

Flexible query formulation for federated search*

Matthew Michelson, Sofus A. Macskassy, and Steven N. Minton

Fetch Technologies, Inc.
841 Apollo St., Suite 400
El Segundo, CA 90245

Abstract

One common framework for data integration in practice is federated search. Here an agent queries disjoint sources simultaneously, and then clusters the returned records in the absence of unique keys. However, formulating the correct queries to the sources can be challenging because of the possible query value variations. For instance, some sources may contain a first name as “John” while other sources use the name “Jonathan” for the same person. If the underlying sources do not support sophisticated matching then a single query of “John” will miss many records from the “Jonathan” sources. This paper presents an approach to formulating queries for federated search that leverages automatically discovered transformations such as synonyms and abbreviations. Our preliminary results demonstrate that indeed, transformations mined from a subset of sources will apply to a new, distinct source, thereby allowing query expansions based on the discovered transformations.

Introduction

One of the problems often tackled in information integration is providing a unified query interface to multiple, heterogeneous sources. In practice, one common implementation of this strategy is “federated search,” whereby an agent distributes simultaneous queries to the sources, and then clusters the aggregated resulting records (in the absence of a unique key). For instance, consider the case of federated restaurant search, as shown in Figure 1. In this figure, a single query with a name “Art’s” is sent simultaneously to three different restaurant search services. The resulting records are then clustered together, such that each cluster represents a different restaurant.

However, suppose the underlying sources do not support sophisticated entity matching, such that the sources return records based solely on matches of the tokens in the fields. For example, suppose that Fodors does not know that “Deli”

*This work was sponsored in part by the Air Force Office of Scientific Research under award number FA9550-09-C-0064. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government. Copyright © 2009, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

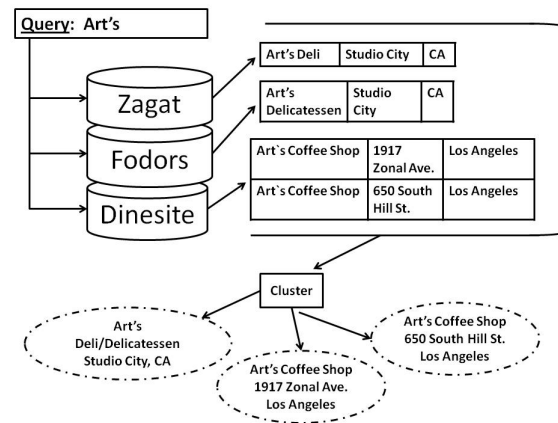


Figure 1: An example of federated search across three sources

is an abbreviation of “Delicatessen.” (This is the case in Figure 1). If the query is “Art’s Deli,” then Fodors will not return any matching records. Further, suppose that the sources support both query on the name and the city. If the query is “name = Art’s” and “city = Studio City,” then only the delis will be returned, since Dinesite is unaware that Studio City is part of Los Angeles. So, if we wanted to create as exhaustive a set of returned records as possible, to maximize our clustering utility, then we must create queries for both the specific neighborhoods and then the cities in which they reside and combine them with the various transformations on the restaurant name.

Transformations such as “Studio City” is “Los Angeles” and “Deli” is “Delicatessen” are called *heterogeneous transformations* (Michelson and Knoblock 2007) because they are not characterized by a single function (versus something such as an edit distance, which is). Heterogeneous transformations include things such as specificity/generality substitutions (Los Angeles is Studio City), nicknames/synonyms (Bob is Robert), acronyms, abbreviations, “random” transformations (a “4D” car is a “4 Door” car), and combinations of the various transformations (a “4D SUV” is a “4 Door Sport Utility Vehicle”).

Problems arise because many of the underlying sources

for federated search are unaware of the possible heterogeneous transformations that might apply across various sources (and hence to their own records) and therefore they can not be expected to transform the inputs to their own values. That is, Fodors can not be expected to know that Zagat defines the “Delicatessen” value as “Deli,” and therefore searches for Deli on Fodors should return the records with the value of Delicatessen. In fact, just creating lists of heterogeneous transformations to apply is difficult. First, enumerating the set of transformations is hard to develop, and second, there are no contracts between disjoint sources regarding what values they should support. Yet, by not considering such transformations, records can be lost in federated search.

In this paper, we present an application of mining such transformations directly from data to the problem of formulating queries for federated search. In particular, we first mine transformations automatically from a set of sources, following previous work (Michelson and Knoblock 2007). Then, we apply these discovered transformations to the problem of query formulation, generating an extended set of queries that when used in federated search yield a more exhaustive set of returned records for clustering.

Note that we mine the transformations from “in domain” sources. That is, for a domain such as restaurants, we collect a number of records from a few sources, mine the transformations that exist between them, and then apply these transformations to federated search within this domain.¹ By constraining the mining to within domain sources, mining the set of transformations becomes more tractable. Rather than having to consider the space of all possible transformations for all values to build a complete thesaurus (as search engines might), we are restricted to the subset defined for the domain. Further, and importantly, by mining *heterogeneous* transformations, we can extend the query formulation beyond the obvious transformation types, such as synonyms, to some of the more obscure transformations that may occur and are harder to develop manually.

The rest of this paper is organized as follows. The next section presents our approach to query formulation based on mined transformations. Then we describe our experiments to validate our approach. The next section discusses related research, and then we relate our conclusions and future research directions.

Formulating queries for federated search

Federated search is the problem of simultaneously querying multiple sources with a single query, and then processing the resulting records (e.g. aggregating, clustering, etc.).

More formally, we are given a set of sources $\{S_1, \dots, S_t\} \in S$, each of which contains record attributes $\{a_1, \dots, a_n\} \in S_i$, and a query Q , composed of search attributes $\{q_1, \dots, q_n\} \in Q$ (assume a value $\{\}$ for query attributes that are ignored). To answer the query the system

¹While in this study we only mine the transformations that occur between two sources, it is not a restriction and we can easily extend the approach to mine transformations between all pairs of sources.

returns a set of records $\{R_1, \dots, R_t\} \in R$ (one result set for each source), such that $R_i = \sigma_{a_j=q_j}(S_i)$ is the selection of records in S_i that match Q . We then define federated search as $f(\cup R_i) \rightarrow C$, where function f combines the returned records (e.g. clustering, aggregation, etc.).

The goal of this paper, then, is to extend Q , such that that for each $q_j \in Q$, we also consider the heterogeneous transformations $ht(q_j) \rightarrow \{q'_{j1}, \dots, q'_{jz}\}$. That is, we instead define $R_i = \sigma_{a_j \in \{q_j \cup ht(q_j)\}}(S_i)$.

To do this, we follow the method of our previous work that automatically discovers heterogeneous transformations between two data sources (Michelson and Knoblock 2007). That work follows three main steps. First, the algorithm finds candidate matching records across sources based on their token level similarity. Then, for those candidate matches, the algorithm looks at each aligned field (it assumes schema mapping is done) and records only the tokens in the fields that do not match exactly. So, in our “Art’s Deli” case, if the candidate match contains the fields “Art’s Deli” from Zagat and “Art’s Delicatessen” from Fodors, then the field value pair {“Deli,” “Delicatessen”} is recorded. As a last step, the algorithm prunes away all such value pairs that do not have a high enough mutual information.

In this paper, we start by mining the heterogeneous transformations from two of the sources in our federated search domain. Then, when we perform the actual federated search, we apply the most probable transformations for each query value, and generate a new set of queries that contains the permutations for the variations of the fields. Concretely, if our query is “Art’s Deli” and the system is aware of the transformation pair {Deli, Delicatessen}, then we create a second query “Art’s Delicatessen,” and we send out both queries and cluster the results. (Note that because we leverage the previous approach, we assume that the recorded transformations are in fact those that are the most probable since they have high mutual information by definition.) Figure 2 shows this process in the restaurant domain.

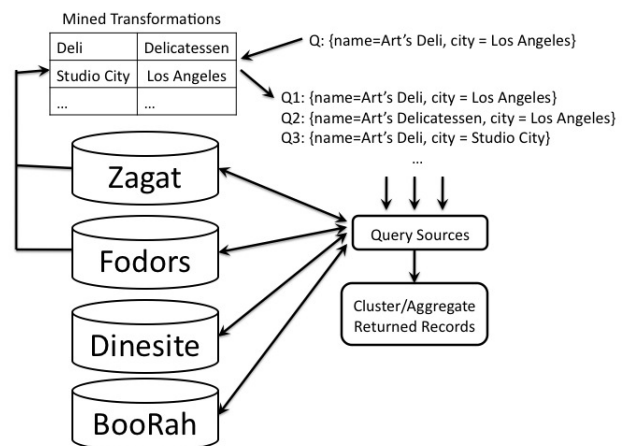


Figure 2: Using mined transformations for federated search

In order to create queries, as above, we need to apply the “within-term” transformations to each query value, and then combine the new query values to fit all possible query matches. So, we start by applying the within term transformations, substituting each token from one side of the transformation for the other, as in the “Art’s deli” case above. These substitutions will always occur for the transformation tokens. That is, if we see a new query with “Matt’s Deli,” then we apply the {Deli, Delicatessen} transformation again, to create the query term “Matt’s Delicatessen,” even if we have never seen Matt’s Delicatessen before.

After applying the within-term transformations, we create the set expanded queries by permuting the various attribute values that result from applying the within-term transformations. That is, Q expands into $Q' = \{(q_1 \cup ht(q_1)) \times \dots \times (q_n \cup ht(q_n))\}$. For example, given the query “name = Art’s Deli” and “city = Studio City,” and the known transformations {Deli, Delicatessen} and {Studio City, Los Angeles}, {El Segundo, Los Angeles}, we would create the six queries shown in Table 1.²

Table 1: A set of generated queries

	Name	City
Q1:	Art’s Deli	Los Angeles
Q2:	Art’s Delicatessen	Los Angeles
Q3:	Art’s Deli	Studio City
Q4:	Art’s Delicatessen	Studio City
Q4:	Art’s Deli	El Segundo
Q5:	Art’s Delicatessen	El Segundo

However, although considering the variations increases the possibility of returning more records, it also can dramatically increase the amount of processing needed since each query variant is sent to the set of sources. Yet, some of the queries will never return useful information for certain variations of the query values because these values do not exist in the underlying source. For instance, Fodors does not support the value Deli, because it uses Delicatessen instead. Therefore, sending queries with Deli to Fodors is wasteful.

To handle this optimization, we keep a history of query value hits for each source such that we can prune out queries to sources that have a history of returning no records. Leveraging the fact that most federated search sites support partial match queries, we take a query string and keep only the values that transform (along with their transformations). That is, for $Q = \{\text{name}=\text{“Art’s Deli,” city}=\text{“Studio City”}\}$ we first create the set of partial queries $Q'_1 = \{\text{name}=\text{“Deli”}\}$, $Q'_2 = \{\text{name}=\text{“Delicatessen”}\}$, $Q'_3 = \{\text{city}=\text{“Studio City”}\}$, and $Q'_4 = \{\text{city}=\text{“Los Angeles”}\}$. Since query matches could only be more constrained by adding conjuncts to the query, we know that if any of these partial queries fails to return a result, then no query that contains those values at that source would return a result.

Next, we send a probe query to each source for each of the partial queries that does not yet have a history record. For instance, we may have never had to use $Q'_3 = \{\text{city}=\text{“Studio City”}\}$ in a federated search before, so before we perform

the actual federated search, we send out a probe search to each source using Q'_3 . We record the results (essentially hits or no-hits), and add it to the history table.

As a last step, before we send any query (or transformed query) to an individual source in the federated search, we first check whether that query contains a value for which no results will be returned (according to the history table). If the history table determines that part of the query will return no hits, that query is discarded and not sent to that specific source (we assume only conjunctive queries). Note, however, that the history table has a decay set for the results of the probe queries. For instance, we can set a decay value of a month, such that after each month the values for probe queries are erased and probe queries must be resent to update the history table. In this manner, we can capture updates to the underlying sources, since pruning on based on probe queries could miss changes to the underlying attribute values at the sources.

In this manner we can exploit the set of possible transformations to yield more exhaustive results in federated search. We can also balance the increase in queries to the sources with the extra overhead of querying the source by pruning out all queries to sources that would be known to return no results.

Experiments

The main contribution of this paper is the use of automatically mined transformations to increase the record responses of sources in federated search. So, to examine our approach we select three sources within a domain, and mine the heterogeneous transformations between two of them. Then, we examine whether these mined transformations apply to the third source. If they do, that indicates the mined transformations would help discover records that would be missed otherwise.

Note, however, it is difficult to decide how well the transformations actually apply to the third source. To make this clear, consider our experimental set-up for the restaurant domain. In this case, we mine transformations using the Zagat and Fodors restaurant sources, and then examine the application of these mined transformations to the Dinesite restaurant source.

One of the transformations mined by the system is that “Asian” cuisine is “Japanese” cuisine. So, that means that if the system is querying Dinesite for “Japanese” restaurants, it should also query for “Asian” restaurants too, to maximize the returned results. However, given all of the restaurants in Dinesite with a cuisine “Asian,” how do we determine which of them are actually Japanese and which are Chinese (since the system also discovers that Asian food is Chinese food)? We only know that the cuisine value is Asian, and not its transformed value. In this sense, it is hard to calculate traditional precision and recall metrics for applying the mined transformations to the third source because we don’t know how to define if it is correct. So, we instead focus on examining how many times a transformation applies between records from the third source (Dinesite), and known matches between the third source and one of the sources used for

²We only include unique values, so Los Angeles is not repeated.

mining (Zagat, in this case). Here we are assuming that because the records match, the transformation is applied correctly (an assumption that is actually true for our datasets, but not always). Therefore, we can determine whether a transformation actually applies correctly.

While not as comprehensive as traditional precision and recall, looking at the applications of transformations across the matches yields a sampling of how well the transformations could aid in discovering new matching records for federated search. Further, we can see which transformations do not apply to the third source, which are exactly those that we would record in the history table to skip later. Therefore, we can simultaneously test the effectiveness of our caching effort on reducing the query load. In this early study we focus on two domains: restaurants, as stated, and cars.

To reiterate, for the restaurant domain, our three sources are Fodors, Zagat, and Dinesite. In particular, we collected 533 records from Fodors, 1,221 records from Zagat, and 1,721 records from Dinesite for this experiment. We mined the transformations between Fodors and Zagat, and then apply them to Dinesite. Since this is a preliminary study, we focus solely the cuisine attribute transformations.

Between the Dinesite and the Zagat records, there are 170 matches. We examined the cuisine values for these matches, and found that the mined transformations apply to 15.29% of the matches. Therefore, if we expanded the queries to Dinesite, using the cuisine transformations mined from Zagat/Fodors, we discover a more exhaustive set of records from Dinesite than if we did not expand the queries using the cuisine transformations.

However, we also note that not all of the transformations that are mined actually applied. Of the 26 mined transformations, only seven transformations actually apply, as shown in Table 2. Therefore, although only 26.9% of the mined transformations actually apply, those that did apply to roughly 15% of the matches. Here is an example where the caching works well. It would eliminate almost 75% of the transformations from contributing to future queries because they do not apply to the source.

Table 2: Restaurant transformations that apply

Transformation		Occurrences
American	Steakhouses	13
American	Californian	8
French	New Pacific Wave	2
Southern	Southern/Soul	2
Asian	Japanese	1
Continental	French Bistro	1
Caribbean	Eclectic	1

We also ran a similar experiment in the cars domain. For this domain our sources include 3,171 records from the Edmunds car information website, 2,777 records from the Kelly Blue Book (KBB) car information site, and 2,238 records from the Craigslist classified site. Each record from the Craigslist site describes a used car for sale, and was manually segmented into the various attributes such as make, model, and trim.

For this domain, we learn the transformations between the Edmunds and Kelly Blue Book sources, and then apply those transformations to the parsed Craigslist records. This experiment focuses on the trim attribute because the mined heterogeneous transformations are not linguistic in the sense that they would be included in a standard thesaurus. The discovery and use of these less linguistic transformations is an important difference between our work and other work on query formulation that uses standard thesauri and ontologies such as WordNet (see Related Work). For example, the system mined trim transformations such as “2D Coupe” is “2 Dr Hatchback,” and “4D Sport Utility” is a “4 Dr SUV.”

As before, we examine the number of times a transformation mined from Edmunds/KBB applies to matches between Edmunds and the Craigslist data. We found 1,568 matches between the Edmunds records and the Craigslist records, and of these matches, the mined transformations apply to 7.08% of the Craigslist records. In this case, the system mined 15 transformations, of which eight returned values, as shown in Table 3. The first two transformations in the Table seem noisy, but in fact, the first transformation only applies to 75 of the 1,568 matches, so it does provide utility. Again, also, we see that caching would help eliminate almost 50% of the spurious transformations.

Table 3: Cars transformations that apply

Transformation		Occurrences
4D	4 Dr	75
2D	2 Dr	14
4D	4 Dr STD	8
4D Sport Utility	4 Dr SUV	6
4D	4 Dr V6	3
2D Coupe	2 Dr Hatchback	2
4D Sport Utility	4 Dr 4WD SUV	2
2D Sport Coupe	2 Dr Hatchback	1

The goal of these experiments is to determine whether transformations can be mined from a set of sources and then applied to another in the context of federated search. Our experiments demonstrate that this is the case, and indeed for the known cases where the transformations could apply, they do so for 11.14% of the records, on average, across the two domains. Since this is a preliminary study we focus only on a single attribute, but we feel that these experiments justify our approach to generating more useful queries for federated search.

Related Work

Federated search has been a popular research problem in computer science for quite a while (e.g. (Czejdo, Rusinkiewicz, and Embley 1987)). In general, the more popular problems involve figuring out the set of sources and which to select from for answering a query ((Si and Callan 2005; Callan 2000; Gravano et al. 1997; Nottelmann and Fuhr 2003)).

However, there has also been work on correctly building the queries for the sources (referred to as query formulation and sometimes query translation). Many of these

“query translations” are based on ontological knowledge or thesauri (Kerschberg et al. 2004; Fuhr et al. 2002; Graupmann, Biwer, and Zimmer 2003). So, the goal is generally to exploit some linguistic information about the query terms as transformations, such as substituting synonyms or more general concepts.

In our work, because we are mining arbitrary heterogeneous transformations we can support query translations that go beyond the scope traditional linguistic transformations. For instance, we can support transformations such as “4D SUV” is “4 Door Sport Utility Vehicle.” This transformation involves an acronym (“SUV” is “Sport Utility Vehicle”), a “substitution” (4D is 4 Door), and their combination. So the scope of transformations does not have to be “linguistic” in the traditional sense. Further, much of the previous work exploits previously constructed ontologies and thesauri such as WordNet. In our work, we explicitly build up the set of transformations to use for the expansions as part of the overall process.

Conclusions

In this paper we present a method to exploit automatically mined transformations for query formulation in federated search. Our experiments show that by exploiting specific transformations, the recall of federated search results increases. We also describe how to deal with the issue of sending queries that historically return no results, which is inefficient. By mining heterogeneous transformations we support transformation types for query formulation that go beyond traditional linguistic transformations.

In the future, we will investigate how to learn the specific transformations for each source. That is, rather than discovering that some source does not support some transformation values, it is preferable to learn how the sources support transformations probabilistically. We might learn that certain combinations of transformations are supported far less than others, and we could plan the query distribution process using this knowledge.

References

- Callan, J. 2000. *Distributed Information Retrieval*. Kluwer Academic Publishers.
- Czejdo, B. D.; Rusinkiewicz, M.; and Embley, D. W. 1987. An approach to schema integration and query formulation in federated database systems. In *Proceedings of the Third International Conference on Data Engineering*, 477–484. Washington, DC, USA: IEEE Computer Society.
- Fuhr, N.; Klas, C.-P.; Schaefer, A.; and Mutschke, P. 2002. Daffodil: An integrated desktop for supporting high-level search activities in federated digital libraries. In *Research and Advanced Technology for Digital Libraries. 6th European Conference, ECDL 2002*, 597–612. Springer.
- Graupmann, J.; Biwer, M.; and Zimmer, P. 2003. Towards federated search based on web services. In Weikum, G.; Schöning, H.; and Rahm, E., eds., *Datenbanksysteme für Business, Technologie und Web (BTW) : 10. GI-Fachtagung*, volume P-26 of *Lecture Notes in Informatics*, 384–393. Leipzig, Germany: Bonner Kllen.

Gravano, L.; Chang, C.; Garcia-Molina, H.; and Paepcke, A. 1997. Starts: Stanford proposal for internet meta-searching. In *Proceedings of SIGMOD*.

Kerschberg, L.; Chowdhury, M.; Damiano, A.; Jeong, H.; Mitchell, S.; Si, J.; and Smith, S. 2004. Knowledge sifter: ontology-driven search over heterogeneous databases. *Scientific and Statistical Database Management, International Conference on* 0:431.

Michelson, M., and Knoblock, C. A. 2007. Mining heterogeneous transformations for record linkage. In *Proceedings of the 6th International Workshop on Information Integration on the Web (IIWeb)*, 68–73. AAAI Press.

Nottelmann, H., and Fuhr, N. 2003. Evaluating different methods of estimating retrieval quality for resource selection. In *Proceedings of SIGIR*.

Si, L., and Callan, J. 2005. Modeling search engine effectiveness for federated search. In *Proceedings of ACM SIGIR*, 83–90. New York, NY, USA: ACM.