Exploiting Web Search to Generate Synonyms for Entities

Surajit Chaudhuri Venkatesh Ganti Dong Xin
Microsoft Research
Redmond, WA 98052
{surajitc, vganti, dongxin}@microsoft.com

ABSTRACT

Tasks recognizing named entities such as products, people names, or locations from documents have recently received significant attention in the literature. Many solutions to these tasks assume the existence of reference entity tables. An important challenge that needs to be addressed in the entity extraction task is that of ascertaining whether or not a candidate string approximately matches with a named entity in a given reference table. Prior approaches have relied on string-based similarity which only compare a candidate string and an entity it matches with. In this paper, we exploit web search engines in order to define new similarity functions. We then develop efficient techniques to facilitate approximate matching in the context of our proposed similarity functions. In an extensive experimental evaluation, we demonstrate the accuracy and efficiency of our techniques.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

General Terms

Algorithms

Keywords

Synonym Generation, Entity Extraction, Similarity Measure, Web Search

1. INTRODUCTION

Tasks relying on recognizing entities have recently received significant attention in the literature [10, 12, 2, 14, 11, 9]. Many solutions to these tasks assume the existence of extensive *reference entity tables*. For instance, extracting named entities such as products and locations from a reference entity table is important for several applications. A typical application is the *business analytics and reporting* system which analyzes user sentiment of products. The system periodically obtains a few review articles (*e.g.*, feeds from review website and online forums), and aggregates user reviews for a reference list of products (*e.g.*, products from certain manufacturers, or products in certain categories). Such a reporting application requires us to effectively identify mentions of those reference products in the review articles.

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

Consider another application. The entity matching task identifies entity pairs, one from a reference entity table and the other from an external entity list, matching with each other. An example application is the *offer matching* system which consolidates offers (*e.g.*, listed price for products) from multiple retailers. This application needs to accurately match product names from various sources to those in the system's reference table, and to provide a unified view for each product.

At the core of the above two applications, the task is to check whether or not a *candidate string* (a sub-string from a review article or an entry from an offer list) matches with a member of a reference table. This problem is challenge because users often like to use phrases, which are not member of the reference table, to refer to some entities. These phrases can be an individual's preferred description of an entity, and the description is different from the entity's conventional name included in a reference table. For instance, consider a product entity "Lenovo ThinkPad X61 Notebook". In many reviews, users may just refer to "Lenovo ThinkPad X61 Notebook" by writing "Lenovo X61", or simply "X61". Exact match techniques, which insist that sub-strings in review articles match exactly with entity names in the reference table, drastically limit their applicability in our scenarios.

To characterize whether or not a candidate string matches with a reference entity string, an alternative approach is to compute the string similarity score between the candidate and the reference strings [10, 6]. For example, the (unweighted) Jaccard similarity[1] function comparing a candidate string "X61" and the entity "Lenovo ThinkPad X61 Notebook" would observe that one out of four distinct tokens (using a typical white space delimited tokenizer) are common between the two strings and thus measures similarity to be quite low at $\frac{1}{4}$. On the other hand, a candidate string "Lenovo ThinkPad Notebook" has three tokens which are shared with "Lenovo ThinkPad X61 Notebook", and thus the Jaccard similarity between them is $\frac{3}{4}$. However, from the common knowledge, we all know that "X61" does refer to "Lenovo ThinkPad X61 Notebook", and "Lenovo ThinkPad Notebook" does not because there are many models in the ThinkPad series. We observe a similar problem with other similarity functions as well. Therefore, the string-based similarity does not often reflect the "common knowledge" that users generally have for the candidate string in question.

[1]These similarity functions also use token weights, say IDF weights, which may in turn depend on token frequencies in a corpus or a reference table.

In this paper, we address the above limitation. We observe that the "common knowledge" is often incorporated in documents within which a candidate string is mentioned. For instance, the candidate string "X61" is very highly correlated with the tokens in the entity "Lenovo ThinkPad X61 Notebook". And, many documents which contain the tokens "X61" also mention within its vicinity the remaining tokens in the entity. This provides a stronger evidence that "X61" matches with "Lenovo ThinkPad X61 Notebook". In this paper, we observe that such "correlation" between a candidate string τ and an entity e is seen across multiple documents and exploit it. We propose new document-based similarity measures to quantify the similarity in the context of multiple documents containing τ. However, the challenge is that it is quite hard to obtain a large number of documents containing a string τ unless a large portion of the web is crawled and indexed as done by search engines. Most of us do not have access to such crawled document collections from the web. Therefore, we exploit a web search engine and identify a small set of very relevant documents (or even just their snippets returned by a web search engine) containing the given candidate string τ. We rely on these small set of highly relevant documents to measure the correlation between τ and the target entity e.

Note that our criteria matching τ and e needs the web search results for τ to obtain a highly relevant set of documents or snippets containing τ. Hence, evaluating the similarity between a candidate string with entities in a reference table in general may not be applicable. Our approach here is to first identify a set of "synonyms" for each entity in the reference table. Once such synonyms are identified, our task of approximately matching a candidate string with a reference entity is now reduced to match *exactly* with synonym or original entity names, i.e., the (sub)set of tokens in the candidate string is equal to the token set of either a synonym or of an original entity name. Methods which only support exact match between candidate strings and entities in a reference table are significantly faster (e.g., [3]).

In this paper, we focus on a class of synonyms where each synonym for an entity e is an *identifying* set of tokens, which when mentioned contiguously (or within a small window) refer to e with high probability. We refer to these *identifying token sets* as IDTokenSets. We only consider *IDTokenSets* for an entity e which consist of a subset of the tokens in e for two reasons. First, the reference entity tables are often provided by authoritative sources; hence, each entity name generally does contain the most important tokens required to identify an entity exactly but may also contain redundant tokens which are not required for identifying the entity. Therefore, it is sufficient to isolate the identifying subset of tokens for each entity as an *IDTokenSet*. The *IDTokenSets* of an entity can be considered as keys that uniquely refer to the original entity. Second, our target applications are mainly entity extraction from documents. These documents are mainly drawn from the web such as blogs, forums, reviews, queries, *etc*, where it is often observed that users like to represent a possibly long entity name by a subset of identifying tokens (*e.g.*, 1-3 keywords).

The main technical challenge in identifying *IDTokenSets* for an entity e is that the number of all token subsets of e could be fairly large. For example, the entity "Canon EOS Digital Rebel XTI SLR Camera" has 127 subsets. Directly evaluating whether or not each subset τ_e of e matches with e would require a web search query to be issued. Therefore, the main challenge is to reduce the number of web search queries issued to identify *IDTokenSets* for an entity. Our main insight in addressing this challenge is that for most entities, if the set $\tau_e \subset e$ of tokens identify an entity e then a set τ'_e where $\tau_e \subset \tau'_e \subset e$ also identifies e (*i.e.*, *subset-superset monotonicity*). In other words, if $\tau_e \subset \tau'_e \subset e$, then τ'_e is more correlated to e. This is reminiscent of the "apriori" property in the frequent itemset mining [1, 13], where a superset is frequent only if its subsets are frequent.

We assume that the *subset-superset monotonicity* is true in general and develop techniques which significantly reduce the number of web search queries issued. For example, suppose "Canon XTI" identifies "Canon EOS Digital Rebel XTI SLR Camera" uniquely. Hence we assume that any superset say "Canon EOS XTI" also identifies e_1 uniquely. Therefore, if we efficiently determine the "border" of *IDTokenSets* whose supersets are all *IDTokenSets* and whose subsets are not, then we can often significantly reduce the number of web search queries per entity. In this paper, we develop efficient techniques to determine the border efficiently. We further extend these techniques for multiple entities by taking advantage of entities which are structurally similar.

In summary, our contributions in this paper are as follows.

1. We consider a new class of similarity functions between candidate strings and reference entities. These similarity functions are more accurate than previous string-based similarity functions because they aggregate evidence from multiple documents, and exploit web search engines in order to measure similarity.

2. We develop efficient algorithms for generating *IDTokenSets* of entities in a reference table.

3. We thoroughly evaluate our techniques on real datasets and demonstrate their accuracy and efficiency.

The remainder of the paper is organized as follows. We define problem in Section 2. We develop several efficient algorithms for generating *IDTokenSets* in Section 3, and discuss some extensions in Section 4. We present a case study that uses *IDTokenSets* for entity extraction in Section 5. In Section 6, we discuss the experimental results. In Section 7, we review the related work. Finally, we conclude in Section 8.

2. PROBLEM DEFINITION

We first define the notation used in the paper. Let $\mathcal{E}$ denote the set of entities in a reference table. For each $e \in \mathcal{E}$, let $Tok(e)$ denote the set of tokens in e. For simplicity, we use e to denote $Tok(e)$. We use the notation τ_e to denote a subset of tokens of e. That is, $\tau_e \subseteq e$.

Recall that we focus on identifying token sets which are subsets of the token set of the entity. That is, an *IDTokenSet* of an entity e consists of a subset τ_e of tokens in e. In the following, we formally define *IDTokenSets*. As discussed earlier in Section 1, to characterize an *IDTokenSet*, we rely on a set of documents and analyze *correlations* between the candidate subset τ_e and the target entity e. If a subset τ_e identifies e, then a large fraction, say θ, of documents mentioning τ_e is likely to contain the remaining tokens in $e - \tau_e$. We first define the notion of a document *mentioning* a token subset.