Semantic Negotiation: Co-identifying objects across data sources

R. Guha
IBM Research, Almaden
rguha@us.ibm.com

ABSTRACT

Integrating and composing web services from different providers requires a solution for the problem of different providers using different names for the same object. We argue that while URIs might be adequate for creating shared namespaces for classes and property types, the practical difficulties associated with everyone using the same URI for individual objects makes exclusive dependence on this approach untenable. We first consider the use of shared keys to solve this matching problem and discuss some of the problems associated with this approach. We introduce the concept of DD, a generalization of keys and discuss some conditions under which common DDs guarantee correct matches. We then propose a probabilistic approach to matching and consider, with empirical validation, approximations as a solution to the problem of requiring a combinatorially large number of probabilities. Finally, we introduce Semantic Negotiation, a process by which two agents can negotiate a mutually comprehensible reference for an object.

1. BACKGROUND

The ease with which web sites could link to each other doubtless contributed to the rapid adoption of the Web. It is hoped that as Web Services become more prevalent, programs will be able to similarly weave together disparate Web Services. It is becoming clear that just agreeing on common protocols such as SOAP is not enough to enable such compositions. Not just the packing of messages, but also their content needs to be mutually comprehensible, i.e., there needs to agreement on the semantics of the terms used in the messages. As providers independently start providing services, it is inevitable that they will use different different names/URIs for the same concepts and individual objects, which will make it difficult to compose and integrate these services.

Example: Consider two services, one from CDDB for listing a music artists albums and their songs (and vice versa), and the other from an online store such as Amazon. Now imagine composing these two services so that, given a particular song, we identify the album(s) on which the song appears (using CDDB) and order the album from Amazon. The biggest problem in doing this simple composition is caused by the fact that each each of these services uses its own set of global identifiers for each musician and album. Consequently, we cannot just take the CDDB identifier for album and pass it to Amazon, but instead have to map the identifier across the two namespaces. In practice, the effort and expense of this mapping overshadows the benefits of the automated service composition.

It is expected that standardized ontologies, which make everyone use the same name for each object, will solve this problem. We believe this will be hard to achieve. When there is a clear ownership of each object by one of the providers, as with domain names or web pages, or there is a relatively small number of names that everyone has to agree upon, such as the names of the HTML tags, this approach works well. However, when this condition does not hold (as in the above example), this approach has met with less success. In particular, we don't expect common names for individual objects (people, places, products, ...). Consequently, programs will need to map the different names used by different providers onto each other. Given the large number of individual objects (compared to schema objects such property types and classes), manual mapping is not practical. In this paper, we discuss the pros and cons of the traditional approach to solving this problem using keys and propose Semantic Negotiation, a mechanism for dynamically negotiating mutually understandable references. We present and discuss both deterministic and probabilistic versions of Semantic Negotiation.

2. RELATED WORK

The problem of matching objects across different data sources has been studied, quasi-independently in at least three fields.

To our knowledge, this was first studied as the "Record Linkage" problem ([4] [8] [5]). Much of that work focuses on statistical methods for determining when the values of the same field in different records are syntactically different, but should actually be the same, e.g., to recognize that the phone numbers "634-988-2900' and "(634) 988 2900" are really the same. In our work, we assume the use of these techniques or some level of canonicalization so that this is not a problem.

There has been much work in the field of databases on the problem of data integration, most of which has focussed on the schema level. Only recently have researchers [9] [11] started paying attention to the problem of mapping individual objects[1].

Recently, researchers have applied Probabilistic Relational Models to this problem [14], applying these techniques to problems such as removing duplicates in citation databases. Our work on Probabilistic Semantic Negotiation has been inspired by this work.

[1]For the remainder of this paper, we will use the term object to refer to individual objects

3. SHARED KEYS

A pre-determined *shared key* can be used to match objects across different databases. So, in the above example, a set of attributes of musicians and albums that uniquely identifies every musician and album (i.e., no two musicians/albums share the same values for that set of attributes) can be used. The software for composing the two services would obtain the values for these attributes from CDDB, then do a search on Amazon on these attributes to obtain the musician/album's identity on Amazon and then use this to place the order.

From a logical standpoint, consider the key κ consisting of the attributes p_1, p_2, ..., p_n. To say that κ is a shared key between the providers A and B is to say the following:

$$\begin{aligned}&\forall(x\epsilon A, y\epsilon B)^2(\exists(a_1, a_2, ..., a_n)\\ .\quad &(p_1(x, a_1) \wedge ...p_n(x, a_n)) \wedge (p_1(y, a_1) \wedge ...p_n(y, a_n)))\\ .\quad &\qquad\quad \Longrightarrow (x = y)\end{aligned}$$

This definition of shared keys says that if there exists a set of values $(a_1, a_2, ..., a_n)$ that an object x at provider A and object y at provider B have for the attributes $(p_1, p_2, ..., p_n)$, then the two objects are the same.

3.1 Common Key $\neq$ Shared Key

κ being a shared key between A and B is stronger than κ just being a common key, i.e., a key for A and a key for B. κ being a common key only says that κ picks out a unique object in each of the two providers. Saying that κ is a shared key asserts that these unique objects identified at the two providers are the same. To see the difference, consider the following example. Consider two databases about cities, one about cities in Texas and the other about cities in France. The name of the city can uniquely identify the city in each database, i.e., $name$ is a key. However, it is not a shared key. The cities with the name 'Paris' identify different cities in the two databases, Paris, France and Paris, Texas. So, if κ is to be a shared key, it has to be a key over the domain $(A \cup B)$ and not just independently over the domains A and B. As illustrated by the above example, a set of attributes that is a key over the domain A and a key over the domain B need not be a key over the domain $A \cup B$.

3.2 Problems with Shared Keys

There are two problems associated with using shared keys.

- Though most objects in the domain require only a small number of attributes to identify, often, a few objects require a substantially larger number of attributes to identify. Since the same set of attributes have to work with all the objects in the domain, they are determined by these extreme cases and tend to be very large. For example, since there are multiple musicians named 'Eva', some of whom have albums named 'Eva', we need to include other attributes (such as the album label or list of songs) to distinguish between these. In another example, since the album 'Dark Side of the Moon' by 'Pink Floyd' was first released in 1974 and re-mastered and re-released after 25 years, the keys will have to include the release date as well. It is easy to see how even in the case of musicians and music albums, where there is a strong incentive to pick names that are distinctive, we end up requiring a large keys, i.e., keys with many attributes.

  Large keys are undesirable. For a key to be useful, both services need to have all the attributes in the key. So, larger keys tend to be less useful since there is a lesser likelihood of both services having all the attributes in the key. Further, there are often substantial variations in the spellings, capitalization, punctuation, etc. between different providers. Heuristic matchings [3] solve some of these problems, but also introduce false positives. The likelihood of wrong or missed matches increases with the size of the key.

- Requiring pre-identified keys between every pair of services that we might need to integrate makes it very hard to compose and integrate new services on the fly. This problem may be partially overcome by each provider publishing valid local keys of each type of object. However, as mentioned earlier, shared keys are stronger than common keys and assuming that a key that is used by two providers is a shared key could lead to mistakes.

[2] For the sake of brevity, we are using the symbols A and B to denote both the data sources and the domains of the data sources.

In the next two sections, we introduce the concept of Discriminant Descriptions and Semantic Negotiation to overcome the first problem. In the following section, we introduce the concept of probabilistic matching as a potential strategy to overcome the second problem.

4. DISCRIMINANT DESCRIPTIONS (DD)

The problem of large keys arises because of the requirement for a single set of attributes to work for all the objects in the domain. Even if most objects can be uniquely identified by a small number of attributes, the extreme cases may force the inclusion of a number of additional attributes. So, our first step is to introduce the concept of a Discriminant Description (DD). A DD φ for an object O is a formula which only it satisfies, i.e., discriminates it from all other objects.

If κ is a key for A, then we have,

$$\forall(x\epsilon A, y\epsilon A) \quad (\kappa(x) \wedge \kappa(y) \Rightarrow (x = y)) \qquad (1)$$

In contrast, if φ is a DD for O, then we have,

$$\forall(x\epsilon A)\varphi(x) \Rightarrow (x = O) \qquad (2)$$

These formulas show us that keys are much stronger constraints than DDs. Each DD applies at the instance level, for one particular object O. Keys on the other hand, apply to all objects in the domain of the provider.

Continuing with our music examples, most classical composers can be uniquely identified by their last names. So, Pytor Ilyich Tchaikovsky has the DD $lastName(x, `Tchaikovsky')$[3] However, since there are multiple composers with the last name 'Bach', $lastName$ is not a key. Interestingly, since the names of most musicians and music albums is unique, the average length of DDs for musicians and music albums in FreeDB (a free version of CDDB) is just a little over 1. Indeed, in many domains, we hope to find the average size of DDs to be much smaller than that of keys.

An object which has a DD in a database will likely have many DDs. If $\varphi(O)$ is a DD for O, $\varphi(O) \wedge \beta(O)$, where $\beta(O)$ is true is also a DD for O. We are often interested in *Minimal DDs*. φ is a Minimal

[3] Since DDs are formulas with exactly one free variable, a more precise syntax would be $(\lambda(x)lastName(x, 'Tchaikovsky'))$. For the sake a brevity, we will drop the λ.

DD iff no subset of φ is also a DD, assuming that DD is purely conjunctive.

We can use DDs to match object across service providers. If a description is discriminant for the domain $(A \cup B)$, and picks out an object each from A and B, then these two objects are the same. As with the problem of going from common keys to shared keys, we have to solve the problem of going from knowing that a description is discriminant in A and in B to the description being discriminant in $(A \cup B)$. Though we might have a common DD and a pair of objects, one on each side, for which the description is discriminant, we cannot assume that these objects are the same. The example of Paris, Texas vs Paris, France, where we illustrated the difference between common keys and shared keys applies here as well. Logically speaking, the common DD φ tells us that,

$$(\forall(x\epsilon A)\varphi(x) \Leftrightarrow (x = O_A)) \;\wedge\; (\forall(x\epsilon B)\varphi(x) \Leftrightarrow (x = O_B)) \tag{3}$$

From this, we wish to conclude $(O_A = O_B)$. We can do this if one or more of a certain set of additional assumptions are satisfied. Some examples of such assumptions, without proofs, are given below:

- All of the p_i in φ are one-to-one relations.
- We are apriori given that
  - the object occurs in both A and B
  - A and B are both complete in each of the p_i occurring in φ, i.e., for each object in A and B, the providers have all the values for each of the p_i.

These are similar to conditions that need to be satisfied for a common key to be used as a shared key. In many cases, such as when one or both of the providers is known to have comprehensive information about the domain, it may be reasonable to make these assumptions. In other cases, when little can be assumed about the providers, these assumptions might not be reasonable.

There are a number of applications where we don't need a hard guarantee that the match is always correct. For example, in many applications, the result of the computations are presented to a user who can easily identify bad matches. For such applications, in the case where we cannot make the kind of assumptions described above, it would be useful to have a framework for identifying matches that while not guaranteed, have a high likelihood of being correct. We present such a framework in a later section.

5. PROBABILISTIC MATCHING

O_A and O_B satisfying a DD φ at A and B respectively, increases the likelihood that they are the same object. So, the problem of Semantic Negotiation can be cast in terms of probabilities as follows. We would like to find a common DD φ conditioned on which the probability that O_A and O_B are the same is sufficiently close to 1, i.e.,

$$P(O_A = O_B|(\varphi(O_A) \wedge \varphi(O_B))) = 1 - \delta \tag{4}$$

This formulation of the matching problem, though less demanding of prior guarantees such as the assumptions listed in the earlier section, requires more prior information in the form of conditional probabilities. In fact, since DDs by their very nature correspond to unique configurations, obtaining exact values for these conditional probabilities could be very difficult, if not impossible. To overcome this problem, we develop approximations that can be empirically verified. In the next section, we describe work, that is currently underway, which is aimed at developing and validating approximations.

Before we get into approximations, we first review the basics.

We are given two sets, A and B, and we know that exactly one object in each of the two sets satisfies φ. We would like to estimate the probability that these two objects, O_A and O_B are the same. $O_A = O_B$ if and only if $O_B \epsilon A \cap B$. If the two sets are disjoint, $O_A \neq O_B$. Conversely, if B is a subset of A (assuming, without loss of generality, that B is smaller than A), $O_A = O_B$. So, the probability of $O_A = O_B$ is a function of the overlap between A and B. Alternately, $P(O_A = O_B)$ is a function of the probability of an element of B also being an element of A.

For example, A and B might be travel sites. If φ is $name(x, `Paris')$, we would like to estimate the probability of the object on each site called 'Paris' being the same. For our model to be realistic, this probability has to account for the fact that Paris, France occurs in more travel sites than Paris, Texas. In our earlier example of cities in France and cities in Texas, it should be able to incorporate knowledge about the two domains being disjoint. In another example, A and B might correspond to the musicians whose albums are sold by two different web sites. Again, our model has to accommodate the fact that music sites are more likely to sell more popular musicians.

Let the probability of an element of B also being an element of A be α. Let the probability of a randomly chosen element of $A \cup B$ satisfying φ be ρ. We compute $P(O_A = O_B)$ for a given i, where $i = |A \cap B|$ and then sum over i, factoring in the probability of each i and the knowledge that we have only one or two objects satisfying φ in $A \cup B$. So $P(O_A = O_B)$ (the Co-Identification Probability) is given by:

$$\frac{\sum_{i=1}^{B} \binom{B}{i} \alpha^i (1-\alpha)^{B-i} i\rho(1-\rho)^{A+B-i-1}}{(\rho(1-\rho)^{A+B-i-1}) + (\rho^2(1-\rho)^{A+B-i-2})} \tag{5}$$

We can expect to know the cardinalities of A and B. If A and B are random or Bernoulli samples from some underlying set (as is the case in our empirical evaluations), we can easily estimate α. Otherwise, if the elements of A and B have been selected according to some other criterion, we need to know α. Finally, we need the probability ρ corresponding to φ. Estimating the probability ρ of an element satisfying an arbitrary formula φ is much more difficult. In the next section, we outline some different approaches to computing (approximations to) ρ.

5.1 Approximations

Given an formula φ, we would like to determine the probability ρ that a randomly chosen object satisfies it. Since the space of potential formulae is very large, we would like a compositional approach, wherein the probability corresponding to a formula φ is computed from sub-formulae of φ. For now, we restrict our attention to flat formulae, i.e., formulae which have the structure $p_1(x, a_1) \wedge p_2(x, a_2) ... p_n(x, a_n)$, where $a_1, a_2 ... a_n$ are constants.

Our goal is to be able to compute approximations to the probability at a very low cost, while retaining the ability to get progressively more accurate for subsets of the domain where the approximation either fails or where we cannot afford to be inaccurate. We do this as follows.

We first compute approximations to the probabilities associated with each of the $p_i(x, a_i)$. We call these *Atomic Probabilities*. Then, building upon these, we compute approximations of the joint probability.

5.2 Atomic Probabilities

For example, if p_i is the attribute $name$, we would like to know the probability that a randomly chosen city has the name 'Paris'.

There is a very large amount of work in the fields of statistical analysis [18] and data management ([2], [13]) on how one can compute P($p_i(x, a_i)$) to varying levels of accuracy. All of those methods are applicable here. We give a brief overview of the approaches tried by us in our empirical studies.

The approach is a function of the set of possible values for a_i. We have restricted our attention to the case where a_i is one of a discrete set of values.

If we have access to a comprehensive database of values of this attribute (for the class of objects under consideration), we can compute exact values of these probabilities for each a_i. So, for example, in the case of places, if we had a gazette which listed all the cities along with their names, we could compute an exact value of this probability. This will tell us that there is only one city called 'Los Altos', but that there are 8 cities called Springfield.

In many cases, we might not have access to this data or the set of a_i may be unconstrained. For example, if we were looking at people (instead of cities), we cannot expect to get a comprehensive database of all people or even a comprehensive list of all first and last names. In such cases, assuming we have a sufficiently large (but not comprehensive) database which is adequately representative of the domain, based on an appropriate set of samples, we can approximate this probability to be the same for all a_i.

Such estimates can be augmented with data about particular attribute values. So, for example, if the domain is US residents, we can use information from the Census Bureau [1] about the most common first and last names to provide exact values for these common attribute values and use the approximation for names that don't appear in this list.

This approach of using approximations augmented with exact data for particular attribute values is especially suited for domains when the attribute values exhibit Zipf distributions [19].

We expect the emergence of web services that provide these probabilities for a wide range of core vocabulary items (i.e., for common classes and property types). Such data, available as a service, would be analogous to the role played by actuarial services in the insurance industry.

5.3 Joint Probabilities

Given the atomic probability associated with each $p_i(x, a_i)$, we next compute the joint probability associated with the conjunct $p_1(x, a_1) \wedge p_2(x, a_2) ... p_n(x, a_n)$. Unfortunately, the combinatorially large number of possible formulae makes it virtually impossible to compute exact joints. We consider the following 2 approximations.

Independence: We can assume independence between the different $p_i(x, a_i)$. So, the probability associated with $p_i(x, a_i) \wedge p_j(x, a_j)$ would be $P_1 * P_2$, where P_1 and P_2 are the probabilities associated with each of the conjuncts.

Partial Determination: Instead of computing an exact joint or assuming complete independence for each a_i and a_j for $p_i(x, a_i) \wedge p_j(x, a_j)$, we can approximate this to dependency between p_i and p_j. So, the probability associated with $p_i(x, a_i) \wedge p_j(x, a_j)$ would be $(P_1 * P_2 + \theta(p_i, p_j))$, where $\theta(p_i, p_j)$ is a measure of the dependence between p_i and p_j.

This is a probabilistic variant of the concept of Determinations which was introduced in [17] to formalize analogical reasoning and later by [12] and [10].

The utility of probabilistic matching is a function of how far we can get with simple approximations. Though the approximations described here are very crude and simplistic, it would be good if they enabled us to match objects with a reasonable level of reliability. Of course, the applicability of any given set of approximations is a function of the domain and the particular attributes used in the bootstrap. We expect experimentation and validation of approximations for different domains and attributes. In the next section we describe an empirical evaluation of these approximations for two domains, both to show that there are at least some important domains where these simple approximations work, and to propose a framework for performing such evaluations.

6. EMPIRICAL EVALUATIONS

In this section we describe an empirical evaluation which shows that simple approximations are adequate at least for the domains of people and places. Our evaluations are based on the following two data sets:

- A list of 24,347 cities across the world, obtained by aggregating data from the US Census Bureau, TAP and the web site CityMayors.org. For each city, we know its name and the country. For some of them, we also know the province/state it occurs in. Duplicates in this set were manually removed.
- A list of 172,797 employees of IBM in the US. For each employee, we have the first and last names, position, work location and division. Since the goal is to do object matching, we have not used unique identifiers such as employee serial numbers. These fields were used for checking the correctness of the predictions.

To test the validity of the approximations, we compared the performance of probabilistic matching with the probability predicted by our approximation. We computed the probability of each $p_i(x, a_i)$ using the method described earlier. In all cases, except for the names of people and cities, we estimated the probability from the data sets. For the case of names of people, the probabilities were generated using the sampling method with a special case for commonly occurring names, which was obtained from the census bureau. The joint probabilities were computed assuming independence.

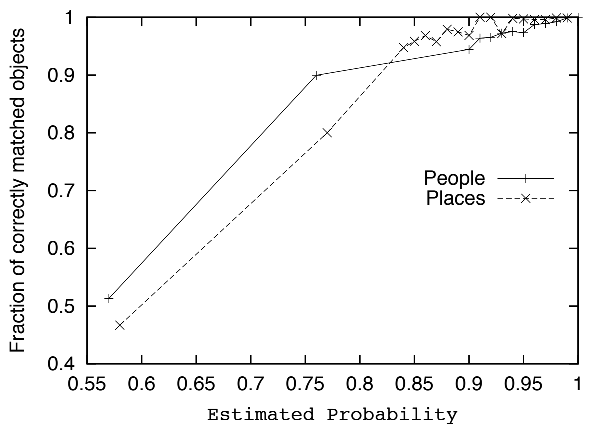

Figure 1: Fraction of Correct Matches vs Identifying Prob.

From each data set we created two subsets of elements chosen as follows. Each set was generated by walking through the data set, generating a random number for each item and including the item in the set if the generated number was greater than 0.66 (so, the two sets are Bernoulli samples). For each item in the smaller set, for each DD, we tried to find a unique object in the other set satisfying that DD. For each case where we found a shared DD, we calculated the probability associated with that DD.

Overall, 99.7 percent of matches in the case of people and 99 percent of matches in the case of places were correct. This includes matches over all available DDs, not just the minimal ones. Using just minimal DDs, these numbers were 97.7 percent for people and 97.9 percent for places. Figure 1 shows the graph of the predicted probability vs. the actual fraction of correct matches with a predicted probability in that 5% range. More details are available in [6]. From this, we conclude that at least for these two kinds of objects and these sets of attributes, these approximations yield good results.

7. SEMANTIC NEGOTIATION

One advantage with keys is that they apply to entire databases. On the other hand each DD applies only to a single object. Further each object may have multiple DDs. Hence, it is not reasonable to expect a site to advertise the DDs for each of its objects. Semantic Negotiation is the process of dynamically discovering and negotiating common DDs.

There are many potential negotiation strategies that can be followed, depending on whether the goal is to maximize the likelihood of finding the match, to maximize the certainty that the match is correct, to minimize the amount of information revealed or to minimize the number of exchanges. If the goal is to minimize the data revealed by one agent to another, then the use of minimal DDs is preferred. If the goal is to maximize the likelihood of a match being correct, then the use of the DD with the highest Co-Identification Probability is preferred. In practice, some tradeoff between the two extremes will have to be made.

A good example of a protocol for Semantic Negotiation is the one used by the TAP [15] system in the context of GetData, a web service interface provided by TAP for accessing one or more attributes of an object. Using GetData, a client program, which has no prior agreement (regarding names for individual objects) with the service provider, can obtain the values of various attributes of an object from that provider. One of the difficulties that needs to be solved is for the client and provider to create a mutually comprehensible reference to the object. In this context, the negotiation proceeds as follows. It is assumed that the client and provider share a common vocabulary of attributes (p_1, p_2, ... p_n) and attribute values (a_1, a_2, ... a_n).

1. the client sends the server a GetData query, using the description to refer to the object whose property is being accessed. The description is assumed to be discriminant on the client.

2. if the server does not understand the description, i.e., it uses terms that server does not know about, the server responds with an error code indicating that the description was not understood. In this case, it also lists the particular terms not understood and if the description included the class of the object, and the class was understood, it might include some of the properties of that class it does know about. Based on this feedback, the client can try to provide a description that the server is more likely to understand.

3. if the server understands the description but there are no objects matching that description, it returns an error code saying so. It can optionally also tell the client which fragment of the description was not satisfied by any of its objects.

4. if the server understands the description but there are multiple objects matching the description, it returns an error code saying so. In this case, depending on how many different objects match, the server may return a list of these, along with descriptions that are discriminant on the server. The client may choose one amongst these and retry the query.

5. if the server understands the description and there is a single object matching the description, it returns the values that were requested. In the case where the answer is a list of objects, the answer may include additional data about each object, which the client may cache, in anticipation of future queries about these objects. This is just a form of proactive caching. Optionally, this additional data may also include the server's names for these objects so as to reduce the need for negotiation for the names of these resources.

The process described above allows a client to dynamically negotiate references with servers based on their sharing a core vocabulary. Experience with TAP and GetData ([16], [7]) shows that this approach works very well.

8. CONCLUSIONS

Integrating and composing web services from different providers requires a solution for the problem of different providers using different names for the same object. In this paper, we considered the use of shared keys and the problems associated with that approach. We introduced the concept of Discriminant Descriptions that solves the problem of long keys and a probabilistic framework for matching, with approximations which relaxes the assumptions that are required to go from common to shared keys. We finally described Semantic Negotiation, a process by which a client and a service can negotiate mutually shared references.

9. REFERENCES

[1] www.census.gov.

[2] R. Agrawal and A. Swami. A one-pass space-efficient algorithm for finding quantiles. In S. Chaudhuri, A. Deshpande, and R. Krishnamurthy, editors, *Proc. 7th Int. Conf. Management of Data, COMAD*, 28–30 1995.

[3] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23.

[4] M. Cochinwala, S. Dalal, A. Elmagarmid, and V. Verykios. Record matching: Past, present and future. citeseer.nj.nec.com/588173.html.

[5] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 1969.

[6] R. Guha. Object co-identification on the semantic web. tap.stanford.edu/CoIdent.pdf.

[7] R. Guha, R. McCool, and E. Miller. Semantic search. In *Proc. WWW 13, Budapest, Hungary*, 2003.

[8] S. J. A. H. B. Newcombe, J. M. Kennedy and A. P. James. Automatic linkage of vital records. *Science*, 130:954 – 959, 1959.

[9] M. A. Hernandez, R. J. Miller, and L. M. Haas. Clio: A semi-automatic tool for schema mapping. In *ACM SIGMOD*, 2001.

[10] S. Kramer and B. Pfahringer. Efficient search for strong partial determinations. In *Knowledge Discovery and Data Mining*, pages 371–374, 1996.

[11] N. K. L. Gravano, P. Ipeirotis and D. Srivastava. Text joins for data cleansing and integration in an rdbms. In *ICDE*, 2003.

[12] P. Langley. Induction of condensed determinations. In *Knowledge Discovery and Data Mining*, pages 327–330, 1996.

[13] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. pages 426–435, 1998.

[14] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. www.cs.berkeley.edu/ milch/papers/nipsnewer.ps.

[15] R.Guha and R. McCool. Tap: Towards a web of data. http://tap.stanford.edu/.

[16] R.Guha and R. McCool. Tap: A semantic web platform. *Computer Networks*, 42:557 – 577, 2003.

[17] S. Russell. *The Use of Knowledge in Analogy and Induction*. Pitman, 1989.

[18] C. Sarndal, B. Swensson, and J. Wretman. *Model assisted survey sampling*. Springer-Verlag, 1992.

[19] G. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.