# Inducing Relatedness Graphs for Data Integration

Jeremy Engle[#1], Ying Feng[#2], Rob Goldstone[#3]

[#]*Indiana University*
*Bloomington, Indiana, USA*
[1]jtengle@indiana.edu
[2]*yingfeng@indiana.edu*
[3]rgoldsto@indiana.edu

*Abstract*— **In this paper, we present the AbsMatcher system for schema matching which uses a graph based approach. AbsMatcher creates a graph of related attributes within a schema, mines similarity between attributes in different schemas, and then combines all information using the ABSURDIST graph matching algorithm. The focus of this paper is on methods for generating relationships which are semantic in nature, but only require a simple data model. These relationships sources provide a baseline to be used when no others are available. Simulations demonstrate how the use of automatically mined graphs of within-schema relationships, when combined with cross-schema pair-wise similarity, can result in matching accuracy not attainable by either source of information on its own.**

## I. Introduction

Data integration has application to a wide variety of fields from e-commerce to bioinformatics. This paper explores how the results of web queries and other mined information can be used as a baseline in building relatedness graphs and using them for graph-based schema matching. The AbsMatcher framework uses a two stage process, by first mining different forms of information and then applying the ABSURDIST algorithm which determines match correspondences between attributes in each system. The contribution of AbsMatcher is in the modules used to mine different forms of relationships to create graphs and the integration of those graphs with the ABSURDIST algorithm. These modules present new sources of information that allow a graph-based approach to be applicable beyond the limits of relational databases or XML as seen with previous graph based approaches.

Problems within schema matching are often described as addressing entity matching, attribute matching, or value matching. This paper concentrates on the attribute matching problem where the goal is to find correspondences between attributes in a source and target schemas. We concentrate on one-to-one matches as an initial effort, leaving complex n-to-one [8] matches to future work.

The primary challenge in graph-based matching systems is how to generate edges for a graph. Previous systems [2, 10, 15, 17] avoid this problem by only accepting data models that have an intuitive translation to graph form. Two common examples of this are creating a tree-like graph from XML data and using the metadata of a relational database to create graphs. When all the relationships generated for a graph a specific to a single data model matching is limited to only data sets that share the same data model. Additionally, in these scenarios the generated relationships are often based on the design of a data set. The usefulness of these relationships is then dependent on consistency among all database designers, which is unlikely. An example of this is a relationship for attributes in XML that are nested together. AbsMatcher instead attempts to build ubiquitous graphs by basing relationships on semantics of the concepts that attributes represent. In order to make this broadly applicable AbsMatcher then assumes only a simple data model. This also has the benefit of demonstrating a lower bound for when previous systems are not applicable. Given this baseline, AbsMatcher was designed with the flexibility to include more advanced forms of information when present.

The ABSURDST algorithm combines within-schema information, graphs, and cross-schema information, similarity matrices, using an iteratively converging global optimization algorithm. ABSURDIST [9, 11] was originally developed to translate between conceptual systems in a psychologically plausible manner. ABSURDIST represents each schema to be matched by a graph with attributes as nodes and relationships as edges between nodes. We refer to these graphs as internal information, because a graph only contains information relating attributes within the same schema. ABSURDIST incorporates similarities between attributes in the two schemas as a matrix of similarities, which we refer to as external information. As an added layer of flexibility, ABSURDIST has a weighting ratio to determine the balance of influence on the outcome of internal and external information.

We use the terms AbsMatcher and ABSURDIST throughout this work. AbsMatcher is the overall system which formulates graphs after mining internal information and aggregates mined sources of external similarity. ABSURDIST is the algorithm which iteratively combines internal and external information to determine a set of correspondences.

## II. ABSURDIST Background

ABSURDIST was developed to solve the general problem of translating between two conceptual systems. We adapt this approach to data integration by treating attributes as concepts to be matched. A complete discussion of ABSURDIST and how information factors into the iterative process can be found in [11]. Information in ABSURDIST is classified as internal (within-schema) or external (cross-schema). External information provides the ability to input cross-schema similarity into the ABSURDIST algorithm. Different external sources are aggregated into an NxM matrix of values between

0 and 1, where N and M are the sizes of the schemas to be matched. The dividing line between internal and external is that internal information is relationships between attributes in the same schema, whereas external similarity is a comparison between attributes in two separate schemas.

ABSURDIST iteratively updates correspondences using internal and external information until reaching a stable point, terminates, and selects the final matches. ABSURDIST as an error minimization algorithm selects the set of matches that result in the least total link error. This section discusses the conceptual motivations of ABSURDIST and leaves specific examples internal and external information for later sections.

## *A. Internal Information as Graphs*

Information in ABSURDIST is classified as internal, when it is derived from a single data set, and will therefore is a one-time cost no matter how many other schemas it is matched against. For each schema, ABSURDIST takes internal information as input in the form of; information on relationship types, node types, node information, and a graph of relationships. Internal information factors into the *R* and *I* terms of Equation 1. The minimal requirement for node information is a unique identifier and a node type. If only one type exists then the effects of node types become irrelevant. Relationships in ABSURDIST represent a conceptual association between attributes creating a generalized interpretation of structure. The requirement for a relationship type is a label and an indication of being either directed or undirected. Relationships are instantiated as edges, which collectively form a graph. Edges are binary, have a relationship type, and are weighted zero to one. If the same weight is used for every edge, it becomes irrelevant. When a relationship type does not naturally have a weight, the same weight is used in order to meet ABSURDIST requirements. Specific types of internal information we automatically mine for in Section 3.

## *B. Iterative Algorithm*

ABSURDIST is an iterative algorithm which updates a NxM matrix of correspondences where N and M refer to the number of attributes in the source schema, *A*, and target schema, *B*, respectively. Each cell in the correspondence matrix, $C_t(A_q, B_x) C_t(A_q, B_x)$ represents how strong a match is at iteration step *t* for attribute *q* in schema *A* and attribute *x* in schema *B*. The algorithm terminates when the matrix has converged or a maximum number of iterations is reached. For each iteration, ABSURDIST updates each cell according to the following equation:

$$N(A_q, B_x) = \alpha E(A_q, B_x) + \beta R(A_q, B_x) - \chi I(A_q, B_x)$$

Equation 1 Correspondence Update Equation

Equation 1 shows how internal (*R* and *I*) and external (*E*) information combine to update the correspondence from attribute *q* in schema *A* to attribute *x* in schema *B*. The *E* term represents similarity based on external information, the *R* term represents similarity based on internal information, and the *I* term uses internal information to inhibit incorrect correspondences. α, β, and χ are weights that control the influence of forms of information, where α and β are set as a ratio to each other and χ is set independently of the others. For example, when α is one and β is zero only external information is used to find correspondences.

## III. Related Work

There has been extensive work on graph matching and schema matching. A number of surveys have been done which cover the different aspects of the schema matching problem [1-3]. One of the established approaches to schema matching [4] is to use candidate matchers to generate candidate matches which are aggregated into a final set. Graph-based systems, including AbsMatcher, have multiple modules to generate edges in the graph, multiple modules to generate the equivalent of external information, and then use a graph matching algorithm to generate correspondences based on graphs. It is possible that correspondences generated using a graph matching algorithm could be used as a candidate matcher in a system. The Fuzzy Constraint Matching [5], Cupid [6], and Similarity Flooding [7] systems all use graph matching to accomplish schema matching. COMA++ [8], is a generalized framework for schema matching which was used in the Similarity Flooding system to combine the results from graph matching with non-graph-oriented candidate matchers. The difference between AbsMatcher and these previous systems is the generality of AbsMatcher and generating graphs based on semantics instead of data model metadata.

Previous graph-based schema matchers construct graphs based on the metadata for the data model. These systems have modules specifically built for translating different data models -- such as relational databases, XML, ontologies, or conceptual hierarchies -- into a graph form. This approach makes the graphs generated dependent on the thoroughness of the data set creator, and completely different graphs will be generated even when the same data set is stored in different data models. The advantage of these systems is that they leverage the effort of data set creators. For example considerable effort is generally put into the design phase of a relational database. Examples of using metadata would be creating a relationship between parent and child XML attributes or the fact that an attribute is a primary key in a relational database. The disadvantage of basing graphs on metadata is that derived relationships often have more to do with how data is stored and less about semantic relationships. The goal of the information sources we present in this paper is that they can be used regardless the data model and still generate semantic relationships.

The Semantic Matching [9] system provides the closest comparison to AbsMatcher. It creates a graph based on metadata and a limited number of semantic relationships. Semantic Matching uses electronic thesauri in order to create *overlap, mismatch,* and *general/specific* relationships. The one issue with electronic thesauri is that they only work with words in their index and are unable to handle abbreviations or phrases which are often used to name attributes. AbsMatcher shares the same motivation as Semantic Matching, but uses the web to create semantic relatedness relationships and mines the data sets for statistical relatedness relationships. Additionally, ABSURDIST was designed with a general idea of relationships, which makes adding new forms of internal relationships a simple process.

## IV. Mining ABSURDIST graphs

The focus of this paper is on sources of internal information which are new to graph-based schema matching. Some of these sources use only the metadata while others mine the data itself. An important feature of these information sources is that they only rely on ubiquitous elements of data models, e.g. named attributes, and yet are still able to create a graph of semantic relationships. These relationships provide a baseline for a graph-based approach to schema matching in the absence of rich metadata and/or user created structure that previous systems require. As we describe forms of internal information we will also describe the requirements for their use.

Mining an ABSURDIST graph is a two-stage process. The first is mining edges of the desired relationship type and the second is filtering out noisy edges. For brevity's sake we omit a discussion of the filtering process. We have two categories of relationships; ones which use the entropy of the data and the semantic relatedness of attribute names using web query counts. For data sets in XML format we tested a form of internal information which created relationships between sibling attributes, but omit it for brevity because it did not have a significant impact on performance.

### *C. Entropy Relationships*

Entropy-based relationships use an information theoretic approach to look at the information content of attributes based on their data. The goal is to look for patterns which defy statistical trends and therefore are more likely to represent user intended relationships. We use the Information Dependency (InD) measure [10], which is based on Shannon's Entropy, to look at the information content of attributes. We used three different entropy relationship types. *Entropy* relationships require at least a sample of the data.

#### *1) Attribute Entropy Relationships*

Attribute entropy relationships measure the degree to which attributes resemble keys, which have a different value in each record in the data set for the attribute, or constants, which have the same value in each record in the data set for the attribute. An attribute being close to a key or constant is a unique statistical property which is a result of how data is created. Because of this attributes in other data sets are likely to share the same statistical property. Some data, e.g. ISBN, inherently possesses key prosperities, so when keys or constants occur they are strong indicators of a likely match. In Table 1, PersonName is an example of a key and Gender is an example of an attribute that is almost a constant.

#### *2) Data Set Key Relationships*

Data set keys are sets of attributes that together have a unique set of values for the data set and therefore form a key. Data set key relationships are created between pairs of attributes that together are close to forming, or do form, a data set key, but neither attribute is a key on its own. An example from Table 1 is that by combining Address and Gender a unique set of values exists for every row. The above example would result in an edge *PairKey*(Address, Gender) to be created in the graph. A data set key relationship creates undirected edges between attributes and uses the entropy value as the weight.

TABLE I A SAMPLE DATA SET OF PEOPLE

| **PersonName** | **Address** | **Gender** |
|---|---|---|
| Santa Claus | 100 North Pole | Male |
| Mrs. Claus | 100 North Pole | Female |
| Jeremy Engle | 215 Lindley Hall | Male |
| Rob Goldstone | 120 Psychology | Male |

#### *3) Dependency Relationships*

The last entropy relationship type uses Approximate Functional Dependencies (AFDs). AFDs are probabilistic rules, $X \rightarrow Y$, which measure the ability of values for a left hand side (LHS) attribute set to determine values of the right hand side (RHS) attribute set. The closer an AFD's measured value is to 1 the better the LHS is at predicting the RHS. We use AFDs which have a single attribute LHS and a single attribute RHS in creating dependency relationships. The mining of AFDs has been studied in [11, 12], though by only using single attributes on each side the search space is reduced from $2^{N+M}$ to NxM. Though Functional Dependencies (FDs), which AFDs extend, have been used in schema matching, this is to our knowledge the first use of AFDs.

### *D. Semantic Relatedness Relationships*

The premise behind using semantic relatedness is to create a relationship between attributes that are thematically related. A trivial example of this would be attributes for the first and last name of a person. If the respective attribute labels are "first" and "last" then a graph edge is created between these attributes based on the thematic association of these labels.

$$WebJaccard(P,Q) = \begin{cases} 0, & H(P \cap Q) < c \\ \dfrac{H(P \cap Q)}{H(P) + H(Q) - H(P \cap Q)}, & H(P \cap Q) \geq c \end{cases}$$

Equation 2: WebJaccard using Yahoo! Query Hits

One of the common tools for mining semantic relatedness is using WordNet [13]. Semantic relationships are found for two words according to their common membership in sets of synonyms, or synsets. Though WordNet has a large dictionary, the tools that rely on it fail when one of the two words is not in the dictionary. There are two common scenarios which increase the likelihood of WordNet failing. The first is that data sets commonly have domain specific terms that are less likely to be in a general dictionary like WordNet. The second problem is that data sets commonly have attribute names that are multiple words and/or use abbreviations. The tools making use of WordNet are not capable of handling either of these cases. In order to overcome these issues, we use tools that query the World Wide Web instead of WordNet.

We use the WWW as a source of information and adapt existing information retrieval measures to use the number of results from queries to compute similarity. Our semantic relatedness relationships are based on work by Bollegala et al. which queried Google and used the number of query results in computing existing similarity measures [14], however they only tested its use on single words. AbsMatcher uses query counts ($H$) with the WebJaccard measure as expressed in Equation 2. Before querying attribute names, we clean attribute names by tokenizing them on occurrences of underscores and capital letters. Though this is not foolproof it does provide a best effort for creating multi-term queries. When available we also include the data set name as a query term to provide sense disambiguation. We then use Yahoo! as a source for querying.

## V. Mining the external similarity matrix

We use existing sources of external information, and therefore only discuss them briefly. External information directly compares attributes in the source and target schemas to look for similar attributes. While mining external similarity both attribute names and values from the data are used. We tested basic sources of external information to investigate the effects of combining internal and external information. Two sources of external similarity were prototyped and tested.

The first source of external similarity is string edit distance [15], which is a lexical comparison of attribute names. String edit distance represents a method for finding matches that are "low hanging fruit." We use the jSimlib [1] library that normalizes string edit distance by the sum of the length of the two strings.

The second source of external similarity is cosine similarity, which is commonly used to compare the similarity of two free text documents. The similarity of the two documents is computed as the cosine value between the term frequency vectors for each document. For attribute-to-attribute schema matching, when the attributes contain text we treat them as documents and create term frequency vectors. The Lucene[2] framework was used to calculate the cosine similarity.

## VI. Data sets

We tested three groups of data sets that vary in domain and size which come from the Illinois Semantic Integration Archive (ISIA) [16]. The Courses data sets have listings of classes from four different universities, data sets sizes range from twelve to sixteen attributes. The second group of data sets is the Real Estate I (REI) data sets, which includes the homeseekers, nky, windermere, and yahoo data sets. Three of the data sets have sizes in the mid-thirties and the final one is in the sixties. The third group of data sets is the Real Estate Core (REC) data sets. REC data sets are the same as the REI data sets, but only include attributes that have a match in one of the other data sets. This removal of attributes reduced the size of the data sets to the low twenties, except one having twenty-eight attributes. The REC group is used to test the effects on matching performance when attributes with no matches are removed.

## VII. Experimentation

The goals in evaluating AbsMatcher are to look at the performance of internal information by itself and whether there is still a benefit when combined with external information. *Semantic* and *Entropy* internal relationships are meant to provide a baseline ability for schema matching so performance is judged first by whether consistent evidence of an ability to find matches, and second by looking for evidence that combining internal and external information is better than only external information. Finding evidence of these two points would indicate matches being found which internal information can uniquely contribute to finding. Performance is measured using recall. Many schema matching systems provide statistical matches, as opposed to absolute matching, so we present recall for correct matches made and for the correct match being one of the top 3 best matches. This more liberal scoring criterion provides information on whether AbsMatcher has partial information that could be leveraged by future improvements to the algorithm or information sources. Precision is not included because currently AbsMatcher returns a match for each attribute in the smaller of the two schemas. This means that the number of matches returned for a pair of schemas will remain constant no matter what other parameters change. This point is discussed further in future work.

One of the challenges that AbsMatcher faces is tuning various parameters. The problem of how to find the best weightings among various matchers is a problem common in schema matching systems. There has been work to automate this tuning process [17]. As AbsMatcher is an initial effort we perform manual tuning.

## VIII. Sources of internal information

For testing purposes, the sources of internal relationships described in Section 3 are used in three combinations. The *Entropy* combination includes attribute entropy, data set key, and dependency relationships. The *Semantic* combination consists of semantic relatedness relationships based on Yahoo! results. Finally, the *All* combination includes both *Entropy* and *Semantic* relationships.

### *E. Results*

We first look at the extent to which schemas can be matched using only the mined graphs for the two data sets. When using only this limited source of information a high level of performance cannot be expected. However, this limitation is useful in making an initial judgment of whether mined graphs contain useful information. For each group of data sets we select the best performing parameters and present in. Fig. 1 results for all three combinations of internal relationships and all three groups of data sets.

[1] https://jsimlib.dev.java.net/

[2] http://lucene.apache.org/java/docs/index.html

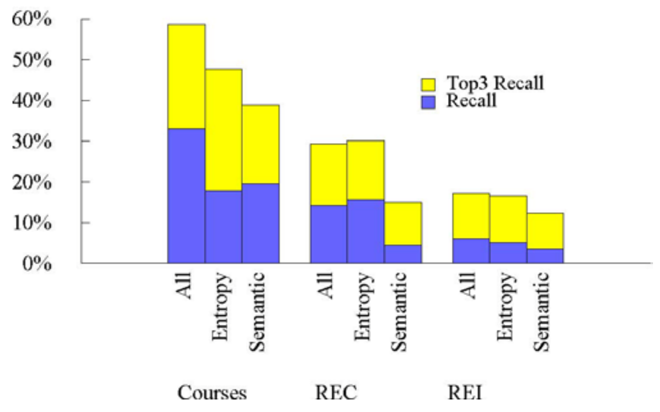

Fig. 1 Data sets by types of internal relationships

The first result to examine is AbsMatcher's ability to find correct matches. Though the results in Fig. 1 are relatively low in the context of overall performance of schema matching systems, the more appropriate context is as a source of matches which would be used in a broader system. In this context *Semantic* and *Entropy* relationships do show consistent ability to find at least some matches. The performance of the top 3 correspondences improves over just correct matches indicating that AbsMatcher can provide supporting evidence which would affirm or discredit correspondences from other candidate matchers. As seen in Fig. 1 the top 3 correspondences can provide useful results on a third to half of all matches. The top 3 matches can be useful when considering that the weights of correspondences in the top 3 can often be very close.

The second result to examine is what sources or combinations of sources of internal relationships are the most effective. Neither the *Entropy* or *Semantic* combinations were consistently the best between the different data set groups. Though neither was consistently the best, the positive result is that when combined in *All*, performance improved or matched the performance of the best performing source of internal relationships. The fact that adding sources of internal relationships does not degrade performance strengthens the potential that when other existing forms of internal relationships are added, performance could be improved.

## IX. External and Internal combined

For some matches the information which best indicates the correct match is derived by comparing an attribute from each data set. In ABSURDIST this means the use of external information that is combined with internal information using Equation 1. In Equation 1 there are two weighting coefficients, α and β, which determine the balance between external and internal information. The α:β ratio represents the comparative weights of external:internal information. We tested AbsMatcher with different ratios, where each represented a different balance between external and internal information. Fig. 2 presents results for a representative three of those ratios. The 0:1 data point represents using only internal information, which corresponds with the results in Fig. 1. The 1:0 data point represents only using external information. The 3:1 data point tested the effort to combine the use of internal and external information. The goal in this evaluation is to determine whether combining internal and external information has a benefit over just using external similarity.

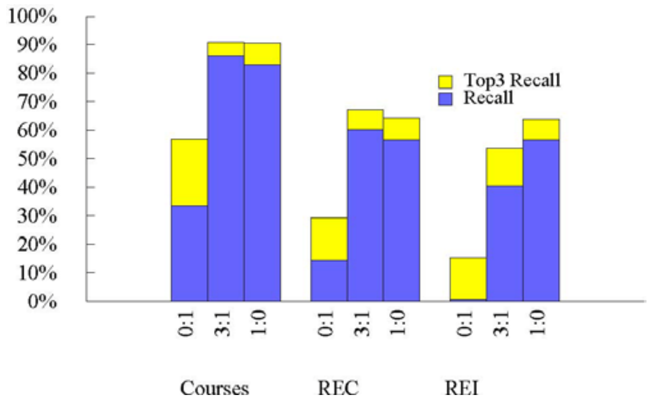

Fig. 2 Data sets by Ext:Int Ratio

### F. *Results*

Fig. 2 provides evidence that combining internal and external information can for some data sets provide better results than either one in isolation. Though the improvement for Courses and REC data sets is small the fact that it occurs for both supports the claim that internal structure can improve matching performance. It must be remembered that the results for Courses and REC represent the average performance across twelve different pairs of data sets matched. The ability of internal structure to find correct matches and the additional beneficial effect that it can have when combined with external similarity indicates that internal structure is to some extent finding both unique and useful information for schema matching.

The REI data sets do not benefit from internal information. This could in part be due to the fact that REI alignments leave more attributes unmatched. The REC data sets are versions of the REI data sets where attributes with no matches removed. They average 15.5 correct matches between a pair of data sets, meaning that on average half of the attributes in a data set for REI are not being matched, yet information is still mined for them. Courses and REC have a different scale yet both show similar trends in the ability to find correct matches. The only difference between REC and REI data sets is the existence of unmatched attributes, so the difference in performance can be unambiguously attributed to this. This indicates that information which indicates invalid matches could be an important feature to add to AbsMatcher.

## X. Conclusion

The goal in developing AbsMatcher was to create a schema matching system that used a graph based approach, but was not reliant on a specific data model as a source of information. To this end, we limited AbsMatcher to pervasive information for creating graphs in order to create a baseline. The *Entropy* and *Semantic* forms of relationships also are based on finding relationships between semantic concepts and not how data is being stored. We then tested these graphs using the ABSURDIST graph matching system. ABSURDIST is ideally suited because of its ability to accept graphs with a wide variety of forms (weighted, unweighted, directed, undirected, labelled, and unlabeled) and ABSURDIST was designed specifically with the idea of combining internal and external information together.

The goals in testing AbsMatcher were to look at whether *Entropy* and *Semantic* internal structure are useful for schema matching on their own and whether they have benefits when combined with external similarity. Experiments demonstrated that to varying extents the tested forms of internal structure are able to accomplish both of the goals. The results presented in this paper where aggregated over multiple individual experiments. The additive benefit of our sources of internal structure is important because it argues that internal structure holds unique information for finding correspondences.

These results were based on aggregating results from a number of matching pairs. It is important to note that there were outliers on both the positive and negative side. This is a common problem in schema matching, where sources of information perform well in certain scenarios and poorly in others. It is this point which motivated the approach of aggregating many disparate measures of similarity. This leads to the idea that by adding new information sources into AbsMatcher we can improve even beyond the baselines presented in this work.

## XI. Future Work

AbsMatcher has provided an initial effort with a baseline of results. Going forward there are advances that could be made for all three aspects of AbsMatcher; internal information, external information, and the ABSURDIST algorithm. One of the techniques which could be added for both internal and external information is negative information. We saw when comparing REC and REI that there are situations when not making a match can be just as important as making a match.

There are a number of potential ways that internal structure could be improved. The first and most interesting would be using the techniques in [6, 7] that look at relational metadata or tree representations and combine them with *Entropy* and *Semantic* relationships. This would also provide the potential to expand AbsMatcher to include entity-to-entity matching along with the current attribute-to-attribute matching.

As with other systems similar to AbsMatcher, the ability to automatically tune the various parameters will remain a direction of future work.

## References

[1] A. Halevy, A. Rajaraman, and J. Ordille, "Data integration: the teenage years," in *VLDB*, 2006, p. 06.

[2] P. Shvaiko and J. Euzenat, "A Survey of Schema-Based Matching Approaches " *Journal on Data Semantics IV*, vol. 3730, pp. 146-171, 2005.

[3] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *The VLDB Journal*, vol. 10, pp. 334-350, 2001.

[4] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos, "iMAP: discovering complex semantic matches between database schemas," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data* Paris, France: ACM, 2004.

[5] A. Algergawy, E. Schallehn, and G. Saake, "Fuzzy Constraint-based Schema Matching Formulation," in *1st Workshop on Advances in Accessing Deep Web (ADW 2008)*, 2008, pp. 141--152.

[6] J. Madhavan, P. A. Bernstein, and E. Rahm, "Generic Schema Matching with Cupid," in *VLDB*, 2001.

[7] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching," in *ICDE*, 2002.

[8] D. Aumueller, H.-H. Do, S. Massmann, and E. Rahm, "Schema and ontology matching with COMA++," in *Proceedings of the ACM SIGMOD international conference on Management of data* Baltimore, Maryland: ACM, 2005.

[9] F. Giunchiglia and P. Shvaiko, "Semantic Matching," *Knowl. Eng. Rev.*, vol. 18, pp. 265-280, 2003.

[10] M. M. Dalkilic and E. L. Roberston, "Information dependencies," in *Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* Dallas, Texas, United States: ACM, 2000.

[11] J. T. Engle and E. L. Robertson, "HLS: Tunable Mining of Approximate Functional Dependencies," in *British National Conference On Databases (BNCOD)* Cardiff, Wales, 2008, pp. 28-39.

[12] Y. Huhtala, J. Karkkainen, P. Porkka, and H. Toivonen, "Tane: An Efficient Algorithm for Discovering Functional and Approximate Dependencies," *The Computer Journal*, vol. 42, pp. 100-111, February 1, 1999 1999.

[13] C. Fellbaum, *Wordnet: An Electronic Lexical Database*: Bradford Books, 1998.

[14] D. Bollegala, Y. Matsuo, and M. Ishizuka, "Measuring semantic similarity between words using web search engines," in *Proceedings of the 16th international conference on World Wide Web* Banff, Alberta, Canada: ACM, 2007.

[15] G. Navarro, "A guided tour to approximate string matching," *ACM Comput. Surv.*, vol. 33, pp. 31-88, 2001.

[16] A. Doan, "Illinois Semantic Integration Archive, http://pages.cs.wisc.edu/~anhai/wisc-si-archive/."

[17] Y. Lee, M. Sayyadian, A. Doan, and A. S. Rosenthal, "eTuner: tuning schema matching software using synthetic scenarios," *The VLDB Journal*, vol. 16, pp. 97-122, 2007.