Qualitative Effects of Knowledge Rules in Probabilistic Data Integration

Maurice van Keulen [#1], Ander de Keijzer [*2]

[#] *Faculty of Electrical Engineering, Mathematics and Computer Science*
[1] m.vankeulen@utwente.nl

[*] *Institute of Technical Medicine, Faculty of Science and Technology*
[2] a.dekeijzer@utwente.nl

[#*] *University of Twente, P.O. Box 217, 7500AE, Enschede, The Netherlands*

***Abstract*— One of the problems in data integration is data overlap: the fact that different data sources have data on the same real world entities. Much development time in data integration projects is devoted to entity resolution. Often advanced similarity measurement techniques are used to remove semantic duplicates from the integration result or solve other semantic conflicts, but it proofs impossible to get rid of all semantic problems in data integration. An often-used rule of thumb states that about 90% of the development effort is devoted to solving the remaining 10% hard cases. In an attempt to significantly decrease human effort at data integration time, we have proposed an approach that stores any remaining semantic uncertainty and conflicts in a probabilistic database enabling it to already be meaningfully used. The main development effort in our approach is devoted to defining and tuning knowledge rules and thresholds. Rules and thresholds directly impact the size and quality of the integration result. We measure integration quality indirectly by measuring the quality of answers to queries on the integrated data set in an information retrieval-like way. The main contribution of this report is an experimental investigation of the effects and sensitivity of rule definition and threshold tuning on the integration quality. This proves that our approach indeed reduces development effort — and not merely shifts the effort to rule definition and threshold tuning — by showing that setting rough safe thresholds and defining only a few rules suffices to produce a 'good enough' integration that can be meaningfully used.**

I. Introduction

Data integration is a challenging problem in many application areas as it usually requires manual resolution of semantic issues like schema heterogeneity, data overlap, and data inconsistency. In this report we focus on data overlap as a major source of semantic uncertainty and conflicts, hence for the need for human involvement. Data overlap occurs when data sources contain data about the same real world objects (RWOs). This problem is often referred to as *entity resolution*. Often advanced similarity measurement techniques are used to remove semantic duplicates and solve other semantic conflicts. An often-used rule of thumb states that about 90% of the development effort is devoted to solving the remaining 10% hard cases, because human knowledge is required to ultimately decide if two data items refer to the same RWO and, if so, how to resolve conflicts between the two. Note that strictly speaking, even for humans making an absolute decision may be extremely labor-intensive. Fig. 1 illustrates this: it may very well be that these two data items refer to the same RWO, namely a woman who recently got married and moved in with her husband.

Most data integration approaches require resolution of semantic uncertainty and conflicts before the integrated data can be meaningfully used [1]. We believe, however, that data integration can be made into less of a development obstacle by removing this restriction striving for less perfect, but near-automatic integration, i.e., "good is good enough" data integration. The idea behind our probabilistic data integration approach is to postpone resolution of the remaining 10% semantic uncertainty to a moment more natural to human involvement namely during querying. Being able to properly handle uncertainty in data can provide for near-automatic data integration. Parts of the data that require tighter integration can be improved incrementally while the integrated source is being used [2].

Data source 1

name:	Elisabeth Johnson
address:	Wall street 12
phone:	555-823 5430

Data source 2

name:	Beth Clark
address:	Robertson Ave 2
phone:	555-234 8751

Fig. 1. Example instances of two data sources with address cards

An overview of our approach is given in Fig. 2 [3], [4]. We view a database as a representation of information about the real world based on observations. In this view, data integration is a means to combine observations stored in different data sources. Since we focus on data overlap, we assume that the schemas of the data sources are already aligned. The DBMS becomes uncertain about the state of the real world when observations conflict or cannot be traced back to RWOs with certainty. We have chosen a representation of uncertain data that compactly represents in one XML tree all possible states the real world can be in, the *possible worlds*. Posing queries to an uncertain database simply means that an application may receive several possible answers. In many application areas, this suffices if those answers can be properly ranked according to likelihood. A user interacting with an application can provide feedback on the correctness or plausibility of these answers. This feedback can be traced back to possible worlds, hence be used to remove *impossible* worlds from the representation in the database. This incrementally improves the integration by reducing uncertainty [2].

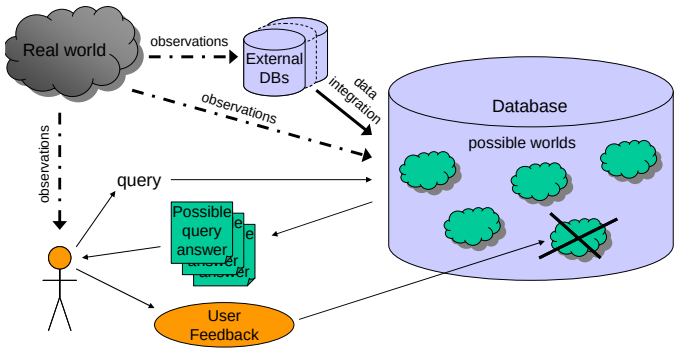


Fig. 2. Information Cycle

An automatic process can in theory never make an absolute decision for entity resolution. However unlikely, there are always situations imaginable where data items that completely differ still refer to the same rwo (as in Fig. 1) or where similar data items refer to different rwos. But, if a probabilistic integration system would consider and store all theoretically possible alternatives, the integrated data set would explode in size. Therefore, a human is still needed to help with the reduction in possibilities by specifying knowledge rules that make absolute decisions. An example of such a rule could be: "if according to some distance measure, address cards are further away than some threshold, assume that they do not refer to the same rwo. We observed that simple rules like this ruling out non-sensical alternatives often suffice to reduce the amount of uncertainty to a manageable size [5].

To prove, however, that our approach indeed reduces development effort significantly — and not merely shifts the effort to rule definition and threshold tuning — we need to show that setting rough thresholds and defining only a few rules suffices to produce a 'good enough' integration that can be meaningfully used. 'Good enough' here refers to the quality of the integration result. We define the quality of an integrated data set by means of the quality of the possible answers to queries. Ruling out *incorrect* possibilities not only results in a reduction in size of the integrated data set, it also results in a better *quality* integration, because incorrect possibilities lead to incorrectness in the answers. When rules make absolute decisions, they may, however, make a wrong decision (e.g., the given rule would make the wrong decision for Fig. 1). Ruling out *correct* possibilities in this way leads to a lower quality integration result. Therefore, there is a trade-off between a developer trying to specify stricter rules to reduce the amount of uncertainty and increase integration quality, and the likelihood that his rules make wrong decisions, which decreases integration quality. Observe that current data scrubbing and entity resolution techniques make an absolute decision at integration time for all data items, hence are likely to make such mistakes and therefore do not produce the best quality integration result.

The effects of rules and thresholds on integration quality is far from trivial, because a particular case of semantic uncertainty may affect one query and not another. Moreover, there often exist dependencies between semantic conflicts. Upfront, it is also not evident how big the impact of additional uncertainty is on the integration quality. In other words, how rule definition and threshold tuning precisely affect integration quality needs to be investigated experimentally with real-life data. Such an investigation is the focus of this report: This report presents an experimental investigation of the effects and sensitivity of rule definition and theshold tuning on the integration quality.

Albeit an intuitively attractive notion, 'integration quality' is rather vague. Therefore, we have defined information retrieval-like query answer quality measures based on precision and recall. The statistical notion of *expected value* when applied to precision and recall naturally takes into account the probability with which a system claims query answers to be true.[1] The effect is that the quality of a correct answer is higher if the system dares to claim that it is correct with a higher probability. Analogously, incorrect answers with a high probability are worse than incorrect answers with a low probability.

The trade-off of a developer trying to specify stricter rules and thresholds to reduce the amount of uncertainty and increase integration quality, and the likelihood that his rules make wrong decisions which decreases integration quality, can now be more precisely defined. We forsee that as long as a developer's rules and thresholds do not make wrong decisions, precision goes up and recall remains the same with stricter rules and thresholds. If, however, they become too strict and start to make wrong decisions, precision and recall go down. Our hypothesis is, however, is that

1) Precision and recall are not very sensitive to low thresholds, hence developers can save much effort on threshold tuning by taking rough but safe thresholds, and
2) Just a few rules suffice to achieve acceptable integration quality, hence there is also not much effort required in rule definition.

A. Contribution

The contributions of this report are

- an experimental investigation of the sensitivity of rule definition and threshold tuning on integration quality, and
- based on these insights provide experimental evidence that our probabilistic integration approach indeed significantly reduces development effort.

B. Running example

As a running example and set-up for our experiments, we selected a typical web application that requires integration of data sources on the internet. The purpose of the application is to collect information on movies that are about to be aired on TV. It uses an internet TV guide[2] for finding out which

[1] In [6], we also proposed adapted notions of precision and recall. The latter coincided with the expected value of recall, but the former does not. We have chosen to also use the expected value of precision in this report instead of the adapted precision measure of [6].

[2] `www.tvguide.com`

www.tvguide.com	www.imdb.com
title: **The Namesake** year: 2006 genre: Drama actors: Jacinda Barrett (Maxine) Benjamin Bauman (Donald) Sudipta Bhawmik (**Subroto** Mesho) Sibani Biswas (Mrs. **Mazumdar**) Jessica Blank (Edith) Sabyasachi **Chakraborty** (Ashima's Father) Gary Cowling (Hotel Manager) Gretchen Egolf (Astrid) Rupak Ginn (Uncle) Josh Grisetti (Jerry) Jagannath Guha (Ghosh) Ruma Guha Thakurta (Ashoke's Mother) Glenne **Headley** (Lydia) Maximiliano **Herandez** (Ben) **Irrfan** Khan (Ashoke) Jhumpa Lahiri (Jhumpa **Mashi**) Dhruv Mookerji (Rana) Gargi Mukherjee (Mira Mashi) Sahira Nair (Sonia) B.C. Parikh (Mr. **Mazumdar**) Kal Penn (Gogol) Zuleikha Robinson (Moushumi) Sebastian **Roche** (Pierre) Justin Rosini (Marc) Tamal **Roy Choudhury** (Ashoke's Father) **Tanushree** Shankar (Ashima's Mother) Bobby Steggert (Jason) Tabu (Ashima) Baylen Thomas (Blake) Amy Wright (**Pam**) Jo Yang (Ms. Lu) *24 more actors*	title: **Namesake, The** year: 2006 genres: **Comedy**, Drama, **Romance** actors: Barrett, Jacinda (Maxine **Ratliff**) Bauman, Benjamin (Donald) Bhawmik, Sudipta (**Subrata** Mesho) Biswas, Sibani (Mrs. **Mazoomdar**) Blank, Jessica (Edith) **Chakravarthy**, Sabyasachi (Ashima's Father) Collins, Marcus (Graham) Cowling, Gary (Hotel Manager) Egolf, Gretchen (Astrid) Gerroll, Daniel (Gerald Ratliff) Ginn, Rupak (Uncle) Grisetti, Josh (Jerry) Guha, Jagannath (Ghosh) Guha Thakurta, Ruma (Ashoke's Mother) **Headly**, Glenne (Lydia **Ratliff**) **Hernández**, Maximiliano (Ben) Khan, **Irfan (I)** (Ashoke **Ganguli**) Lahiri, Jhumpa (**Aunt** Jhumpa) Mookerji, Dhruv (Rana) Mukherjee, Gargi (Mira Mashi) Nair, Sahira (Sonia **Ganguli**) Parekh, B.C. (Mr. **Mazoomdar**) Pasquale, Rose (Woman in Laundromat) Penn, Kal (Gogol Ganguli) Ritter, Allison Lee (Emily) Robinson, Zuleikha (Moushumi **Mazoomdar**) **Roché**, Sebastian (Pierre) Rosini, Justin (Marc) **Sengupta**, Tamal (Ashoke's Father) Sethi, Payal (Ashima's friend) Shankar, **Tanusree** (Ashima's Mother) Steggert, Bobby (Jason) Tabu **(I)** (Ashima **Ganguli**) Thomas, Baylen (Blake) Wright, Amy **(I)** (**Pamela**) Yang, Jo **(I)** (Ms. Lu)
time: 12:30 pm/ET date: June 5, 2008 channel: CMAX *other data like parental-rating, country, running-time, format, production-company, released-by*	directors: Nair, Mira *other data like locations, keywords, and plots.*

Fig. 3. Illustration of differences between our data sets (important differences in boldface).

movies are about to be aired as well as other information the website provides for these movies. The application furthermore *enriches* this information with data from IMDB[3]. In our experiments, we enrich it with information about genres, actors, directors, locations, keywords, and plots.

Fig. 3 shows the data about the movie 'The Namesake' from both data sets as an illustration of the differences and entity resolution problems the application faces. One entity resolution problem is that it cannot determine with certainty which movie of the TV guide corresponds with which movie in IMDB. We have observed typos and different naming conventions in the titles (e.g., 'The Namesake' vs. 'Namesake, The'). Furthermore, both data sources have data on actors and their roles in the movie. This poses a second entity resolution problem: the application cannot determine with certainty which actors and roles correspond. There are many typos and differences in naming conventions in actor names and role descriptions (e.g., 'Glenne Headley' with role 'Lydia' vs. 'Headly, Glenne' with role 'Lydia Ratcliff'). There is even a major semantic conflict on the role of Ashoke's Father: 'Tamal Roy Choudhury' vs. 'Tamal Sengupta' for which even a human would doubt whether or not this is the same actor.

[3]www.imdb.com

C. Overview

The report is organized as follows. Section II describes related research. We then introduce our probabilistic integration approach in Section III. Since measurement of the usually hard to grasp concept of quality plays a vital role in this research, we devote an entire section to this topic (Section IV). We then present our experiments in Section V. Finally, we present conclusions and future work in Section VI.

II. Related Research

Several models for uncertain data have been proposed over the years. Initial efforts all focused on relational data [7] and also currently efforts are being made in the relational setting [8], [9], [10], [11], [12]. With relational data models, two methods to associate confidences with data are commonly used. The first method associates the confidence scores with individual attributes [7], whereas the second method associates these confidence scores with entire tuples [9].

Semistructured data, and in particular XML, has also been used as a data model for uncertain data [13], [14], [3]. There are two basic strategies. The first strategy is *event based* uncertainty, where choices for particular alternatives are based on specified events [14], [13]. Nodes are annotated with event expressions which validate or invalidate the node and its subtree according to combinations of occurrences of events. One particular combination of events represents a possible world, hence all possible worlds can be obtained by enumerating all possible combinations. In event based models, the events are assumed to be independent of each other.

The other strategy for semistructured models is *choice point based* uncertainty [3]. With this strategy, at specific points in the tree a node representing a choice between substrees is inserted. Choosing one child node, and as a result an entire subtree, invalidates the other child nodes. As with the event based strategy, one particular possible world can be selected by making a choice for each choice point. The model presented in this report is based on the choice point strategy.

The amount of work on information integration is enormous. The topic has been studied for several decades already and will remain a research question for many more to come. This is due to the semantics captured in the schema and data. Since semantics is impossible to handle by a machine, human involvement will always be necessary to make final decisions

on semantical issues. A first and already well-studied challenge in information integration is schema matching. The result of this process is a mapping between data sources relating not only element types, but also providing mapping functions that indicate how the *data* should be transformed from one source to the other.

A recent overview of schema integration is given in [1]. The Learning Source Descriptions (LSD) project [15], [16] from the same authors is widely recognized as a big step forward in the schema integration field. In this project, machine learning techniques are applied to effectively use data *instances* for schema matching. Furthermore, it employs a multi-strategy approach where clues obtained from several base learners are combined into a joint similarity estimate by a meta learner.

Although, the schema integration phase is an important and difficult part of the entire integration process, it is not the focus of this report. In this report, we assume that schema integration has already taken place and we focus on the integration issues in the instance data. The one work we found that also uses a probabilistic XML representation in an attempt to integrate XML documents is [17]. Others [18], [19] also use probabilistic databases, but they focus on uncertainty pertaining to the mappings produced by a schema matcher. Nevertheless, this also creates uncertainty about existence of tuples in the integrated database.

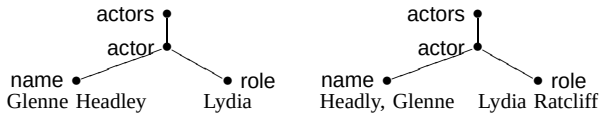
The problem addressed in this report is highly related to entity resolution or record linkage. In [20], a generic approach to entity resolution is presented. The method is generic in the sense that both comparing and merging of records are viewed as black-boxes. In an earlier report [21], the generic approach is used to also manage confidences along with the data. [22] describes an approach for extracting entities from unstructured (textual) sources. An algebra with extraction operators is presented which allows to combine evidences from different sources. For example, for extracting researchers mentioned as PC-members in conference notifications, the researchers are disambiguated by checking in DBLP whether or not two researchers have ever been co-authors, which is a highly domain-specific but very accurate technique for entity resolution for research entities. Closely related is the topic of *entity search* [23]: search engines not geared towards finding web pages, but entities such as persons or telephone numbers.

III. Probabilistic Data Integration

In this section, we present the probabilistic data integration approach used in IMPrECISE, our prototype used in the experiments. Our approach is based on *possible worlds* theory, which is explained next. We explain our representation for uncertain data, the integration algorithm itself and the knowledge rules that are used during integration.

A. Possible worlds

An ordinary database can be considered a representation of (a part of) the real world. In an ideal system, this representation perfectly matches the real world. There are, however, many application areas where this is not the case. An uncertain database allows to store multiple possible representations for a real-world object (rwo) in case it is uncertain which representation is the correct one. In that sense, an uncertain database is a representation of possible worlds. Possible worlds are mutually exclusive, and as a consequence, at most one of the possible worlds actually correctly represents the real world. Each possible world can have an associated probability indicating the confidence with which the database beliefs that the particular possible world is correct w.r.t. the real world.



(a) Fragment TV guide (b) Fragment IMDB

Fig. 4. Example fragments from both sources

Definition 1: Let $\mathcal{D}$ be the universe of database states. We vary D over $\mathcal{D}$. Then, $\mathcal{PD} = \mathbb{P}(\mathbb{R} \times \mathcal{D})$ is the universe of probabilistic database states where $\mathbb{P}$ denotes the power set constructor. We vary PD over $\mathcal{PD}$.

Example 1: To illustrate our approach for representing the uncertainty involved in data integration, let's suppose we like to integrate the example fragments of Fig. 4. The example pertains to an actor list from both sources each containing one actor, namely 'Glenne Headley'. As we have seen in Fig. 3, data about this actor from both sources is conflicting. Suppose it is impossible at integration time to make an absolute decision whether both actor elements refer to the same rwo, i.e., the system estimates that they match with probability α. Based on these facts and assumptions, there are 5 possible worlds. In one world there are two actors (the case that they do not refer to the same rwo). Since there are two possibilities for the name and independently two possibilities for the role, there are four possible worlds for the case that they do refer to the same rwo. Let us furthermore assume that there is no further evidence that either name or role text is correct, i.e., the probability for both is 0.5. See Fig. 5 for an illustration of this set of possible worlds.

Note that we did already use a bit of domain knowledge here, namely, that the elements actors, name, and role can only occur once under their parent elements. Without this DTD knowledge, there would be many more possible worlds.

B. Compact representation of possible worlds

To capture uncertainty in the XML datamodel, we introduce two new node types: probability nodes ($\bigtriangledown$) to represent choice points and possibility nodes ($\circ$) to represent the individual choices. Child nodes of probability nodes are always possibility nodes. Each possibility node has an associated probability, which denotes the confidence of the database that the node and its subtree correctly reflects the real world. Sibling possibility nodes are mutually exclusive, hence probability nodes indicate *choices*. Child nodes of possibility nodes are regular XML

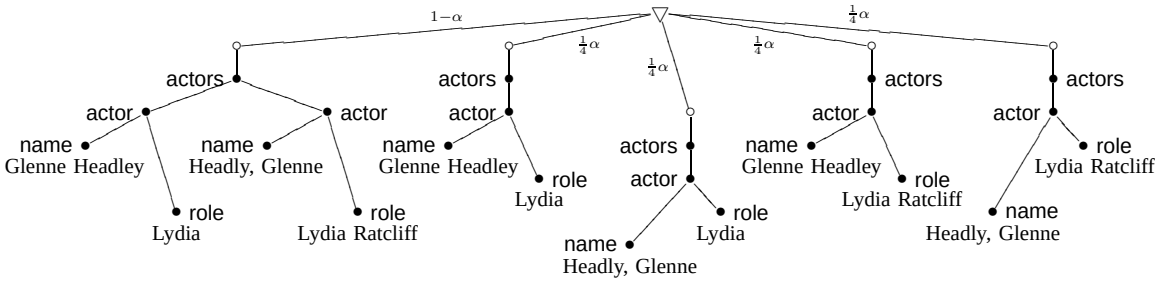


Fig. 5. Possible world representation

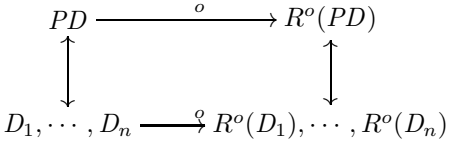


Fig. 7. Commutative diagram (o denotes an operation, R_o the result of o)

nodes (•). Child nodes of regular XML nodes can be either other XML nodes or probability nodes. A formal definition of a slightly stricter probabilistic XML data model is given in [3].

The root node of the tree in Fig. 5 can be seen as one big choice point enumerating all possibilities. We can, however, obtain a more compact representation of the set of possible worlds by *pushing down* the choice points. If we do this for our example we arrive at the tree in Fig. 6. We call this the *compact representation*. An XML representation of this tree is what IMPrECISE stores as its (uncertain) database state. Observe that the individual aspects of uncertainty in the example, i.e., (i) whether or not both actor subtrees refer to the same real world actor, and if so (ii) which name is the correct name, and (iii) which role is the correct role, are now separated and local to the elements with which they are associated.

For smaller examples, the reduction in size is minimal, but with a growing number of worlds and much overlap between worlds, the size benefit is much larger.

C. Querying possible worlds

Querying, as all operations on a probabilistic database, should adhere to possible world theory. The theory dictates that the semantics of an operation on an uncertain database is the same as the combination of the evaluation of the operation on each world independently. Since a possible world is an ordinary database state, it is clear what the semantics is of the evaluation of the operation on one particular world, hence we can deduce the semantics of operations working on the probabilistic database. The commutative diagram in Fig. 7 illustrates this principle.

Definition 2: Let $[\![Q(D)]\!]$ be the semantics (i.e., result) of query Q on regular database D. Then, the set of all possible answers is defined by

$$\mathsf{Ans}_Q(PD) = \{Q(D) \,|\, (p, D) \in PD\}$$

Since the same answer may be produced by several possible worlds, the probability of an answer a is defined by

$$P(a \in Q(PD)) = \sum_{(p,D) \in PD \wedge a \in Q(D)} p$$

The semantics of query Q on a probabilistic database PD can now be defined as

$$[\![Q(PD)]\!] = \{(a, p) \,|\, a \in \mathsf{Ans}_Q(PD) \wedge p = P(a \in Q(PD))\}$$

For querying, the principle means that the semantics of a query is the collection of query results for all possible worlds. A naive implementation following this principle closely is of course very inefficient as it requires the enumeration of all possible worlds. Therefore, the actual implementation differs in that it works directly on the compact representation. How this is accomplished is beyond the scope of this report; it suffices to know that query results conform to this semantics.

For XPath or XQuery the result of an answer is a sequence. Possible answers usually only differ in the existence of one or more elements. For applications, it suffices and is more practicable to just know on a 'per-element' basis with which probability it occurs a certain number of times in the answer.

Definition 3: Let $\overline{\mathsf{Ans}}_Q(PD) = \{v^m \,|\, a \in \mathsf{Ans}_Q(PD) \wedge v^m \in a\}$ be the set of all possible occurrences of an element v where $v^m \in a$ denotes that v occurs m times in a. The probability of v occurring m times in the result is defined as

$$P(v^m \in Q(PD)) = \sum_{(p,D) \in PD \wedge a \in Q(D) \wedge v^m \in a} p$$

'Per-element' *answer style* semantics can now be defined as

$$[\![\{Q(PD)\}]\!] = \{(v^m, p) \,|\, v^m \in \overline{\mathsf{Ans}}_Q(PD) \wedge p = P(v^m \in Q(PD))\}$$

Example 2: Consider the movie document in Fig. 5 and the query Q='Give me the role of the actor named Glenne Headley' expressed for example as the XPath query

```
//actor[name="Glenne Headley"]/role
```

Only 3 of the possible worlds will return a non-empty answer, two containing the role 'Lydia', one containing the role 'Lydia Ratcliff'. Therefore, the result is

`Lydia`^{1}	$1 - \alpha + \frac{1}{4}\alpha$
`Lydia`^{0}	$\frac{3}{4}\alpha$
`Lydia Ratcliff`^{1}	$\frac{1}{4}\alpha$
`Lydia Ratcliff`^{0}	$1 - \alpha + \frac{3}{4}\alpha$

Note that the probabilities for each element v correctly add up to one.

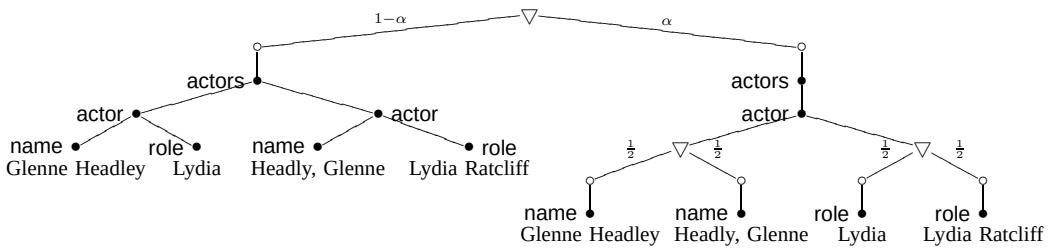


Fig. 6. Compact representation

$$\mathsf{matches}(\mathsf{A},\mathsf{B}) = \{\langle \mathsf{a}{=}c_a, \mathsf{b}{=}c_b, \mathsf{e}{=}est\rangle | c_a \in \mathsf{A} \wedge c_b \in \mathsf{B} \wedge est = \mathsf{Oracle}(c_a, c_b) \wedge est > 0\}$$

$$\mathsf{growCluster}(\mathsf{C},\mathcal{M}) = \text{IF } \mathsf{GC} = \mathsf{C} \text{ THEN } \mathsf{C} \text{ ELSE } \mathsf{growCluster}(\mathsf{GC},\mathcal{M}) \text{ WHERE } \mathsf{GC} = \{m \in \mathcal{M} | \exists c \in \mathsf{C} : m{\cdot}\mathsf{a} = c{\cdot}\mathsf{a} \vee m{\cdot}\mathsf{b} = c{\cdot}\mathsf{b}\}$$

$$\mathsf{cluster}(\mathcal{M}) = \text{Take arbitrary } m \in \mathcal{M};\ \mathsf{C} := \mathsf{growCluster}(\{m\});\ \mathsf{R} := \mathcal{M} \backslash \mathsf{C};\ \text{RETURN IF } \mathsf{R} = \emptyset \text{ THEN } \{\mathsf{C}\} \text{ ELSE } \{\mathsf{C}\} \cup \mathsf{cluster}(\mathsf{R})$$

$$\mathsf{combinations}(\mathsf{C}) = \{S \cup \{a \in \mathsf{A} | \neg\exists m \in \mathsf{C} : m{\cdot}\mathsf{a} = a\} \cup \{b \in \mathsf{B} | \neg\exists m \in \mathsf{C} : m{\cdot}\mathsf{b} = b\} \mid S \subseteq \mathsf{C} \wedge \forall m_1, m_2 \in S : (m_1{\cdot}\mathsf{a} = m_2{\cdot}\mathsf{a} \vee m_1{\cdot}\mathsf{b} = m_2{\cdot}\mathsf{b}) \Rightarrow m_1 = m_2\}$$
$$\text{WHERE } \mathsf{A} = \{m{\cdot}\mathsf{a} | m \in \mathsf{C}\}, \mathsf{B} = \{m{\cdot}\mathsf{b} | m \in \mathsf{C}\}$$

```
integrate(a, b) =
  IF a and b are text nodes
  THEN IF a = b THEN RETURN a
       ELSE RETURN ⟨prob⟩⟨poss prob=0.5⟩a⟨/poss⟩
                   ⟨poss prob=0.5⟩b⟨/poss⟩⟨/prob⟩
  /* Matching phase */
  M := matches(a/child::*, b/child::*)
  cert := {m ∈ M | m·e = 1}
  possible := {m ∈ M | ¬∃m' ∈ cert : m'·a = m·a ∨ m'·b = m·b}
  /* Clustering phase */
  clusters := cluster(possible)
  /* Result construction phase */
  result := NEW ELEMENT "same name as a and b"
  FOREACH e ∈ a/child::* WHERE ¬∃m ∈ M : m·a = c
    DO INSERT e INTO result
  IF full integration (not only enrichment)
  THEN FOREACH e ∈ b/child::* WHERE ¬∃m ∈ M : m·b = c
         DO INSERT e INTO result
  FOREACH m ∈ cert DO INSERT integrate(m·a, m·b) INTO result
  FOREACH C ∈ clusters DO
    prob := NEW ELEMENT "prob"; INSERT prob INTO result
    FOREACH comb ∈ combinations(C) DO
      poss := NEW ELEMENT "poss"; INSERT poss INTO prob
      FOREACH m ∈ comb DO
        IF m is a pair
        THEN INSERT integrate(m·a, m·b) INTO poss
        ELSE INSERT m INTO poss
  RETURN result
```

Fig. 8. Integration algorithm

D. Integration Algorithm

The algorithm for our probabilistic data integration approach is given in Fig. 8. Note that this algorithm is an improvement of the one presented in [3] in that it results in a more compact integrated result. In this section, we explain the algorithm in general terms and illustrate it by explaining its behaviour for the example elements given in Fig. 4.

The input to the algorithm are two XML elements a and b. We assume that their schemas are aligned. Integration is performed recursively on a level-by-level basis. This can be seen in the calls to integrate within the integrate function itself. Its arguments are always m·a and m·b which are children of the input a and b respectively. The algorithm first checks if the input nodes are text nodes, because then it can immediately return a result. Each recursion step is divided into three phases.

Matching phase: First we need to find possible matches between the children of the two input elements. Matching is performed by calling a function Oracle for each possible pair of children. It estimates the similarity of two given elements based on a set of knowledge rules. We discuss these knowledge rules in Section III-E. The Oracle returns an estimate of 0 when the given two elements have a different element name. In this way, it does not matter for our algorithm if the children it is matching are all of the same type (e.g., the `actor` children of the `actors` element) or of mixed types (e.g., the `name` and `role` children of the `actor` element).

Some matches have similarity 1, which means that the Oracle is certain that they match. We distinguish those in cert from the rest, because if there is a certain match for a child, then we need not consider other matches of lower similarity for this element. The remaining matches in possible represent all entities for which we cannot make an absolute decision. Therefore, for each of these we need to consider both the possibility that they refer to the same rwo and the possibility that they do not.

If invoked on the `actors` elements, the Oracle estimates the similarity between the two `actor` elements and returns α. Because $\alpha \neq 1$, its match ends up in possible and cert remains empty. In a deeper recursion step, integrate is invoked on both `actor` elements. Here the Oracle produces 4 estimations: two are 0 because name $\neq$ role and two are 0.5, because we assumed that there was no further evidence that either name or role is correct. Both matches end up in possible and cert remains empty.

Clustering phase: The set of remaining uncertain matches usually contains small clusters of similar entities. For example, movies from IMDB similar to “The Hustler” are usually not similar to “Stage Beauty”. It is beneficial to find these clusters to get a compact representation of the end result.

In the `actors` recursion step, there was only one match, so we obtain one cluster with the match. In the `actor` recursion step, there were two matches which do not overlap, so we obtain two clusters with each one match.

The improvement of the algorithm of Fig. 8 over the one presented in [3] lies in this additional clustering phase. In [3] we observed that the number of combinations increases drastically with rising number of possible matches. For example, if both a and b have 5 children and they all possibly match, then we end up with 1546 combinations, hence 1546 possibility nodes. As we argued above, many small clusters can usually be found. As a consequence, we do not produce one probability node with a huge number of possibilities, but several probability nodes, one for each cluster, with each just a few possibilities.

Result construction phase: Now we are ready to construct the result. We create one element which represents the integrate result. All children of a that are not matched, are added to the result. The same applies to children of b, but only when we do a full integration and not just enrichtment. For our TV guide example, we are not interested in a full integration of movies, but only in enrichment of the movies of the TV guide with data *only on these movies* from IMDB. In deeper recursion steps, we are interested in full integration of all elements. Subsequently, all certain matches are integrated and added as children.

For each cluster of matches we are faced with several possibilities. Therefore, we create a `prob` element (and add it to the result) for each cluster. A cluster contains a set of matches that can either be correct (i.e., the elements indeed refer to the same rwo in reality) or incorrect. The choices for either correct or incorrect for each match are independent. The only restriction is that if an element a is matched to an element b, then a cannot at the same time be matched with another element $b' \neq b$, because in the original data source, b and b' were individual elements representing different entities. The combinations function determines all possible combinations of correct/incorrect choices for the matches of a cluster. Correct choices represent possibilities where we assume the elements refer to the same rwo, so we integrate them. Incorrect choices represent possibilities where we assume the elements do not refer to the same rwo, so they end up as individual elements in the combination and eventually as individual children of the `poss` element. To keep the presentation of the algorithm in Fig. 8 clear, we have omitted the calculation of the probability of a combination. It basically is the product of the probabilities of the correct/incorrect choices for the matches that led to the particular combination.

In the `actors` recursion step, we had one cluster with one match. There are only two possible combinations: $\{a, b\}$ and $\{a/b\}$. Therefore, we construct a `prob` element with two `poss` elements. This is the root node in Fig. 6. In the `actor` recursion step, there were two clusters with each one match. This produces the other two probability nodes in Fig. 6.

E. The Oracle and its Knowledge Rules

The invokation of the Oracle is the only point in the data integration algorithm where a semantical decision is being taken. In this way, we have strictly separated the integration *mechanism* from the integration *intelligence*. The Oracle obtains its intelligence from knowledge rules. We distinguish between *generic* and *domain-specific* knowledge rules. The current set of generic rules in IMPrECISE is discussed in Section III-F. Domain-specific rules are defined by the developer to enable the The Oracle to produce good estimations by taking into account the specificities of the application domain. The role and effect of the knowledge rules in the The Oracle can be understood best by imagining a few hypothetical 'extreme' Oracles:

- *Omniscient Oracle*. This Oracle has perfect knowledge, hence can always give a correct absolute estimate of 0 or 1 for each pair of elements. This is of course a hypothetical situation, but if we would be able to define all required knowledge rules for it, then cert would contain all positive matches and possible would always be empty. Therefore, we obtain only clusters of one match, so no probability and possibility nodes are constructed, and the algorithm produces an integration result without uncertainty.[4]
- *Ignorant doubtful Oracle*. This Oracle has no knowledge rules at all. For differently named elements it produces an estimate of 0; otherwise it produces 0.5 effectively stating that it is always fully in doubt. With this Oracle, all pairs of children would match, hence cert is empty and possible contains a cartesian product of all children. These matches all form one cluster which produces a huge number of possible combinations. Consequently, with this Oracle we get a maximally exploded integration result. Note that the algorithm does work without any knowledge. The only drawback is data explosion, because it considers all (non-sensical) possibilities.
- *Ignorant decisive Oracle*. This Oracle also has no knowledge rules, but stubbornly estimates all matches with 0. This is an interesting case, because this leaves cert and possible empty, hence the integration result contains a union of the children of both elements. This is what is frequently done in practice to get an initial integrated data set, which is subsequently cleaned. Entity resolution happens in the data cleaning phase. Note that with data cleaning solutions, an absolute choice is always made for two data items to be 'duplicates' or not.

The Ignorant doubtful Oracle is what we start with. To obtain an integration that is good enough for a particular application, we add as many knowledge rules as needed to obtain an integration result that balances size of integration result with query answer quality well. IMPrECISE contains a basic set of generic rules, so the developer needs to only define domain-specific rules that partially override the generic rules. We present the generic rules below. The domain-specific rules for our example application are given in Section V-A.

[4] Strictly speaking this is not true. In the presented algorithm, the Oracle does not make decisions about which value to take if data between sources conflicts. Text nodes always receive a 50/50 decision, so probability and possibility nodes are constructed for text nodes.

F. Generic rules

Since we do not focus in our work on similarity measures, but on how to cope with their inherent imperfections, we have implemented a simple edit distance measure that regularly gives too little evidence for an absolute decision. Unless overridden by other rules, two elements are compared based on *inverse relative edit distance* (red) of their string values:

$$\mathsf{red} \;=\; 1 - \frac{\mathsf{editdistance}(a,b)}{\mathsf{max}(\mathsf{length}(a),\mathsf{length}(b))}$$

Because of different naming conventions for names of things and people, e.g., "Kal Penn" and "Penn, Kal" in Fig. 3, we count the switch around the comma in the edit distance as two edits (one delete and one insert). To be able to force absolute decisions, there are two thresholds:

1) A minimum threshold: if the red is below this threshold, The Oracle concludes with certainty that the two elements *do not* refer to the same rwo.
2) A maximum threshold: if the red is above this threshold, The Oracle concludes with certainty that the two elements *do* refer to the same rwo.

For any red between the two thresholds, the system considers the two possibilities seperately: the data items either refer to two different rwos or they refer to the same rwo. The red is taken as probability for the case that they do refer to the same rwo. The default minimum and maximum thresholds are 0.2 and 1.0 respectively (i.e., there is no maximum threshold). The system can be configured with different thresholds for certain paths, but this is domain-specific (see Section V-A for the configurations in our experiments).

Furthermore, elements with different element names cannot possibly refer to the same rwo, because the schemas are assumed to be aligned. Also, constraints on the schema imposed by a DTD are taken into account, e.g., if a certain element can only occur once and both data source contain the element, then they must refer to the same rwo, because otherwise we end up with two elements in the result which is against the DTD.

IV. Measuring Quality

Querying uncertain data results in answers containing uncertainty. Therefore, an answer is not correct or incorrect in the traditional sense of a database query. We need a more subtle notion of answer quality. Also, the quality of the integrated database itself can be quantified, for example, a higher amount of uncertainty in a database suggests a lower quality. Moreover, lower average answer quality also suggests a lower integration quality.

A. A measure for the amount of uncertainty

An often used measure for the amount of uncertainty in a database is the number of possible worlds it represents. However, this measure exaggerates the perceived amount of uncertainty, because it grows exponentially with linearly growing independent possibilities. Additionally, we would like all measures to be in a predefined range, i.e. between 0 and 1. We defined two measures to quantify the quality of the integration result [6].

- *Decisiveness* is an indication how easy query answers can be interpreted, i.e., how decisive the associated probabilities are, hence how easy an application can filter out the 'probably good' ones.
- *Density* is an indication of the ratio between the amount of uncertain and certain data.

For the purpose of this report, we only use density as a measure. It is based on the average number of alternatives per choice point:

$$\text{Dens} = 1 - \frac{1}{N_{cp}} \sum_{j=1}^{N_{cp}} \frac{1}{N_{poss,j}}$$

where N_{cp} is the number of choice points in the document and $N_{poss,j}$ is the number of possibilities at choice point j. Dens is 0 for a databases that contains no uncertainty. Dens decreases if there is more certain data in the database for the same amount of uncertain data. If all choice points contain n alternatives, Dens is $(1-\frac{1}{n})$, which approaches 1 with growing n. The uncertainty density is independent of the probabilities in the database.

B. Answer Quality

In the possible world approach, an uncertain answer represents a set of possible answers each with an associated probability. In some systems, it is possible to work with alternatives without probabilities, but these can be considered as equally likely alternatives, hence with uniformly distributed probabilities.

The set of possible answers ranked according to probability has much in common with the result of an information retrieval query. We therefore base our answer quality measure on precision and recall [25]. We adapt these notions, however, by taking into account the probability with which a system claims a query answer to be true. The intuition behind it is that the quality of a correct answer is higher if the system dares to claim that it is correct with a higher probability. Analogously, incorrect answers with a high probability are worse than incorrect answers with a low probability.

Precision and recall are traditionally computed by looking at the presence of correct and incorrect answers. Since XPath and XQuery answers are sequences and we ignore order, we define H to be the *multiset* of correct answers to a query (as determined by a human), A the multiset of answers produced by the system, and C the intersection of the two, i.e., the multiset of correct answers produced by the system (see Fig. 9).

Since we have an answer per possible world, it is logical to take the *expected value* of the precision and recall. This naturally takes into account the probabilities associated with answers. If we define A as $[\![Q(PD)]\!]$, C as $\{(v^m, p) \in [\![Q(PD)]\!] \mid v^m \in H\}$, and $|S| = \sum_{(v^m,p)\in S} p \times m$ the

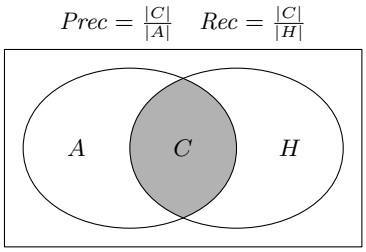


Fig. 9. Precision and recall.

expected cardinality of multiset with probabilities S, then

$$\text{Precision} = \frac{|C|}{|A|} = \sum_{(p,D)\in PD} p \times |\text{Precision}_{Q(D)}|$$

$$\text{Recall} = \frac{|C|}{|H|} = \sum_{(p,D)\in PD} p \times |\text{Recall}_{Q(D)}|$$

For example, suppose the answer to the query "Give me all movies aired on CMAX on June 5" is "The Namesake" and "Namesake, The". We consider textual variations of the same semantical concept as the same answer. If the system returns this single answer (which is correct), but with a confidence of 90%, then precision and recall are both 90%. If, however, it also gives some other (incorrect) movie with a confidence of 20%, precision drops to 82% and recall stays 90%.

V. Experiments

A. Experimental set-up

The aim of the experiments is twofold: to investigate the sensitivity of rule definition and threshold tuning on integration quality and to use this insight as evidence that our probabilistic integration approach indeed significantly reduces development effort. By defining knowledge rules and thresholds, a developer aims to get rid of as many incorrect possibilities as possible to reduce the uncertainty and consequently the size of the integration result, but at the same time to not run too much risk in ruling out correct possibilities. Therefore, the following factors play a role in the experiments.

- Knowledge rules.
- Thresholds.
- Amount of uncertainty in the integration result.
- Size of the integration result.
- Quality of answers to certain queries.

The domain of the experiment concerns movies. In the integration, only the entities 'movie', 'actor', and 'genre' are present in *both* data sources, hence play a role in the entity resolution for this application. Therefore, the domain knowledge added to the system focuses on these entities. Fig. 3 shows which other elements accompany these entities.

The knowledge rules The domain-specific knowledge rules we defined are the following.

1) **DTD rule**: The DTD prescribes that certain elements occur only once among others title, year, and within the actor-element: name and role.
2) **MovieTitle rule** (MT-rule): The probability of two movie elements referring to the same rwo is based solely on the similarity of their titles.[5] To obtain candidate matches for a particular movie title, we search for those movie titles that have the least edit distance. If the best edit distance is not zero, we expand the candidates with all titles within a margin of n additional edit distance (we vary margin n in the experiments).
3) **UniqueRole rule** (UR-rule): If the role-child of two actor-elements is exactly the same and the role is unique for the movie in both data sources, then The Oracle concludes with certainty that it is the same actor. If the role is not the same, then the probability is computed by multiplying the red's of the name and role children. As a consequence, if the role is the same but not unique or the role is missing, then the decision of The Oracle is based on the actor name only (we vary the minimum threshold on actors in the experiments).

Note that the only development effort needed for the TV guide application to safely and sufficiently reduce the number of incorrect matches for entity resolution of movies and actors, hence to be able to meaningfully use the integrated data set, is the definition of the DTD, the definition of the above two rules, and to tune the two given thresholds.

Data set As explained in Section I-B, we use for our experiments the application context of a TV guide application. As representative data, we took the 'Movies on TV' page of `www.tvguide.com`. It contains a selection of movies that are aired that day. Fig. 10 shows two days worth of 'Today's Picks'.

4 June 2008

Bobby
National Lampoon's Vacation
Monsoon Wedding
Escape from New York
Reservoir Dogs

5 June 2008

Notes on a Scandal
The Namesake
District B13
The Howling
American Psycho

Fig. 10. Movies in our TV guide data source

We integrate with data from the IMDB movie database (version 3.24) publicly available from various FTP-sites. We converted the data to XML with a schema aligned with the TV guide data. We excluded documentaries and adult movies to leave a realistic and substantial data set of 243,856 movies.

We chose June 4 and 5 for the TV guide data set, because of the interesting properties below. Data from other days show similar behaviour.

- The movies 'National Lampoon's Vacation' (1983) and 'District B13' (2006) do not appear in the IMDB data set. Especially for the latter, many movies with similar titles end up as possible matches in the integration result. Best matches are 'National Lampoon's Bag Boy' (2008) at 77% and 'Bistrice '63" (1964) at 58% respectively.
- For movies 'Bobby' (2006) and 'Reservoir Dogs' (1992) there are 3 and 1 other movies, respectively, with exactly the same title, hence which the MT-rule cannot distinguish.

[5]This obviously is not the best possible movie matching rule, but that is not what we are after. We want to show that even using simple imperfect rules like these, we can quickly obtain a meaningfully usable integration result.

Group	#Queries	Query pattern
A	18	For each leaf element, an absolute path to the leaf
B	5	`//movie[.//genre='X']/title`
C	5	`//actor[.//role='X']/name`
D	5	`//movie[.//role='X']/title`
E	5	`//movie[.//keyword='X']/title`
F	5	`//movie[.//location='X']/title`
Total	43	
G	12	All actor queries, i.e., groups C and D and the two absolute paths for role and name from A

Fig. 11. Groups of experimental queries (for X we chose 5 different values that focus on different possible mistakes).

- The movies 'The Namesake' and 'The Howling' are written as 'Namesake, The' and 'Howling, The' in IMDB. Our red measure estimates these similarities as 83% and 82% respectively. Since these matches are not perfect, the system starts to confuse these movies with other movies at certain margins. Most similar titles for 'The Namesake' are 'Passage, The', 'Caretacker, The' and others at 50%; most similar titles for 'The Howling' are 'Hazing, The', 'Calling, The' and others at 55%.
- Consequently, the June 5 TV guide data is much more difficult to enrich than the June 4 TV guide data.

Factors We investigate threshold tuning sensitivity by varying the margin for the MT-rule and the minimum relative edit distance threshold for actors that do not hit the UR-rule (i.e., no matching role or the role is not unique). Varying the margin of the MT-rule has no effect on movies that have an exact match. For 'The Namesake' and 'The Howling', only an imperfect match exists, but the best matching titles are also the correct matches, so increasing the margin means that the probability mass going to the correct matches decreases. For movies that do not have a match, increasing the margin only increases the confusement.

Note that when an incorrect match for a movie is considered as a realistic possibility, the actor lists of both movies are integrated as well possibly producing some correct matches, but more likely producing a union of the actor lists when it determines that all actors are different people.

At a high threshold for actors, correct matches are often not found. Fig. 3 shows many cases where for the same real-world actors, names and roles differ substantially in their data of both sources. Decreasing the threshold will increase the likelihood that they are matched, but also the likelihood for incorrect matches.

To investigate the effect of rule definition, we switch off the DTD-rules and the UR-rule respectively and compare the quality of the answers to those where the rules were switched on. We also investigate the change in effect of threshold tuning.

Queries We assess the quality of our integration result by taking the average precision and recall for different subsets of 43 queries (see Fig. 11 for a description of the queries). Groups B through F are chosen based on the following aspects:

- Group B is independent of entity matching problems in actors.
- Group C focuses on actors only, but movie entity matching mistakes may affect the group when actors and roles from incorrectly matched movies are taken into account.
- Group D navigates from actor to movie.
- Group E has a predicate on (enriched) data from IMDB.
- Group F has a predicate on data from the TV guide.
- Group G are all queries that involve actor information; it is a subset of 12 queries.

B. Results

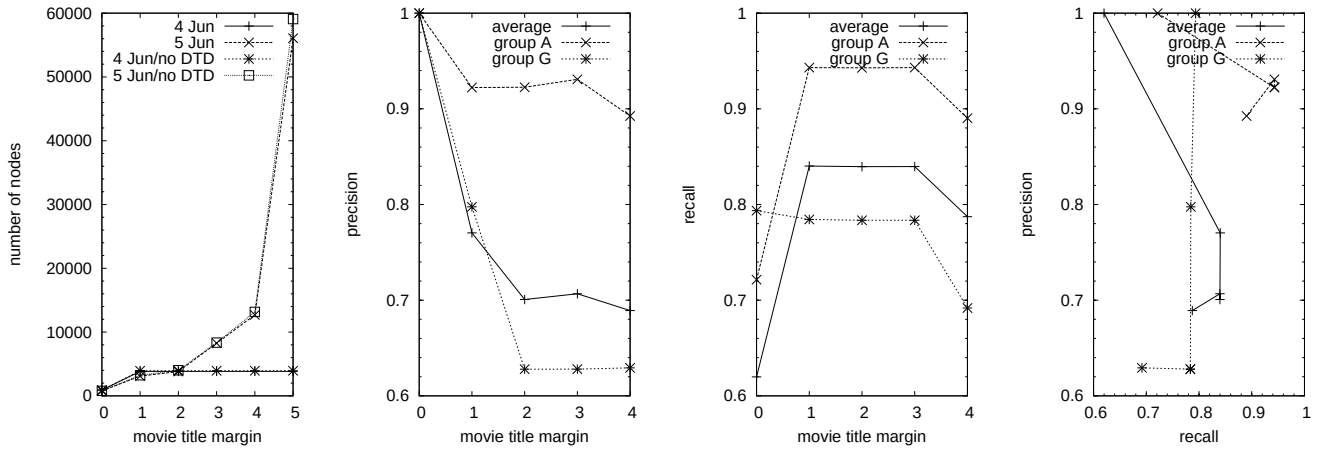
In this section, we discuss the various effects of knowledge rule definition and threshold tuning on the integration quality. We first look at threshold tuning by investigating the effect of varying the movie title margin and actor threshold. We then focus on rule definition by switching of the DTD and the UniqueRole rules.

Effect of movie title margin

Fig. 12 shows the results of varying the movie title margin where we keep the actor threshold at 0.4. The expectation is that the size of the integration result increases with a larger margin as more matches are considered possible. This holds for 5 June, but not for 4 June (see Fig. 12(a)). The reason is that 4 of its movies have an exact match, so any lower quality matches are not considered. 'National Lampoon's Vacation' does not have a correct match, but even at margin 5 no other matches are found. Hence the result at margins 1 through 4 is the same.

June 5 is the more interesting case here. The expectation for answer quality is that with the lowest margin, some correct matches may be missed. Not taking into account effects on probabilities, increasing the margin would at first increase precision and recall. Going beyond the margin, however, where all correct matches are found, would decrease precision and recall would stay the same. This expectation holds for recall (see Fig. 12(c)): at margin 1 the correct matches for all movies are found. Gradually diminishing quality at higher margins can often be explained by effects on probabilities. Even though the extra possibilities that are considered at higher margins have low probabilities, they do 'consume' some of the probability mass, hence correct answers receive slightly lower probabilities. At margin 0, no movies are matched, hence no enrichment and the integration result is the same as the original TV guide data. Since that data is correct and all answers have probability 1, precision is 1 (see Fig. 12(b)). The 'wobble' at margin 3 can be explained by two counteracting effects: (a) probabilities of answers become lower (lower $|C|$ and $|A|$), (b) more (incorrect) answers end up in the result (higher $|A|$).

It depends on the application whether precision is more important or recall. A P/R-curve is useful for determining where the optimal setting for the margin lies (see Fig. 12(d)). Since we are interested in enriching the TV guide data set, we are after good recall, perhaps at the expense of some precision. For all interesting groups of queries we found that at margin 1 we already have the recall we want. Note that a developer does not have this curve at his/her disposal. If he sets the margin



(a) Margin affecting size (b) Margin affecting precision (5 Jun) (c) Margin affecting recall (5 Jun) (d) Margin recall P/R curve (5 Jun)

Fig. 12. Effect of movie title margin (actor threshold 0.4)

at a safe level, e.g., 2 or 3, we see that precision and recall do not change much, hence answer quality is rather insensitive to a moderately safe setting of the margin.

Effect of actor threshold

The effect of varying the actor threshold is expected to be exactly opposite to movie title margin, because a higher threshold means less possibilities to be considered while a higher margin means more. A difference between the actor and movie entities, is that an actor match *depends* on a movie match, because only the actors of possibly matching movies are matched. We do, however, expect similar general behaviour at different margins when varying the actor threshold.

The effect of actor threshold on size of integration result is as expected (see Fig. 13(a)). We furthermore see a more rapid increase for margins 1 and 2 for very low thresholds. At these thresholds the system starts to confuse actors on a massive scale.

As observed before, at margin 0 no movies are matched, nothing is enriched, hence we end up with only the TV guide data. Therefore, precision is 1 (see Fig.s 12(b) and 13(b)). The recall of about 0.8 for actor queries (see Fig.s 12(c) and 13(c)) can be interpreted as the TV guide holding roughly 80% of the correct actors.

At margins higher than 1, we see the same behaviour: Precision goes up (although only slightly), at threshold 0.8 it is at its maximum, and then it gradually decreases. At threshold 1, the system consideres all actors to be different people. Because we specified to be interested in an *integration* of actors from both sources instead of only enrichment (as with movies), mistakenly not matching two actors means that they appear twice in the resulting actor list. For recall this doesn't matter ($|C|$ does not differ for 1 or 2 occurrences of an actor in the result when only one is correct), so only effects of additional possibilities consuming probability mass are visible in recall. Precision, however, is affected by mistakenly not matching actors, because $|A|$ is higher.

Nevertheless, we see that the answer quality is less sensitive to the actor threshold than to the movie title margin. In Fig. 13(d), we can furthermore observe that absolute path queries (Group A) are very insensitive, because it affects only 2 of the 18 queries. We see more effect on the actor queries, but even here the movie title margin effects are bigger.

Effect of DTD rules

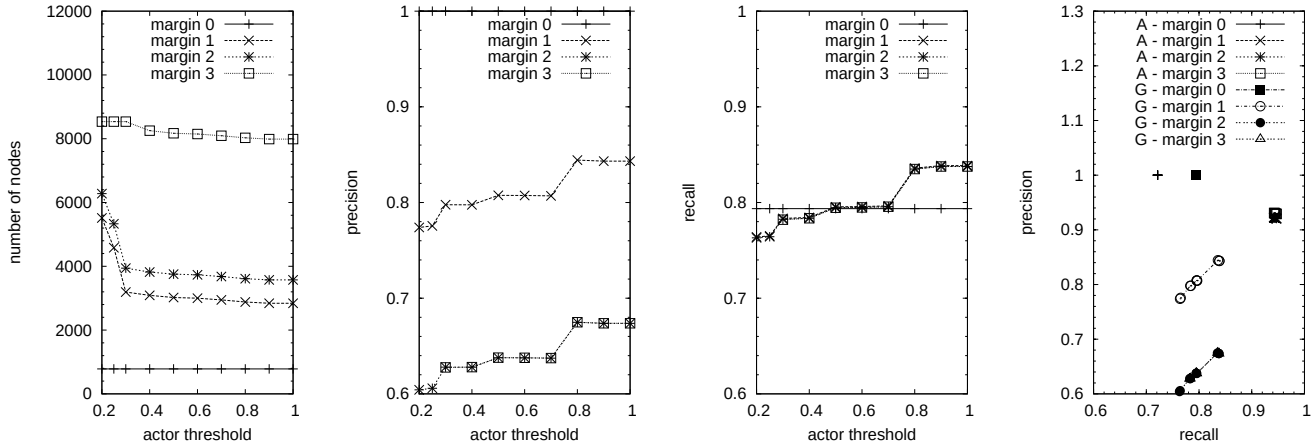
Fig. 12(a) also shows the effect on size of integration result if we switch off some of the DTD rules. As an effect, the system no longer holds the domain information that title, year, genres, directors, plots, and locations can only occur once. The effect on the size is minor, so is the effect on quality (not shown in the graphs to maintain readability).

Effect of UR-rule

Fig. 14 compares the size, precision and recall for situations with and without the UniqueRole-rule. The rule apparently is very effective in protecting against an explosion in the size of the integration when the threshold is set too low. It effectively avoids reliance on string matching for those actors that can be perfectly matched based on their unique role. At higher thresholds, it consistently produces higher precision irrespective of the data involved (4 or 5 June concern distinct sets of movies). We haven't found a good explanation yet for why recall is lower when the UR-rule is switched on.

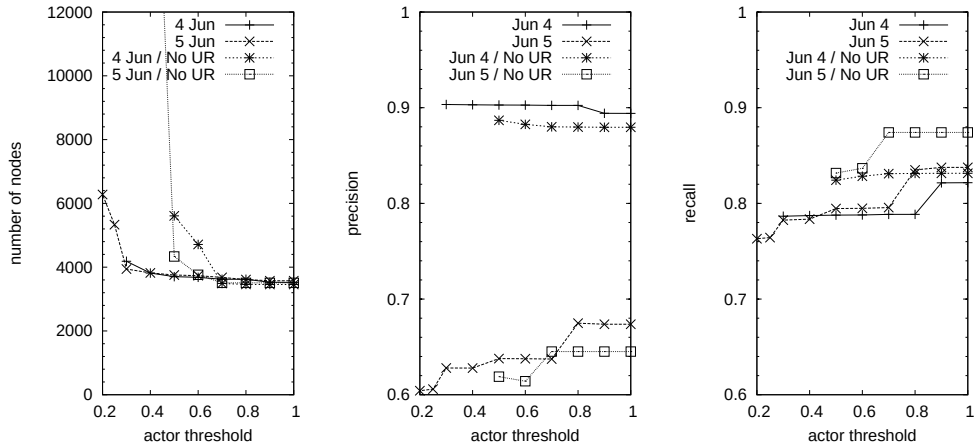
Evidence for development effort reduction

The first observation that we like to emphasize, is that for each entity, in our case movie and actor, we only had to specify *one* domain-specific rule to provide enough knowledge to the system for roughly resolving the entities. For the rest the system relies on generic string matching and appropriate thresholds. The MT-rule is far from perfect, it only relies on movie titles for identification of movies, yet it is already able to achieve average precision and recall around 0.94 and 0.7, respectively, with a safe margin setting of 2 or 3 (see Fig. 12(b) and 12(c)). This often is 'good enough' for an initial integration knowing that answers are accompanied with probabilities with which an application can easily pick the



(a) Threshold affecting size (b) Threshold affecting precision (c) Threshold affecting recall (d) Threshold recall P/R curve

Fig. 13. Effect of actor threshold (Group G; 5 Jun)



(a) Threshold affecting size (b) Threshold affecting precision (c) Threshold affecting recall

Fig. 14. Effect of UniqueRole-rule (margin 2)

most probably ones, and that feedback during use of the application will gradually improve the integration quality. The UR-rule makes an absolute decision for a large subset of actors, which effectively reduces uncertainty significantly. A likely subsequent action of the developer would be to include the year in movie identification. We refrained from doing that to show that even with such a preliminary rule, already a meaningfully usable integrated data set is obtained.

The second important observation is that precision and recall is rather insensitive to moderate variations in threshold settings especially on side of a 'safe setting'. With a safe setting we mean a setting that is not likely to miss any correct matches at the expense of considering quite some incorrect matches as realistic possibilities (high movie title margin or low actor threshold). From this we conclude that a developer need not spend much time on threshold tuning. It suffices for him to just inspect matching results for a sample of data and then to choose a somewhat safer value.

In this way, our approach allows a developer who needs to integrate two data sources, to only focus on resolving the proverbial 90% of easy cases and save 90% of the work he would otherwise have to spend on resolving the 10% of hard cases. The only thing we need to subtract from the 90% development effort reduction is effort on rule definition and threshold tuning. For this, we just argued that few rules and rough settings suffice.

Obviously, the remaining 10% of unresolved entities remain in the integrated data as uncertainty. As we explained in the introduction and [2], we can defer their resolution to the end users of the integrated data using user feedback. During use the integration quality will gradually improve in this way. But most importantly, the application can be already be meaningfully used without resolving the 10% of hard cases, hence 'time-to-market' is reduced significantly.

VI. Conclusions and Future Work

Entity resolution is a challenging problem in many applications that require different data sources to be integrated. An often-used rule of thumb states that about 90% of the development effort is devoted to solving the remaining 10%

hard cases. Our probabilistic integration approach aims at reducing the development effort needed for such applications by allowing some semantic uncertainty to remain in the data, while still being able to meaningfully use this data. The developer is only required to provide a few knowledge rules and rough estimations for thresholds.

The main contribution of this report is a thorough experimental investigation of the effects and sensitivity of rule definition and threshold tuning on the integration quality. Our experiments are based on an application for which movie data from a TV guide is enriched with data from IMDB. Essential to the enrichment is resolution of the entities movie and actor that occur in both data sources. We selected two days worth of data from the TV movie guide (more data from other days shows similar behaviour) and a realistic and substantial data set of 243,856 movies from IMDB. By measuring the expected value of average precision and recall for 43 queries, we determined the quality of the integrated data for various threshold settings and rule configurations. The results convincingly prove that our approach indeed reduces development effort — and not merely shifts the effort to rule definition and threshold tuning — by showing that setting rough safe thresholds and defining only a few rules suffices to produce a 'good enough' integration that can be meaningfully used.

Note that this pertains to the quality of the *initial* data integration. In our probabilistic integration approach, the knowledge rules are only needed to quickly produce an integrated data set of manageable size that can be meaningfully used by an application. Feedback obtained from a user interacting with an application gradually improves the integration [2].

Several aspects of our approach need further research. We plan to continue investigating the interplay between (formulations of) knowledge rules, properties of the integrated data, and quality of query answers, e.g., in other real-life application domains. More insight into these aspects is expected to be important for devising effective tools to support developers in using our approach in practice. For the latter, further research is also needed on how to seamlessly embed feedback giving functionality in GUIs, and on improving the scalability of the algorithms for integration, querying and user feedback in the underlying probabilistic database. Furthermore, our integration approach is expected to benefit from being combined with fuzzy querying techniques. Combining our approach with schema matching has the potential of further decreasing development effort for data integration. We also believe that our probabilistic integration approach can be adapted to a (probabilistic) *relational* database setting (e.g., Trio [9]). Finally, effectively measuring answer and integration quality remains a hard task, so we intend to examine other means for measuring this as well.

REFERENCES

[1] A. Doan and A. Halevy, "Semantic integration research in the database community: A brief survey," *AI Magazine*, 2005.

[2] A. de Keijzer and M. van Keulen, "User feedback in probabilistic integration," in *Second Int'l Workshop on Flexible Database and Information System Technology (FlexDBIST), Regensburg, Germany*. Los Alamitos: IEEE Computer Society Press, Sept. 2007, pp. 377–381.

[3] M. v. Keulen, A. d. Keijzer, and W. Alink, "A probabilistic XML approach to data integration," in *Proceedings of the 21st Int'l Conf. on Data Engineering (ICDE), 5-8 April 2005, Tokyo, Japan*. IEEE Computer Society, 2005, pp. 459–470. [Online]. Available: http://db.cs.utwente.nl/Publications/PaperStore/db-utwente-41064AD3.pdf

[4] A. de Keijzer and M. van Keulen, "IMPrECISE: Good-is-good-enough data integration," in *Proceedings of the 24th Int'l Conf. on Data Engineering (ICDE), 7-12 April 2008, Cancun, Mexico*, Apr. 2008.

[5] A. de Keijzer, M. van Keulen, and Y. Li, "Taming data explosion in probabilistic information integration," in *On-line Pre-Proceedings of IIDB, Munich, Germany*, 2006, pp. 82–86, position paper. http://ssi.umh.ac.be/iidb.

[6] A. de Keijzer and M. van Keulen, "Quality measures in uncertain data management," in *Proceedings of the 1st Int'l Conf. on Scalable Uncertainty Management (SUM), Washington, DC, USA*, ser. LNCS, vol. 4772, 2007, pp. 104–115.

[7] D. Barbará, H. Garcia-Molina, and D. Porter, "A probabilistic relational data model," in *Proceedings of the Int'l Conf. on Extending Database Technology (EDBT) Venice, Italy, March 26-30, 1990*, ser. LNCS, vol. 416. Springer, 1990, pp. 60–74, iSBN 3-540-52291-3.

[8] L. Lakshmanan, N. Leone, R. Ross, and V. Subrahmanian, "ProbView: a flexible probabilistic database system," *ACM Transactions on Database Systems (TODS)*, vol. 22, no. 3, pp. 419–469, 1997. [Online]. Available: citeseer.nj.nec.com/article/lakshmanan97probview.html

[9] O. Benjelloun, A. D. Sarma, C. Hayworth, and J. Widom, "An introduction to ULDBs and the Trio system." *IEEE Data Engineering Bulletin*, vol. 29, no. 1, pp. 5–16, 2006.

[10] J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu, "MYSTIQ: a system for finding more answers by using probabilities," in *Proceedings of the 2005 ACM SIGMOD Int'l Conf. on Management of data, Baltimore, Maryland, USA*, 2005, pp. 891–893.

[11] R. Cheng, S. Singh, and S. Prabhakar, "U-DBMS: A database system for managing constantly-evolving data," in *Proceedings of the 31st Int'l Conf. on Very Large Data Bases (VLDB), Trondheim, Norway*, 2005, pp. 1271–1274.

[12] L. Antova, C. Koch, and D. Olteanu, "MayBMS: Managing incomplete information with probabilistic world-set decompositions," in *Proceedings of the 23nd Int'l Conf. on Data Engineering (ICDE), April 15-20, 2007, Istanbul, Turkey*. IEEE, 2007, pp. 1479–1480.

[13] E. Hung, L. Getoor, and V. Subrahmanian, "PXML: A probabilistic semistructured data model and algebra," in *Proceedings of the 19th Int'l Conf. on Data Engineering (ICDE), March 5-8, 2003, Bangalore, India*, 2003, p. 467, iSBN 0-7803-7665-X.

[14] S. Abiteboul and P. Senellart, "Querying and updating probabilistic information in XML," in *Proceedings of the Int'l Conf. on Extending Database Technology (EDBT), Munich, Germany*, 2006, pp. 1059–1068, lNCS 3896.

[15] A. Doan, P. Domingos, and A. Y. Halevy, "Reconciling schemas of disparate data sources: A machine-learning approach." in *Proceedings of the ACM SIGMOD Int'l Conf. on Management of Data, Santa Barbara, California, USA*, May 2001, pp. 509–520, iSBN 1-58113-332-4.

[16] A. Doan, P. Domingos, and A. Halevy, "Learning to match the schemas of data sources: A multistrategy approach," *Machine Learning*, vol. 50, no. 3, pp. 279–301, 2003.

[17] A. Hunter and W. Liu, "Merging uncertain information with semantic heterogeneity in XML," *Knowledge and Information Systems*, vol. 9, no. 2, pp. 230–258, Feb. 2006.

[18] M. Magnani and D. Montesi, "Uncertainty in data integration: current approaches and open problems," in *Proceedings of the 1st Int'l Workshop on Management of Uncertain Data (MUD2007), 24 September 2007, Vienna, Austria*, ser. CTIT Workshop Proceedings, A. de Keijzer, M. van Keulen, and A. Dekhtyar, Eds., no. WP07-08, 2007, iSSN 0929-0672.

[19] X. L. Dong, A. Halevy, and C. Yu, "Data integration with uncertainty," in *Proceedings of the 33rd Int'l Conf. on Very Large Data Bases (VLDB), Vienna, Austria, September 23-27, 2007*. ACM, 2007, pp. 687–698.

[20] O. Benjelloun, H. Garcia-Molina, H. Kawai, T. E. Larson, D. Menestrina, Q. Su, S. Thavisomboon, and J. Widom, "Generic entity resolution in the SERF project," *IEEE Data Engineering Bulletin*, vol. 29, no. 2, pp. 13–20, 2006.

[21] D. Menestrina, O. Benjelloun, and H. Garcia-Molina, "Generic entity resolution with data confidences," in *Proceedings of the First Int'l VLDB Workshop on Clean Databases (CleanDB), September 11, 2006, Seoul, Korea*, 2006.

[22] P. DeRose, W. Shen, F. Chen, A. Doan, and R. Ramakrishnan, "Building structured web community portals: A top-down, compositional, and incremental approach," in *Proceedings of the 33rd Int'l Conf. on Very Large Data Bases (VLDB), Vienna, Austria, September 23-27, 2007*. ACM, 2007, pp. 399–410.

[23] T. Cheng, X. Yan, and K. C.-C. Chang, "EntityRank: Searching entities directly and holistically," in *Proceedings of the 33rd Int'l Conf. on Very Large Data Bases (VLDB), Vienna, Austria, September 23-27, 2007*. ACM, 2007, pp. 387–398.

[24] M. van Keulen and A. de Keijzer, "Qualitative effects of knowledge rules in probabilistic data integration," Centre for Telematics and Information Technology, Univ. of Twente, Enschede, The Netherlands, Tech. Rep. TR-CTIT-08-42, June 2008, iSSN 1381-3625.

[25] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison Wesley, 1999, iSBN 0-201-39829-X.