, users will not only be interested in learning about comparables but also in knowing the descriptions of these comparisons. To make the results from a comparative analysis self-explanatory, the third step in our framework focuses on providing meaningful descriptions for each pair of comparables identified in Step 2. These descriptions may include information such as, characteristics or attributes that are common to the description of entities (e.g., resolution when comparing cameras), attributes that are *not* common to these entities (e.g., crime alerts when comparing vacation destinations), or reliable sources for extended comparisons (e.g., relevant forums or blogs). Just as in the case of comparables, descriptions must also be assigned a relevance score to identify reliable descriptions from the less reliable ones.

Step 4, Enhance search results: The final step is to enrich search results by introducing comparables and descriptors from Steps 2 and 3.

In this paper, we focus on building a comparables database used in Steps 1 and 2. Automatically building large collections of comparable entities involves several challenges. First, we need to build methods for reliably harvesting large collections of comparables. As we will see, using state-of-the-art information extraction methods can result in significant amount of noise in the output due to the fairly generic nature of our task. Second, text often contains discussion on comparisons of entities along with additional information that must be eliminated to improve the quality of the comparables database. For instance, phrases involving attributes of comparison, (e.g., price, rates, gas mileage) or phrases representing the class that the entities belong to (e.g., camera in the case of Nikon d80, or car in the case of ford explorer) often occur in the proximity of comparable entities. Finally, following most extraction tasks, we must automatically identify tuples with lower confidence from those with higher confidence. This task is generally carried out by exploiting some prior knowledge about the domain of the value to expect. However, as comparables entities may belong to a diverse set of domains (e.g., medicine, autos, cameras, etc.), we need to build generic domain-independent filters that effectively remove noisy tuples from the comparables relation.

3. EXTRACTING COMPARABLES

A straightforward approach towards mining comparables is to use wrapper induction [5], where we create customized wrappers to parse web pages of websites dedicated to comparisons, such as `http://www.cnet.com`. While wrapper induction methods are generally high in precision, they require manually annotating a sample of web pages for each web-

[2]A third alternative is to follow a hybrid approach that combines both the offline and the online approach.

Entity	Comparables
15 year mortgages	30 year mortages
401k	ira, pension, sep ira, 457 plan, simple ira, saving, money market funds
basement	crawlspace, cellar, attic
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, currency, commodities, bonds in 401k
termite	flying ant, worms, formosan termites, ant flies
vinegar	hydrogen peroxide, sodium chloride solution, salt, ascorbic acid, mouthwash, borax, alcohol, amonia

Table 1: Sample comparables generated using extraction methods over query logs.

site, and this manual labor is linear in the number of sites to process. Also, it may be difficult to find websites containing comparables of non-product entities, such as ⟨socialism, capitalism⟩ and ⟨mojito, caipirinha⟩. As an alternative, several domain-independent *information extraction* methods that focus on identifying instances of a pre-defined relation from plain text documents have been proposed [1, 2]. We follow the information extraction approach which we discuss in this section.

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 comparable pair.
(3) Identify and filter out or demote noisy comparables.

To automatically identify candidate comparable pairs, we employ information extraction techniques over a large collection of web pages as well as query logs. Specifically, we learn extraction patterns that capture the context generally associated with the comparable items in natural-language text, using a bootstrapped pattern learning method. Bootstrapping methods 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 instances. For instance, given a seed instance ⟨Depakote, Lithium⟩ which occurs in the text, *My doctor urged me to take Depakote instead of Lithium*, the system learns the pattern, "$\langle E_1 \rangle$ instead of $\langle E_2 \rangle$." Extraction patterns are, in turn, applied to text to identify new instances of the relation at hand. For instance, the above pattern when applied to the text, *Should I buy stocks instead of bonds?* can generate a new instance, ⟨stocks, bonds⟩.

At each iteration, both extraction patterns and identified tuples are assigned a confidence (or relevance) score, and patterns and tuples with sufficiently high confidence are retained. This process continues iteratively until a desired termination criteria is reached. Several bootstrapping methods have been proposed in the literature, varying mostly in how patterns are formed and how unreliable patterns or tuples are identified and filtered out. For our task, we use the bootstrapping algorithm proposed by Paşca et al. [10], which is effective for large-scale extraction tasks.

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, generally text-oriented tasks need to perform data cleaning. Query logs, due to their free-form textual format and terse nature where only keywords are provided, introduce new challenges. To understand the data cleaning issues when processing query logs, consider the following 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). To effectively generate canonical representations of comparables generated from the query logs, we build novel algorithms that explore a large search space of representations and chooses the best option. We omit the details on our algorithms to build canonical representations of comparable entities due of space limitations.

As a final step towards building a well-represented comparables database, we need to check if each comparable pair consists of entities that broadly belong to the same semantic class. However, to allow arbitrary semantic classes to be represented in our comparables relation, we rely on methods that use Web-based statistics to compute the semantic similarity between entities in a comparable pair. Specifically, we employ a distributional similarity calculator that computes the semantic similarity between all words over a large crawl of 600 million webpages [**?**].

4. 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 [4]. Recent efforts have automated the task of generating extraction rules using bootstrapping methods [1, 10, 12]. Extraction systems based on machine-learning and statistical methods have also been extensively studied [3, 13, 7]. 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

Data source	Comparables
Query logs	⟨capital, debt⟩,⟨christian, americans⟩, ⟨bias, biased⟩
Web pages	⟨ignored, hated⟩, ⟨fieling, feiling⟩, ⟨game, company⟩

Table 2: Incorrect comparables extracted from various sources.

also been studied, which aim to reduce the amount of necessary labeled data [6]. 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. [10] 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 [8], and identifying sentiments and reviewer opinions [9].

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, [11] 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.

5. DISCUSSION

We performed a detailed experimental analysis by applying our mining methods to a collection of query logs extracted over a period of four months, as well as a large webcrawl of 600 million documents. As an example, Table 3 lists some comparables generated by our extraction methods for query logs. We evaluated the performance of our methods using set-based metrics, such as, precision and recall, as well as using rank retrieval measures, such as, normalized discounted cumulative gain and average precision. We do not report our results because of space constraints. As a summary of our evaluation conclusions, our extraction methods from query logs performs best and greatly outperforms a strong baseline in terms of precision as well as recall, for a variety of target domains.

As most information extraction methods suffer from erroneous output, we performed a detailed error analysis of our comparables relation. As an example, Table 2 shows examples of errornous facts for the two main sources, namely, Web pages and search query logs. A common error observed is the case where two entities are to be disambiguated: increasingly, people make use of search engines as substitutes for a dictionary, thesaurus, and spell checker. In particular, queries that try to disambiguate the meaning of words or find the correct spelling are interesting here, as they cause a recognizable class of errors for our algorithms; consider for example, 'affect vs effect' or 'cieling vs. ceiling' or 'aptitude vs. ability'. Among the valid instances, a large proportion of comparables involve matches between countries or people (e.g., Barcelona vs. Chelsea). Related to this are instances involving court cases (e.g., Brown vs. Board of Education), which were rarely observed. While these are valid comparables, as an extension of our work, we plan to classify such comparables. To understand this, consider the entity *Hong Kong* for which the results contain *Shanghai* and *New York*, however, at third position, we observe *Bahrain* due to soccer matches.

In summary, 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 techniques and semantic similarity identification techniques to build a comprehensive yet precise collection of comparable entities. Our work in the paper has so far only established the foundations of this area, and many interesting research problems remain open in this line of research. One such problem is to design methods that generate self-explanatory comparables. Another important problem is to explore further techniques to rank and score unreliable comparables without sacrificing the generality of the relation.

6. REFERENCES

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

[2] S. Brin. Extracting patterns and relations from the world wide web. In *WebDB*, 1998.

[3] W. Cohen and A. McCallum. Information extraction from the World Wide Web (tutorial). In *KDD*, 2003.

[4] 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.

[5] N. Kushmerick, D. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *IJCAI*, 1997.

[6] I. Mansuri and S. Sarawagi. A system for integrating unstructured data into relational databases. In *ICDE*, 2006.

[7] A. McCallum and D. Jensen. A note on the unification of information extraction and data mining using conditional-probability, relational models. In *IJCAI*, 2003.

[8] Nadeau and P. Turney. A supervised learning approach to acronym identification. In *AI*, 2005.

[9] K. Nigam and M. Hurst. Towards a robust metric of opinion. In *AAAI-EAAT*, 2004.

[10] 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.

[11] 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.

[12] P. Pantel and M. Pennacchiotti. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of ACL/COLING-06*, pages 113–120. Association for Computational Linguistics, 2006.

[13] S. Sarawagi and W. Cohen. Semimarkov conditional random fields for information extraction. In *21st International Conference on Machine Learning (ICML 2004)*, 2004.