Extracting Assumptions from Missing Data

Honglei Zeng[1] and Richard Fikes[1]

[1] Computer Science Department, Knowledge System, AI Laboratory, Stanford University, California, USA 94305.
{hlzeng, fikes}@ksl.stanford.edu

Abstract. Information integration is the task of aggregating data from multiple heterogeneous data sources. The understandings of context knowledge of data sources are often the keys to challenging problems in information integration such as handling missing and inconsistent data. Context logic provides a unified framework for the modeling of data sources; nevertheless, the acquisition of large amounts of context knowledge is difficult. In this paper, we study the importance of a special type of context knowledge, namely assumption knowledge. Assumption knowledge refers to a set of implicit rules about assumptions on which a data source is based. We develop a decision tree classifier to extract assumption knowledge from missing data and formalize the knowledge in context logic. Finally, we build an information aggregator with assumption knowledge reasoning, which is capable of explaining incomplete data aggregated from heterogeneous sources.

1 Introduction

Information integration is the task of aggregating data from multiple independent, heterogeneous data sources. As the World Wide Web is evolving into a gigantic repository of information and knowledge, information integration on the Web has become increasingly important and of tremendous practical use from online business to personal assistant agents. Information integration on the Web poses a greater challenge than the traditional database information integration due to the unstructured or semi-structured, uncertain, incomplete and inconsistent nature of the Web.

Context Logic, formalized by McCarthy ([1]), Guha ([2]) and Buvac ([3]), provides a unifying framework for the modeling of data sources. The basic notation is ist(c, p) meaning that the proposition p is true in context c. A context is an abstraction of a data source. Intuitively, it may be modeled by a set of true statements that describes the world associated with the source.

The following example illustrates the expressiveness of context logic in data integration. Suppose we want to find movies starring actor Tom Hanks on the Internet Movie Database ([4]) and the Yahoo Movies ([5]). This task often starts by extracting data from semi-structured web pages to user defined data structures. For example, if we know from Yahoo Movies website "The release year of 'The Terminal' is 2004", we may encode the fact in a context logic formula:

```
ist(YahooContext, ReleaseYear("The Terminal", 2004)).
```

YahooContext refers to the context of Yahoo Movies website. If variable x is defined as a movie and variable y is defined as a year, then ReleaseYear(x, y) means "the release year of movie x is y".

Context logic is a powerful theory in addressing many semantic issues in information integration. Data sources do not typically have shared ontology and vocabulary which causes many semantic issues, such as polymorphism of names. For example, the same movie is listed as having the title "Return of the Jedi" on Yahoo and as having the title "Star Wars: Episode VI - Return of the Jedi" on IMDB. Lifting axioms, or axioms that relate one context to another, are natural choices for representing such knowledge as follows:
`ist(YahooContext, ReleaseYear("Return of the Jedi", 1983))` ←→ `ist(IMDBContext, ReleaseYear("Star Wars: Episode VI - Return of the Jedi", 1983)).`

Furthermore, if the information extractors for IMDB and Yahoo are developed separately, especially if by different persons, the extracted data may be stored in different data schemas. For example, we may use ReleaseYear relation in the Yahoo extractor while using MovieReleaseYear relation in the IMDB. The new lifting axiom is:
`ist(YahooContext, ReleaseYear("The Terminal", 2004))` ←→ `ist(IMDBContext, MovieReleaseYear("The Terminal", 2004)).`
In General, such mapping can be easily encoded in lifting axioms due to the expressiveness of context logic.

The understandings of context knowledge of data sources are often the keys to challenging problems in information integration, such as handling missing and inconsistent data. It is difficult to give a precise definition of context knowledge. In this paper when we say context knowledge of a data source, we are informally meaning the collection of provenance, situations, assumptions, biases, domain knowledge, prior events and other relevant information of that source. Context logic provides a unifying framework for the modeling of data sources; nevertheless, the acquisition of large amounts of context knowledge is difficult and possibly infeasible.

One important type of context is "assumption context knowledge" or simply "assumption knowledge". Assumption knowledge refers to a set of implicit rules about assumptions on which a source is based. We focus on assumption knowledge in this paper for two reasons: First, we believe assumption knowledge is one of the most important context knowledge in solving many difficult issues in data integration; secondly, we develop a decision tree classifier to extract assumption knowledge from incomplete data, thereby making the knowledge acquisition automatic and efficient.

Continue our example on finding movies of Tom Hanks from IMDB and Yahoo. As one may expect, both sites contain mostly the same Tom Hanks' movies, except for maybe a few. For example, the movie "A Cold Case" on IMDB is missing from Yahoo, while "Solaris/Cast Away", (which is not a movie, but rather a DVD combo) only appears on Yahoo. Apparently, both IMDB and Yahoo movies are credible sources of Tom Hanks' movies. Who can one trust if they give different answers?

A simple approach would be to list any movie that appears on either IMDB or Yahoo. A more sophisticated approach leverages users' preferences over data sources when trying to combine partial results. For example, this approach may accept data that a majority of the sources agree on or reject data from less credible sources. Nevertheless, both approaches fail to explain why a certain movie is only contained on IMDB but missing from Yahoo and vice versa. Lack of information about the discrepancies between multiple websites undermines user trust. In many cases, such trust is necessary for users to utilize the aggregated data to its fullest.

Our approach relies in the fact that IMDB and Yahoo have different assumptions to list movie information. For example, Yahoo has an incomplete list of upcoming movies: typically, only high-profile upcoming movies, such as Star Wars III are listed on Yahoo Movies. That's the reason "A Cold Case" is missing from Yahoo. Such assumption knowledge about IMDB and Yahoo is the key to user trust. Further, we may make aggregated data more accurate by excluding data with undesired underlying assumptions. In our example, we may exclude "Solaris/Cast Away" from Tom Hanks movies list since it is a DVD combo.

Throughout this paper, we restrict our attention to information integration from sources with data that are similar and closely related, as in the above movie example. This type of integration is a special form of horizontal integration, in which instances of a certain concept (e.g. movies of Tom Hanks) are distributed across multiple sources. In other types of information integration, such as vertical integration (in which, fragments of instances are distributed across source), handling missing instances in different sources is not a major issue; therefore, they are not discussed in this paper.

Even though knowing assumption knowledge is critical in many scenarios, little has been done to acquire assumption knowledge from sources automatically. To address such a need, we build a decision tree based classifier to learn assumption knowledge from the discrepancies between data sources. In section 2, we show decision trees and the techniques for using them to learn assumption knowledge. In section 3, we discuss the benefits of having assumption knowledge. In particular, we build an information aggregator with assumption knowledge reasoning, which is capable of explaining incomplete data aggregated from heterogeneous sources. The last section concludes with a summary and a discussion of related work and future work.

2 Extracting assumption knowledge from missing data

Decision tree induction is one of the most useful classification methods in machine learning. In that method, decision trees are constructed from a training set consisting of data tuples described by a fixed number of values of predictor attributes. Each data tuple in the training set is also associated with one classification label. Two popular decision tree programs C4.5 [6] and CART [7], as well as most other programs, go through two phases for constructing decision trees: the tree growing phase and the tree pruning phase. A decision tree is ready to be used as a classifier once it is constructed. It takes a data tuple as input and predicts the classification label of the tuple. Decision trees have become a very popular data mining technique because of their predicative and descriptive power to classify new data and to summarize new concepts, in addition to their easily accessible algorithms. In the following example, we demonstrate how decision trees can be used to retrieve context knowledge from information sources and explain incomplete data.

Table 1 consists of six data tuples extracted from IMDB and four from Yahoo Movies. These ten tuples denote eight movies as tuple t5 and t7 refer to the same movie, and similarly for t6 and t8. In this simplified example, each data tuple has five attributes: Tuple ID, name, main actor, type, and release year. Each tuple also has exactly one class label, in three possible values: IMDB, Yahoo, and IMDBYahoo. This class label indicates that a movie denoted by a tuple is on IMDB only, Yahoo only, or on both.

Table 1. A partial list of movies from IMDB and Yahoo

Information source A (IMDB.com)

Tuple ID	**Name**	**Actor**	**Type**	**Year**	**Class**
t1	The Black Dahlia	Hilary Swank	Movie	2005	IMDB
t2	Little Fish	Cate Blanchett	Movie	2005	IMDB
t3	Band of Brothers	Tom Hanks	TV	2001	IMDB
t4	Midwives	Sissy Spacek	TV	2001	IMDB
t5	A Beautiful Mind	Russell Crowe	Movie	2001	IMDB Yahoo
t6	In the Bedroom	Sissy Spacek	Movie	2001	IMDB Yahoo

Information source B (Yahoo movies)

Tuple ID	**Name**	**Actor**	**Type**	**Year**	**Class**
t7	A Beautiful Mind	Russell Crowe	Movie	2001	IMDB Yahoo
t8	In the Bedroom	Sissy Spacek	Movie	2001	IMDB Yahoo
t9	Solaris/Cast Away	Tom Hanks	DVD	2000	Yahoo
t10	The Terminal/Catch Me If You Can	Tom Hanks	DVD	2005	Yahoo

The key observation in Table 1 is that some movies on IMDB (i.e. t1, t2, t3, and t4) are not on Yahoo and vice versa (i.e. t9 and t10), even though both websites are credible sources for movie information. By assigning a class label to each tuple, we turn the problem of explaining content discrepancies into a problem of classification.

That is, we can build a decision tree given the set of training data in Table 1. If two sources provide identical information, the decision tree contains only one root node that classifies any tuple into one category and is therefore trivial. However, if the information is different, the decision tree would capture the context differences among the sources. From another perspective, the decision trees characterize a source in terms of attribute values by examining the content discrepancies, without which, the underlying assumptions would remain hidden.

A decision tree is shown in Figure 1 which is constructed by feeding the training tuples in Table 1 into the C4.5 decision tree generator. As simple as it is, the tree captures some interesting assumptions about IMDB and Yahoo, and can be used to explain the missing tuples. For example, we know Yahoo Movies website does not list TV programs, which is why tuple t3 & t4 are missing from Yahoo. In other words, we find a set of rules represented in the decision tree to characterize both IMDB and Yahoo.

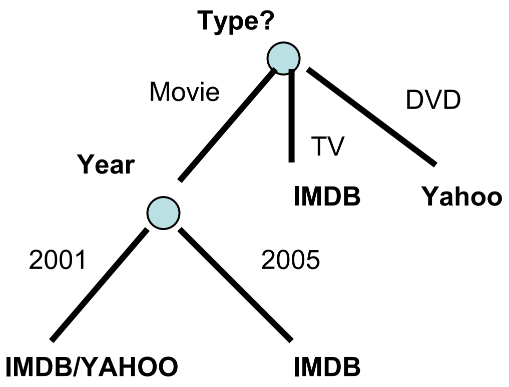

Fig. 1. Decision tree generated from Table 1

Without confusion, Figure 2 shows the same tree rendered in an ASCII format, the standard output format of the C4.5 program. In this example, the root is the attribute test "type", which has three attribute-values: "Movie", "TV", and "DVD". Following the test, is either a classification label if it can be decided by the test, or a sub-tree of another attribute test, indicated by a vertical bar; the sub-tree can be recursive if more tests are required.

```
type = TV: IMDB_only
type = DVD: Yahoo_only
type = Movie:
|   Year = 2005: IMDB_only
|   Year = 2001: IMDBYahoo
```

Fig. 2. The same decision tree as fig 1. in ASCII tree format

Generating training data such as in Table 1 from information sources is the central issue for constructing a decision tree. We consider a two step approach. First, we write web wrappers to extract structured data from HTML documents and map extracted data to a target movie schema. To simplify our experiments, both IMDB and Yahoo extractors share the same schema. However, in the real world, it is very likely that IMDB and Yahoo extractors are developed separately, or even by different persons. In that case, reconciliation of semantic heterogeneities is a major challenge. Nevertheless, we believe approaches like LSD (Learning Source Descriptions) [8] are effective in learning the mappings between source schemas in horizontal integration. The second step is to decide the classification label for each tuple (i.e. identify equal tuples on IMDB and Yahoo). This is currently done by a program automatically checking the similarities of the movie name and attribute values. Nevertheless, this step may be nontrivial in many scenarios; for example, identifying equal tuples on an English movie website and a Spanish movie website.

Formally, we define a tuple $t = (v_1, v_2, ..., v_d)$, where each v_i is a value of an attribute T_i. We assume an information source i can be represented by a collection of tuples with the notation $D_i = \{t_1, t_2, ..., t_m\}$, where t_i are tuples. For example, ("t4," "Midwives," "Sissy Spacek," "TV," 2001) is a tuple of IMDB in Table 1. Data integration can be viewed as combining multiple D_i into a single consistent collection. For simplicity, without loss of generality, only two sources are considered in the rest of this paper. Incomplete data in this regard, is defined to be data tuples that are only contained in one source:

$$\{t_i \mid (t_i \in D_A \wedge t_i \notin D_B) \vee (t_i \in D_B \wedge t_i \notin D_A)\}.$$

Multiple sources can be dealt pair-wise. Our goal is to construct decision trees that represent assumption knowledge of both sources and explains the incomplete data, i.e., finding the reasons that a tuple in one source is not contained in the other and vice versa. Since the decision trees are learned from each source pair, the total number of the decision trees grows in $O(N^2)$, where N is the number of sources. Nevertheless, current integration systems are limited to a small number of sources.

The above definitions assume common attribute schema and vocabulary across the sources. Obviously, these assumptions are hardly true in reality. Instead of relying on schema alignments and ontology translation, our solution is to construct separate decision trees for each source so that each tree is built from tuples from one source rather than from all sources. As a result, we need to build two decision trees in our example; one for IMDB, and one for Yahoo Movies, which are shown in Figure 3. Each tree now has two classification labels instead of three: the IMDB decision tree has (IMDB_only, IMDBYahoo), and the Yahoo tree has (Yahoo_only, IMDBYahoo). This idea minimizes co-reference and schema problems because only tuples from the same source are considered for constructing a tree; nevertheless, it well preserves the effectiveness of representing assumption knowledge. Intuitively, we examine other sources for content discrepancies, but we only construct a decision tree within one source. In Figure 3, two trees collectively represent much of the same assumption knowledge as we obtained in Figure 1, although tuples from IMDB can solely be classified by the IMDB decision tree now and the same is true for Yahoo Movies. The second advantage in our approach is that each tree is a clear characterization of source assumptions. For example, Yahoo Movies website does not list TV information, although they are listed on the IMDB website.

```
Tree 1: IMDB Decision Tree
type = TV: IMDB_only
type = Movie:
|  Year = 2005: IMDB_only
|  Year = 2001: IMDBYahoo

Tree 2: Yahoo Movies Decision Tree
type = Movie: IMDBYahoo
type = DVD: Yahoo_only
```

Fig. 3. Separate decision trees for IMDB and Yahoo

Assumption knowledge represented in decision trees in figure 3 can be easily translated into context logic formulas in figure 4.

Characterization axioms

(1) Ist(IMDBContext, type(x, TV)) → Contains(IMDB, x) ^ ~Contains(Yahoo, x).
(2) Ist(IMDBContext, type(x, Movie) ^ year(x, 2001)) →
Contains(IMDB, x) ^ Contains(Yahoo, x)
(3) Ist(YahooContext, type(x, Movie)) → Contains(IMDB, x) ^ Contains(Yahoo, x)
…

Fig. 4. Predicate Contains(s, x) means that object x appears in source s. The characterization axioms describe possible conditions in which missing data could occur and therefore these axioms can be used to reason about explanations for missing data.

Choosing attributes that represent underlying source assumptions is another challenge in building decision trees. A good set of attributes and values is critical because decision trees built from them can be very effective given the good attributes, but ineffective given the bad ones. We use techniques of both manual attribute extraction [19] and automatic attribute extraction [20]. A large number of websites, including IMDB and Yahoo Movies, create their web pages using the same generating scripts from databases. Therefore, it is possible to discover regularities between a number of related web pages from the same site by analyzing and matching HTML documents. In the final decision tree we built, only 4 tuples out of 122 are misclassified. The predictive accuracy of this decision tree on unseen data tuples of 200 instances is about 87%. Due to the novelty of our approach, there are little direct related results, as far as we know, that we can compare to. We plan to continue experimental evaluation with larger data sets in different domains. In addition, we are investigating automated methods on how to determine good and bad predictor attributes among those extracted.

3 Applications

Having learned assumption knowledge in context logic, we are now able to build an information aggregator capable of explaining incomplete data. The aggregator first gathers information from IMDB and Yahoo through extraction, in response to a user's query. If a movie in the answers does not appear on either IMDB or Yahoo (this is currently done by checking the similarities of the movie name and attribute values), an explanation is then generated through decision tree classification. However, a first order logic reasoner augmented with contexts reasoning would maximize the value of using context logic representation in many ways, such as checking inconsistencies and query-answering.

Partial output of our aggregation is provided in Figure 5. Although the explanations can only be represented in terms of the given attributes, they provide users a rich, trustful and more accurate aggregated data result. For example, a user would accept "The risk pool" as a Tom Hanks movie, but reject "Band of Brothers", based on the different explanations provided, although both are missing from Yahoo, "The risk pool" is an upcoming movie that has just been announced, while "Band of Brothers" is a mini series. Through a similar process, a user would accept "Polar Express: An IMAX 3D Experience" and reject "You've Got Mail/Joe vs. The Volcano."

The benefits of having assumption knowledge go beyond aggregating more accurate and trustful data, because the knowledge also provides good characterizations of information sources that may lead to other practical applications. For example, we can expect to find information on "Bosom Buddies" only on IMDB because it is a TV series. If a prediction can be made based on assumption knowledge, an intelligent aggregator may forward a user query directly to the source that it believes has the answer, thus saving time and resources. Assumption knowledge can also provide useful insights to users when understanding the discrepancies between sources is crucial. For example, it may help a loan agency to analyze credit reports from multiple sources, each of which may have a different set of assumptions and biases. In another example, a person who builds a movie website may validate his design using assumption knowledge produced by our system. This person can question whether missing information of upcoming movies is intentional on his website, or whether having TV series in the movie section is an error.

4 Summary and future work

In this paper, we developed a decision tree-based classifier to extract assumption knowledge from data sources and formalized assumption knowledge in context logic. We discussed various applications applying assumption knowledge; in particular, we built a data aggregator capable of explaining incomplete data aggregated from heterogeneous sources.

User Query: Movies of Tom Hanks
Aggregator output

Movies supported by both IMDB and Yahoo
1. The Polar Express
2. The Terminal
3. The Ladykillers

…

Movies supported by IMDB only
1. The Risk Pool (The reason for missing from Yahoo: VHS is not available, no WGA authentication and in announced status.)
2. A Cold Case (The reason for missing from Yahoo: VHS is not available, no WGA authentication and in pre-production status.)
3. Band of Brothers (The reason for missing from Yahoo: VHS is available and type is mini series)

…

Movies supported by Yahoo only
1. You've Got Mail/Joe vs. The Volcano (The reason for missing from IMDB: type is DVD combo)
2. Polar Express: An IMAX 3D Experience (The reason for missing from IMDB: type is IMAX)

…

Fig. 5. Sample output of information aggregator

Although decision trees have been used for a long time to learn patterns from a set of data instances, our approach applies a novel combination and augmentation of existing techniques to build decision trees from multiple sources, aiming at learning assumption knowledge by examining content discrepancies between sources. In addition, applying assumption knowledge to explain incomplete data is a new and effective approach. Current approaches rely on source preferences to combine partial results. For example, Motro et al. in [9] propose six metadata features, such as recentness, cost, and accuracy to describe the qualities of a data source, and ranks the answers using a utility function in the linear combination of the metadata features. More recently, a framework for specifying and reasoning on preferences among sources is proposed in [10]. Although their goal is to resolve data inconsistencies, their approach may apply to integration of incomplete data as well. A characterization of data provenance in database theory can also be found in [11]. Nevertheless, source preferences may not be sufficient to achieve highly trustful and accurate aggregation.

In the area of Information Retrieval and Data Mining, our work is related to research on combining results from multiple evidences [12] [13] and to Meta-Learning in Distributed Data Mining Systems [14][15]. Our work is significantly distinguished from other decision tree-based methods, which demonstrate applications for constructing decision trees for mining distributed data in which we build decision trees to acquire assumption knowledge to explain incomplete data.

Farquhar et al. ([16]) presented a framework of integrating heterogeneous information sources using context logic. We adopt most of their context logic formations, while our focus has been learning and reasoning about assumption context knowledge.

There are also a lot of important work in classification and clustering in relational data; for example, the probabilistic relation model by Taskar et al. [17] and the First-Order logic inductive learner by Slattery and Craven [18].

There are many possible ways to extend our work:

(1) Improving context reasoning with emerging Semantic Web techniques;

(2) Exploring other machine learning techniques that can extract assumption knowledge from information sources;

(3) Extracting provenance knowledge, bias knowledge, and other types of context knowledge from data sources; and

(4) Applying context knowledge reasoning to solve other difficult problems in data integration, such as resolving data inconsistency and schema alignment.

In the future, we plan to continue experimental evaluation with larger data sets in different domains. We believe context logic provides a unified framework for the representation of data integration. In addition, extractable context knowledge may be critical in solving many difficult issues in data integration.

Acknowledge

We would like to thank Debbie Barros for her help. We would also like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported in part by DARPA Explainable Knowledge Aggregation project.

References

1. McCarthy, J.: Notes on formalizing context. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (1993).
2. Guha, R. V.: Contexts: A Formalization and Some Applications. PhD thesis, Stanford University (1991).
3. Buvac, S.: Quantificational logic of context. In Proceedings of the Thirteenth National Conference on Artificial Intelligence (1996).
4. http://www.imdb.com/name/nm0000158.
5. http://movies.yahoo.com/shop?d=hc&id=1800010392&cf=movies&intl=us.
6. Quinlan, J.R: C4.5: Programs for Machine Learning, Morgan Kaufmann (1993).
7. Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J.: Classification and regression trees, The Wadsworth Statistics/Probability Series, Wadsworth and Brooks (1984).

8. Doan, A., Domingos, P., Levy, AY: Learning source description for data integration, in WebDB (Informal Proceedings), pp. 81–86 (2000)
9. Motro, A., Anokhin, P. and Acar, A.: Utility-based Resolution of Data Inconsistencies, IQIS 2004, International Workshop on Information Quality in Information Systems (2004).
10. Greco, G., Lembo, D.: Data Integration with Preferences Among Sources, ER 2004: 231-244.
11. Buneman, P., Khanna, S., Tan, W.C: Why and Where: Characterization of Data Provenance. ICDT 2001, 8th International Conference, LNCS 1973, Springer 316-330 (2001).
12. Bartell, B., Cottrell, G. and Belew, R.: Automatic Combination of Multiple Ranked Retrieval Systems, Research and Development in Information Retrieval, pages 173-181 (1994).
13. Vogt, C. and Cottrell, G.: Predicting the performance of linearly combined IR systems, proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval (1998).
14. Prodromidis, A., Chan, P. and Stolfo, S.: Meta-learning in distributed data mining systems: issues and approaches, Advances in Distributed and Parallel Knowledge Discovery, AAAI Press/MIT Press, 2000, pp. 81-87 (2000).
15. Caragea, D., Silvescu, A. and Honavar, V. Decision tree induction from distributed heterogeneous autonomous data, Proceedings of the 3rd International Conference on Intelligent Systems Design and Applications, pp.341—350 (2004).
16. Farquhar, A., Dappert, A., Fikes, R. and Pratt, W.: Integrating information sources using context logic. In the AAAI Spring Symposium on Information Gathering from Distributed, Heterogeneous Environments (1995).
17. Segal, E., Koller, D., and Ormoneit, D.: Probabilistic abstraction hierarchies. In Proceedings of NIPS (2001).
18. Slattery, S. and Craven, M.: Discovering test set regularities in relational domains, in Proceedings of ICML (2000).
19. Cunningham, H., Maynard, D., Bontcheva, K. and Tablan, V. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications, Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (2002).
20. Crescenzi, V., Mecca, V. and Merialdo, P. RoadRunner: Towards Automatic Data Extraction from Large Web Sites, Proceedings of 27th International Conference on Very Large Data Bases, pp 109-118 (2001).