Towards a Generic Approach for Schema Matcher Selection: Leveraging User Pre- and Post-match Effort for Improving Quality and Time Performance

Fabien Duchateau, LIRMM - Université Montpellier II

Ph.D. Candidate, 20 November 2009

Jury Members

Bernd Amann, Professeur, Université Paris 6, Rapporteur

Jérôme Euzenat, Directeur de Recherche, INRIA Rhône-Alpes, Rapporteur

Angela Bonifati, Chercheuse, CNR, Italy, Examinatrice

Rémi Coletta, Maître de Conférence, Université Montpellier II, Examinateur

Thérèse Libourel, Professeure, Université Montpellier II, Examinatrice

Zohra Bellahsène, Professeure, Université Montpellier II, Directrice de Thèse

Data Integration and Schema Matching

Data integration aims at providing a uniform access to multiple data sources.

Applications: scientific information systems, B2B, web services composition, semantic web, etc.

A basic operation in data integration is the discovery of correspondences between data sources, especially between schemas
$\Rightarrow$ **schema matching**

Preliminaries (1/3)

Schema matching scenario: set of schemas to be matched

Mapping: a pair of schema elements which represent the same real-world concept

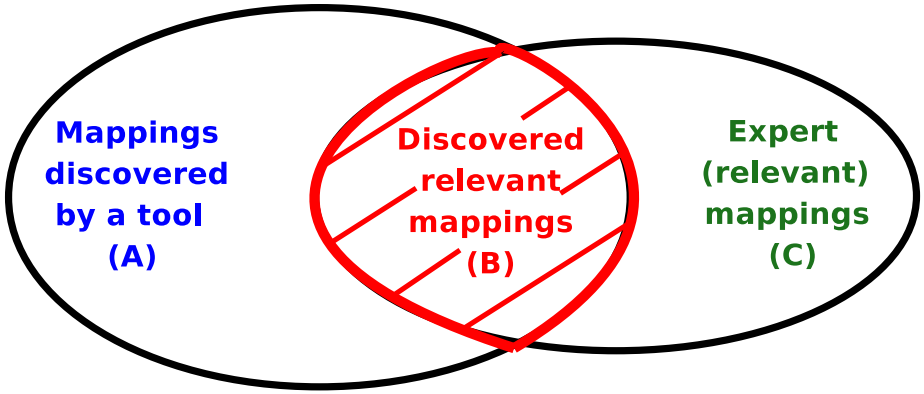
Similarity measure: metric for computing a similarity value between a pair of schema elements (e.g., Levenshtein distance, context measure). If the value is above a given threshold, the pair is considered as a mapping.

Schema matcher: algorithm which combines similarity measures to discover mappings (e.g., aggregation function, decision tree)

Preliminaries (2/3)

- 3 phases during schema matching process:
  - → pre-match (tuning, pre-processing, etc.)
  - → matching
  - → post-match (checking mappings)

- Quality measures:



$$\text{Precision} = \frac{B}{B+A} \quad \text{Recall} = \frac{B}{B+C} \quad \text{F-measure} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

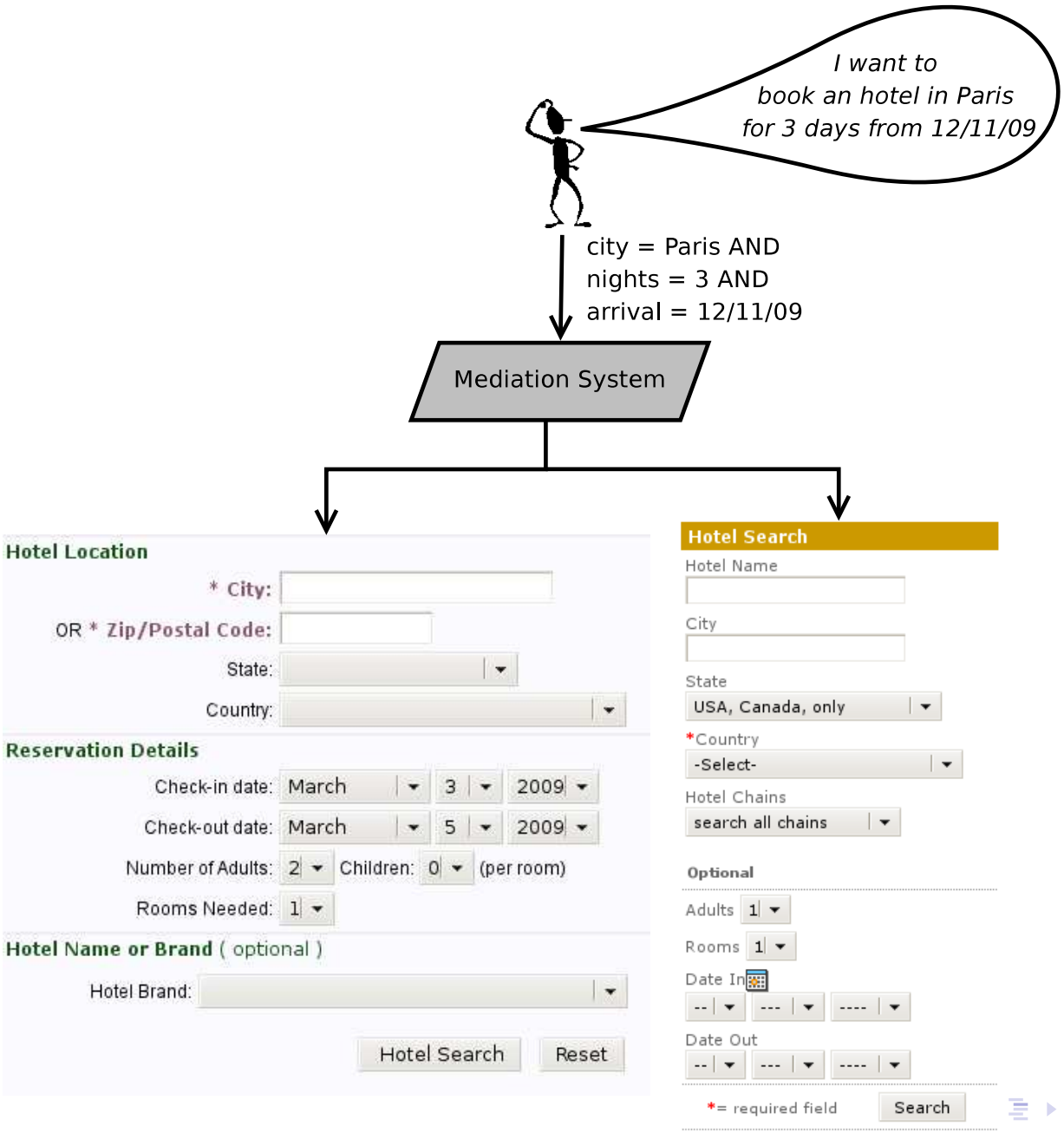
Preliminaries (3/3)

Example of real-world schema matching scenarios that we match:

- web forms (from websites such as betting, finance, etc.)
- from the literature (Thalia, travel UIUC repository)
- domain specific (biology, business order)
- web services (currency, sms)

Let us describe an example with a scenario composed of web forms.

Schema Matching Example



Schema Matching Example

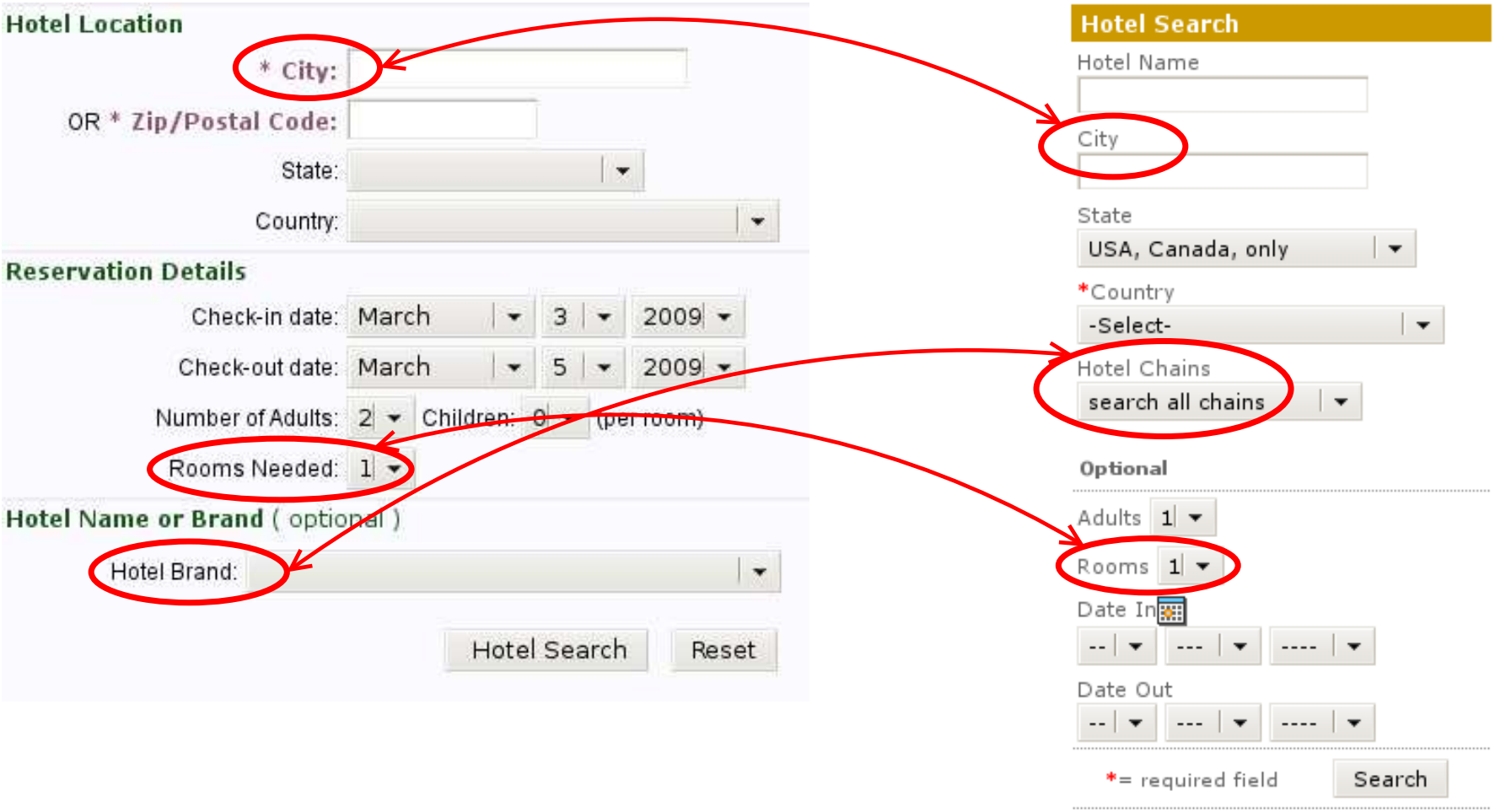
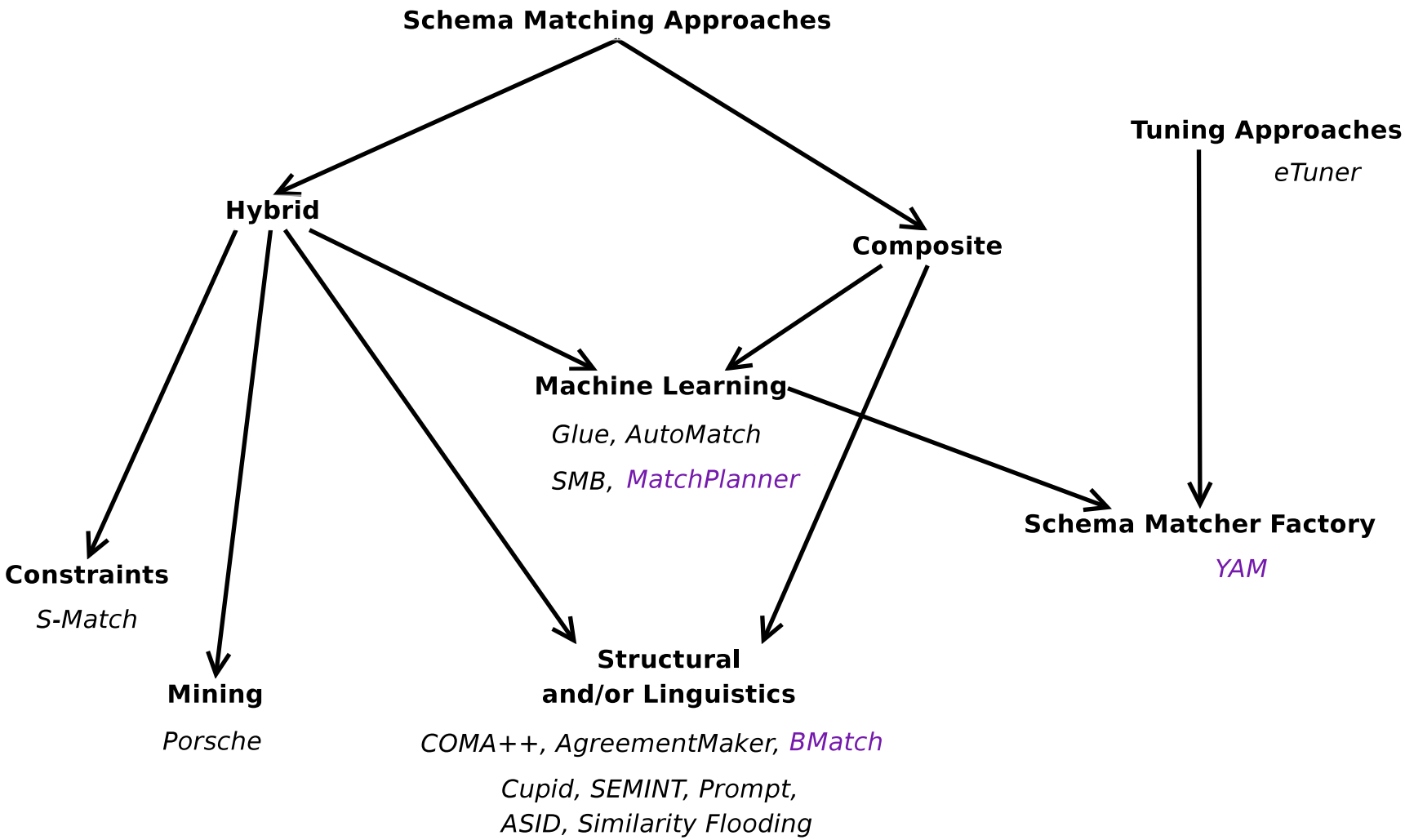


Figure: Schema matching approaches have to combine different types of **similarity measures** to discover **mappings** between schema elements

Classification of Schema Matching Approaches



Main works during my Ph.D.

Our main contributions:

- a structural similarity measure [ISI, 2008]
- new measures to evaluate post-match effort [VLDB, 2007]
- **towards a generic approach for schema matcher selection**
  - → **using plans of similarity measures for schema matching** [OTM, 2008]
  - → **learning tuned plans for matching schemas** [JWS, under revision]
  - → **a generic approach for schema matcher selection** [CIKM, 2009]

Using Plans of Similarity Measures for Schema Matching

1. Motivations
2. Our Approach
3. Results

Many schema matching approaches use an aggregation function to combine similarity measures. This entails several drawbacks:

- **quality** $\longrightarrow$ more weight to closely-related similarity measures (e.g., terminological) can have a too strong impact
- **threshold** $\longrightarrow$ one global threshold instead of a specific threshold for each similarity measure
- **performance** $\longrightarrow$ useless measures are computed

Our goal

Planning a sequence of similarity measures to be computed for each pair of schema elements [OTM, 2008]

Schema matchers can be seen as **machine learning classifiers**. Indeed, they "classify" pairs of schema elements either as relevant (mappings) or irrelevant.

Thus, the aggregation function can be replaced by a **decision tree**. Advantages of a decision tree:

- plan of similarity measures computed for each pair of schema elements
- no significant impact on the performance due to its use
- handles both numerical and categorical data

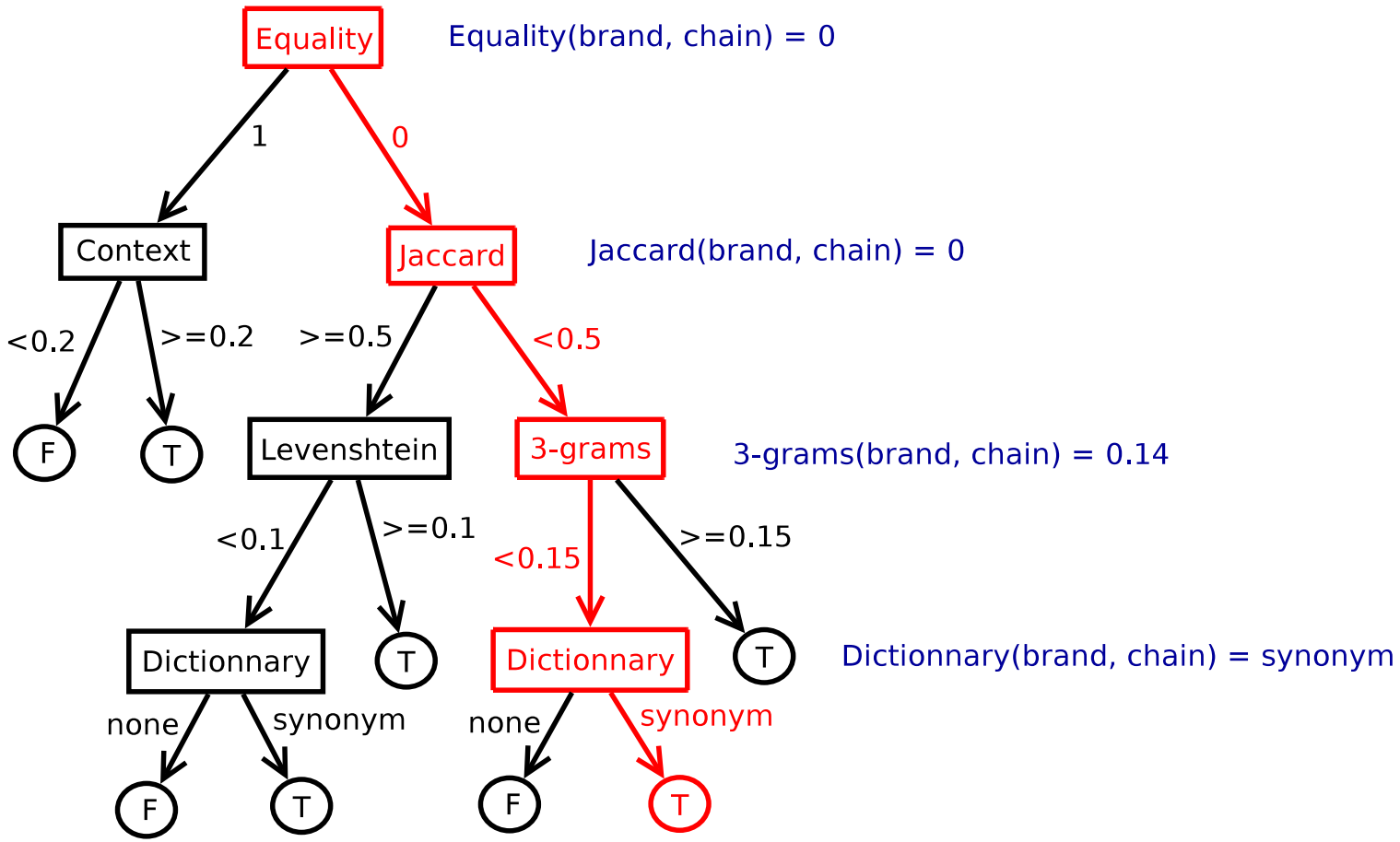


Figure: Example: matching the pair (brand, chain) with a decision tree

A tool has been implemented. We have run various experiments and we have compared our approach with existing schema matching tools.

Experiment report over 14 scenarios:

- **quality** $\rightarrow$ average F-measure is improved by 11% over COMA++ and Similarity Flooding
- **time performance** $\rightarrow$ our approach and Similarity Flooding are both twice faster than COMA++ to discover mappings between large schemas

Next step

How to automatically build appropriate decision trees for a schema matching scenario ?

A tool has been implemented. We have run various experiments and we have compared our approach with existing schema matching tools.

Experiment report over 14 scenarios:

- **quality** $\rightarrow$ average F-measure is improved by 11% over COMA++ and Similarity Flooding
- **time performance** $\rightarrow$ our approach and Similarity Flooding are both twice faster than COMA++ to discover mappings between large schemas

Next step

How to automatically build appropriate decision trees for a schema matching scenario ?

In addition to manual design of the decision tree, other issues which are common to many schema matching approaches:

- **tuning the parameters** $\longrightarrow$ the user is still in charge of this tuning (for weights, thresholds, etc.)
- **extensibility** $\longrightarrow$ how to integrate new similarity measures ?

Our goal

By relying on schemas which have already been matched, using machine learning techniques to learn decision trees
[JWS, under revision]

To learn a decision tree, we apply machine learning classification. We train the decision tree with pairs of schema elements and their mapping relevance (training data).

In our context, there are 2 classes in which the pairs can be classified: *relevant mapping* and *irrelevant mapping*. Attributes of the pairs are the similarity values computed by different similarity measures.

Example with a pair *(brand, chain)*

Trigrams	0.14
Levenhstein	0.2
Wordnet	*synonym*

⇒ **Should the pair *(brand, chain)* be classified as a relevant or irrelevant mapping ?**

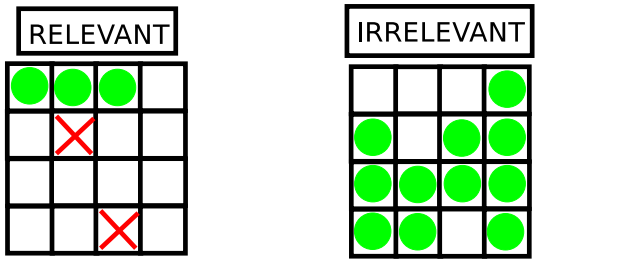
Algorithm for learning a decision tree:

- for each similarity measure, classify the training data and compute the misclassification rate
- the measure with the lowest misclassification rate is added as a new node in the tree
- for each class resulting of the previous classification, repeat the process until there is no more possible classification

Let us introduce an example with:

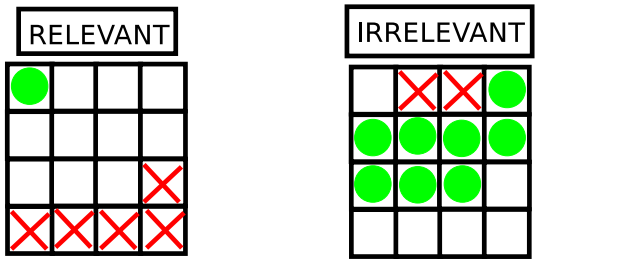
- **training data** $\longrightarrow$ 16 pairs with their mapping relevance
- **attributes** $\longrightarrow$ 2 similarity measures, *Trigrams* and *Context*

TRIGRAMS CLASSIFICATION



$$Threshold_{Trigrams} = X_1$$

CONTEXT CLASSIFICATION

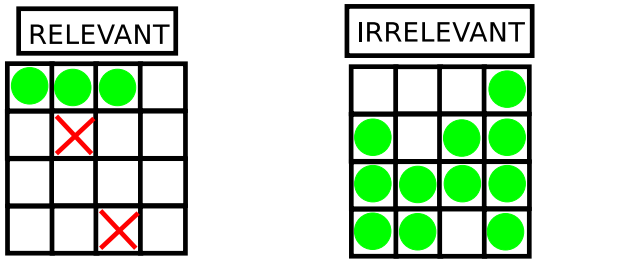


$$Threshold_{Context} = Y_1$$

pair incorrectly classified

pair correctly classified

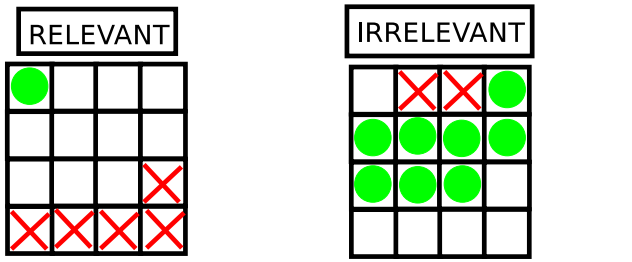
TRIGRAMS CLASSIFICATION



$$Threshold_{Trigrams} = X_1$$

$$\Rightarrow \varepsilon = \frac{2}{16}$$

CONTEXT CLASSIFICATION

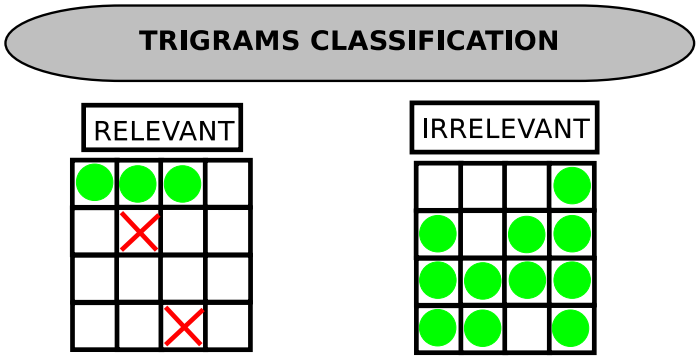


$$Threshold_{Context} = Y_1$$

$$\Rightarrow \varepsilon = \frac{7}{16}$$

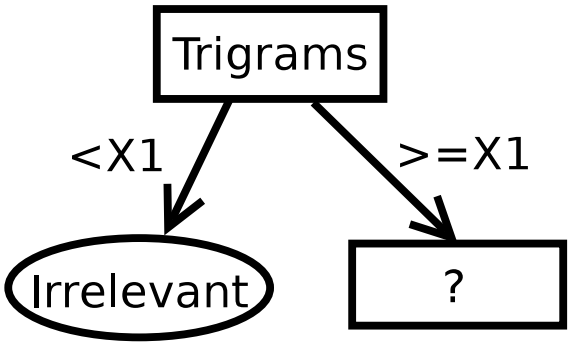
pair incorrectly classified

pair correctly classified



$$Threshold_{Trigrams} = X_1$$

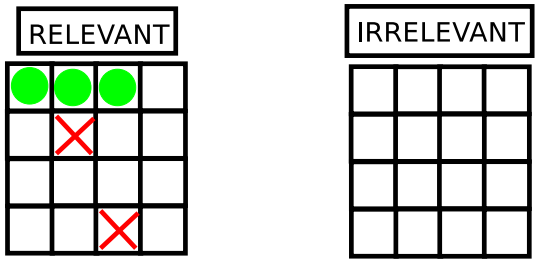
$$\Rightarrow \varepsilon = \frac{2}{16}$$



pair incorrectly classified

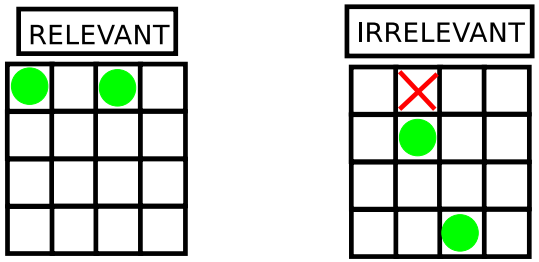
pair correctly classified

TRIGRAMS CLASSIFICATION



$$Threshold_{Trigrams} = X_2$$

CONTEXT CLASSIFICATION

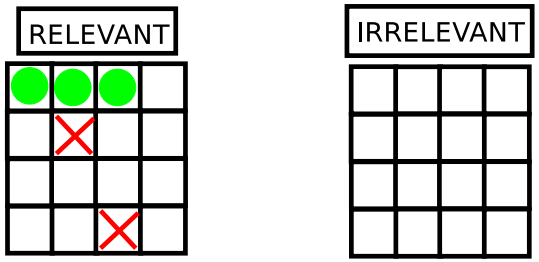


$$Threshold_{Context} = Y_2$$

pair incorrectly classified

pair correctly classified

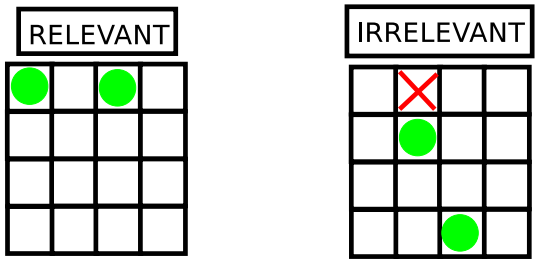
TRIGRAMS CLASSIFICATION



$$Threshold_{Trigrams} = X_2$$

$$\Rightarrow \varepsilon = \frac{2}{5}$$

CONTEXT CLASSIFICATION

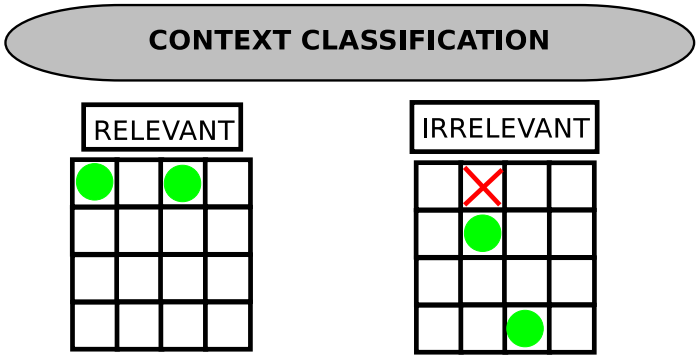


$$Threshold_{Context} = Y_2$$

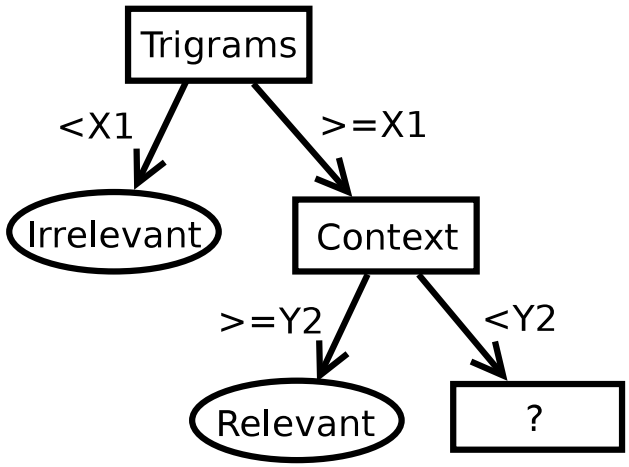
$$\Rightarrow \varepsilon = \frac{1}{5}$$

pair incorrectly classified

pair correctly classified



$$Threshold_{Context} = Y_2$$
$$\Rightarrow \varepsilon = \frac{1}{5}$$



pair incorrectly classified

pair correctly classified

A tool (MatchPlanner) has been implemented with a knowledge base containing hundreds of training data. New similarity measures are automatically integrated during learning.

Experiment report over 14 scenarios:

- **quality** → average F-measure is improved by 16% over COMA++ and Similarity Flooding
- **time performance:**

pre-match	a few minutes for learning a tree
matching	mostly in seconds (namely due to the selection and position of similarity measures in the tree)
post-match	improved (less user interactions)

Next step

Why not generalizing this approach for other classifiers ?

A tool (MatchPlanner) has been implemented with a knowledge base containing hundreds of training data. New similarity measures are automatically integrated during learning.

Experiment report over 14 scenarios:

- **quality** → average F-measure is improved by 16% over COMA++ and Similarity Flooding
- **time performance:**

pre-match	a few minutes for learning a tree
matching	mostly in seconds (namely due to the selection and position of similarity measures in the tree)
post-match	improved (less user interactions)

Next step

Why not generalizing this approach for other classifiers ?

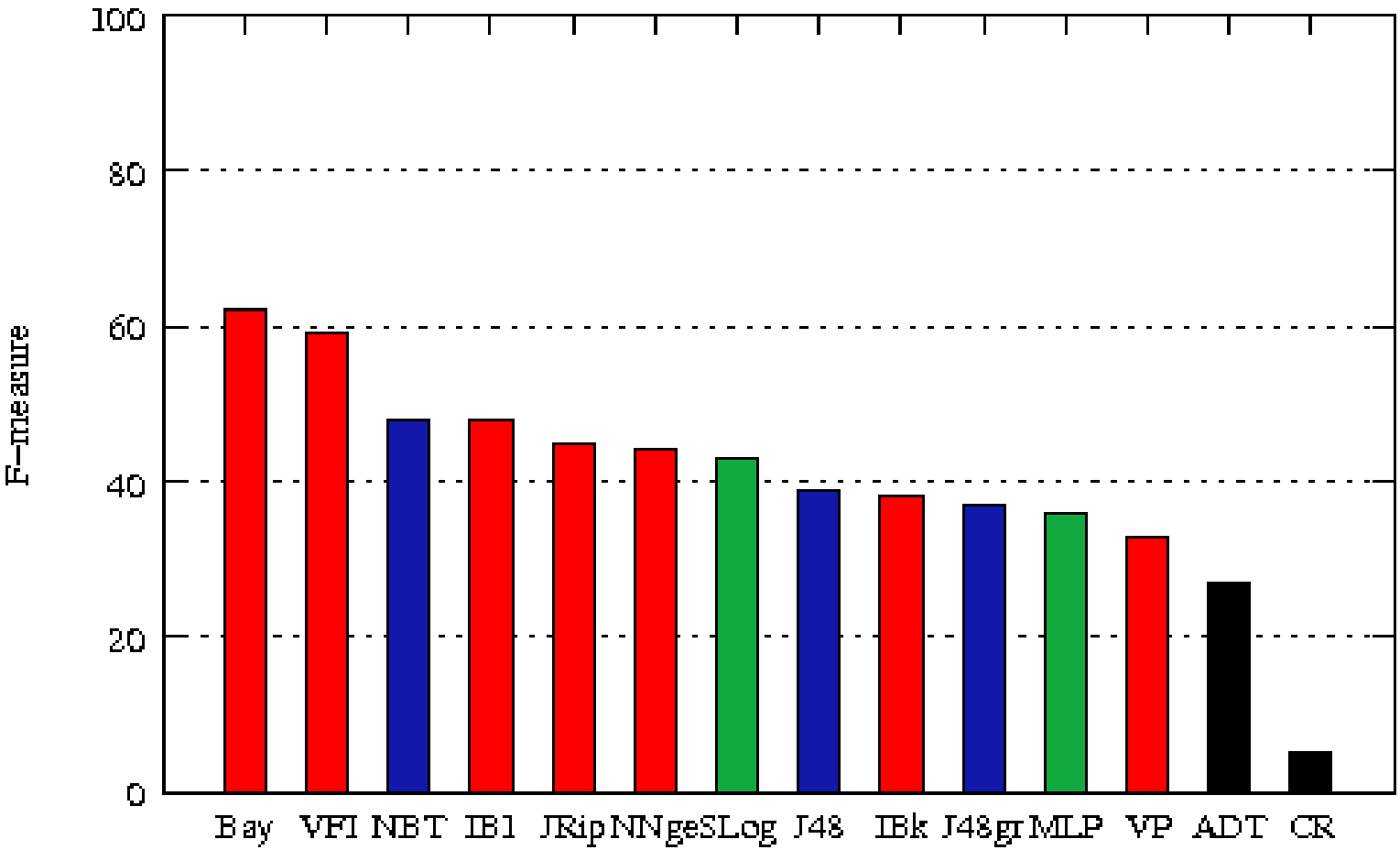


Figure: Average F-measure over 200 scenarios for each classifier

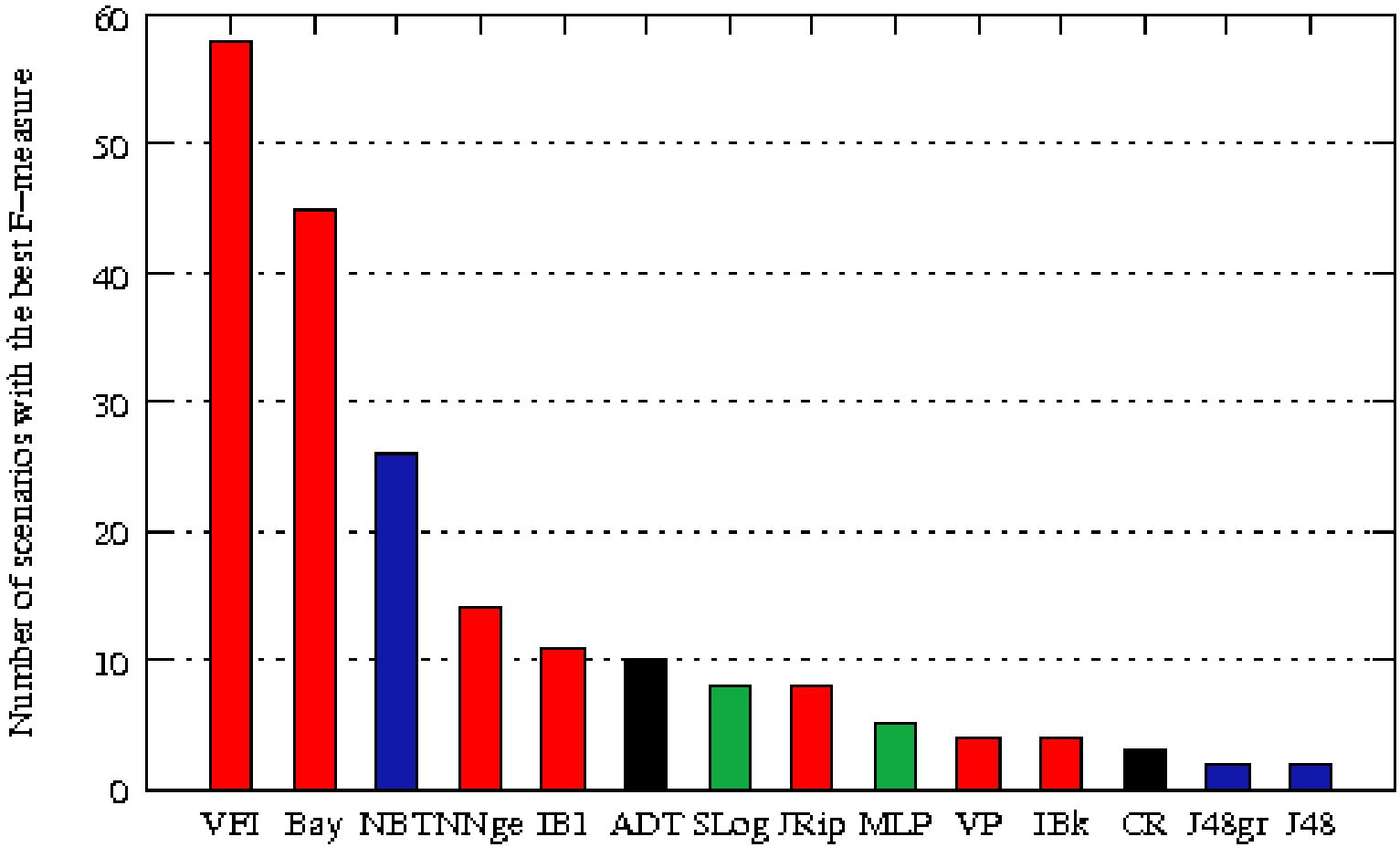


Figure: Number of scenarios (out of 200) for which each classifier obtains the best F-measure

A Generic Approach for Schema Matcher Selection

1. Motivations
2. Our Approach
   - Learning Tuned Schema Matchers
   - Integrating User Inputs
   - Selecting the Dedicated Schema Matcher
3. Results

- **automatic selection of a schema matcher** $\longrightarrow$ for some scenarios, a schema matcher may not discover any mapping.

- **user inputs** $\longrightarrow$ inputs provided by users are not sufficiently integrated in the matching process.

  expert mappings current schema matching approaches do not integrate them to improve results

  similar schemas the user owns some schemas with similar features than those to be matched

preference between precision and recall is a crucial input.

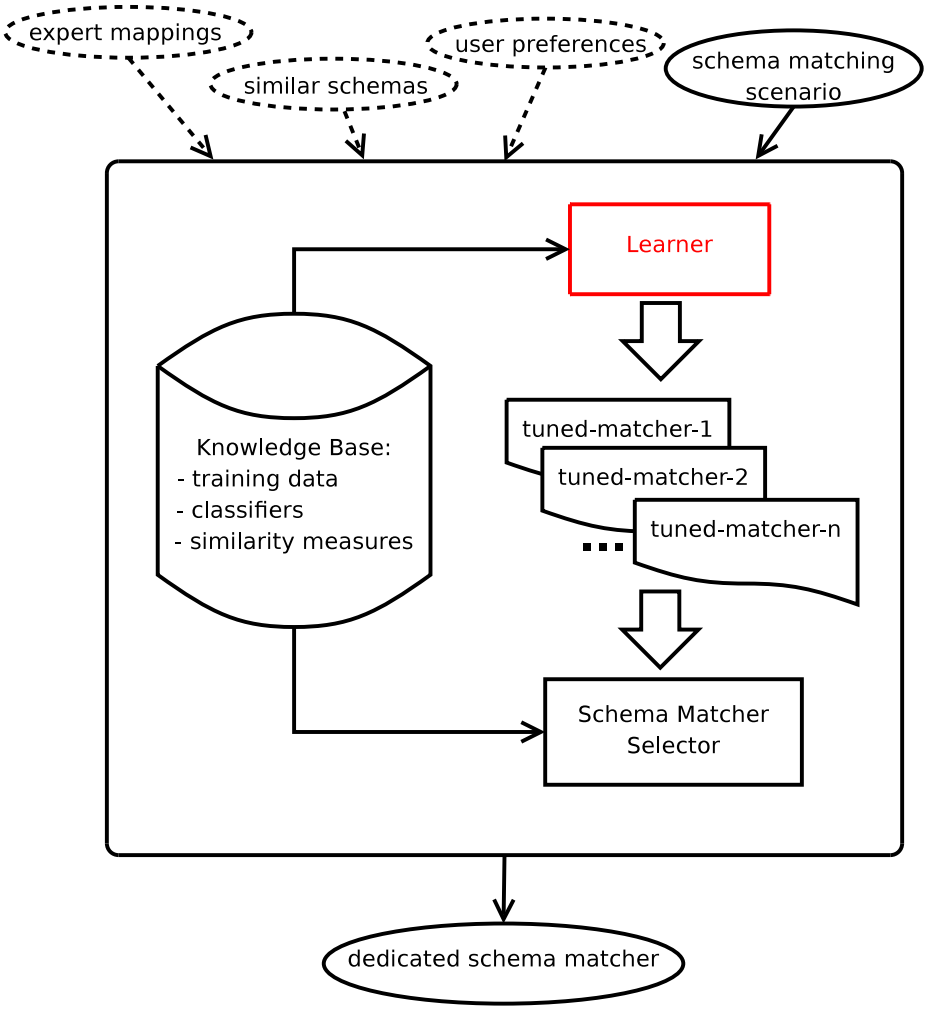
Example: 2 schemas with 100 elements each, 24 relevant mappings between them. Two matchers:

$\longrightarrow$ 16 *discovered mappings including* 12 *relevant (75% precision, 50% recall), then expert invalidates* 4 *irrelevant mappings and has to* ***manually find the 12 missing ones among 7744 pairs***.

$\longrightarrow$ 40 *discovered mappings, including* 18 *relevant (45% precision, 75% recall), then expert invalidates* 22 *irrelevant mappings and has to* ***manually find the 6 missing ones among 6724 pairs***.

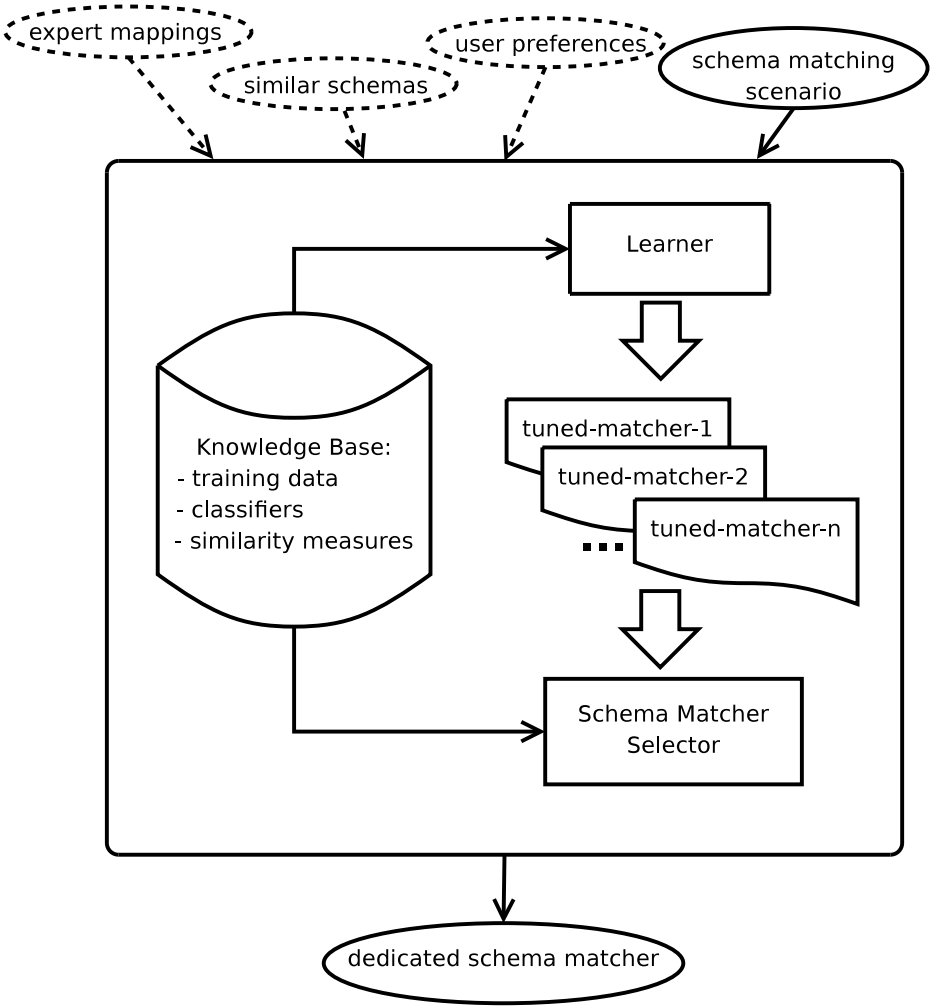
Our goal

Building a factory of schema matchers that take into account user preferences [CIKM, 2009]



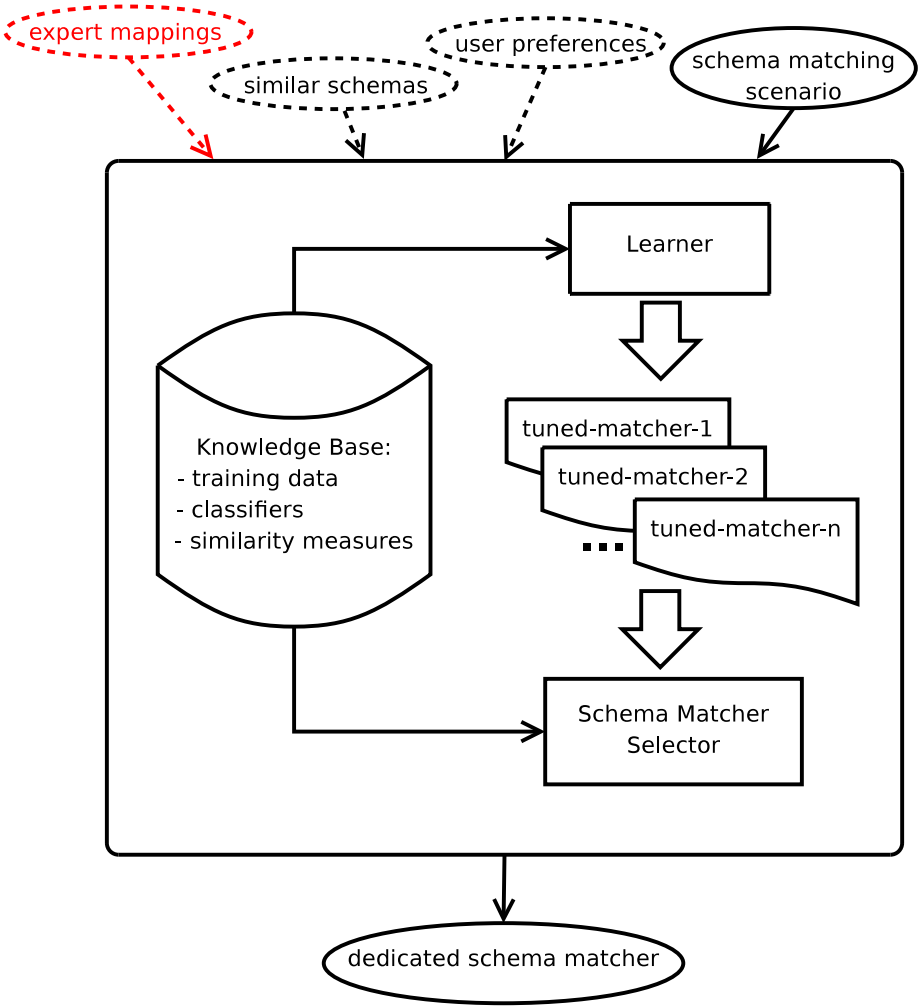
For each classifier in the KB, the *learner* generates a tuned schema matcher:

- learning is specific to each classifier



User can provide 3 **optional** inputs:

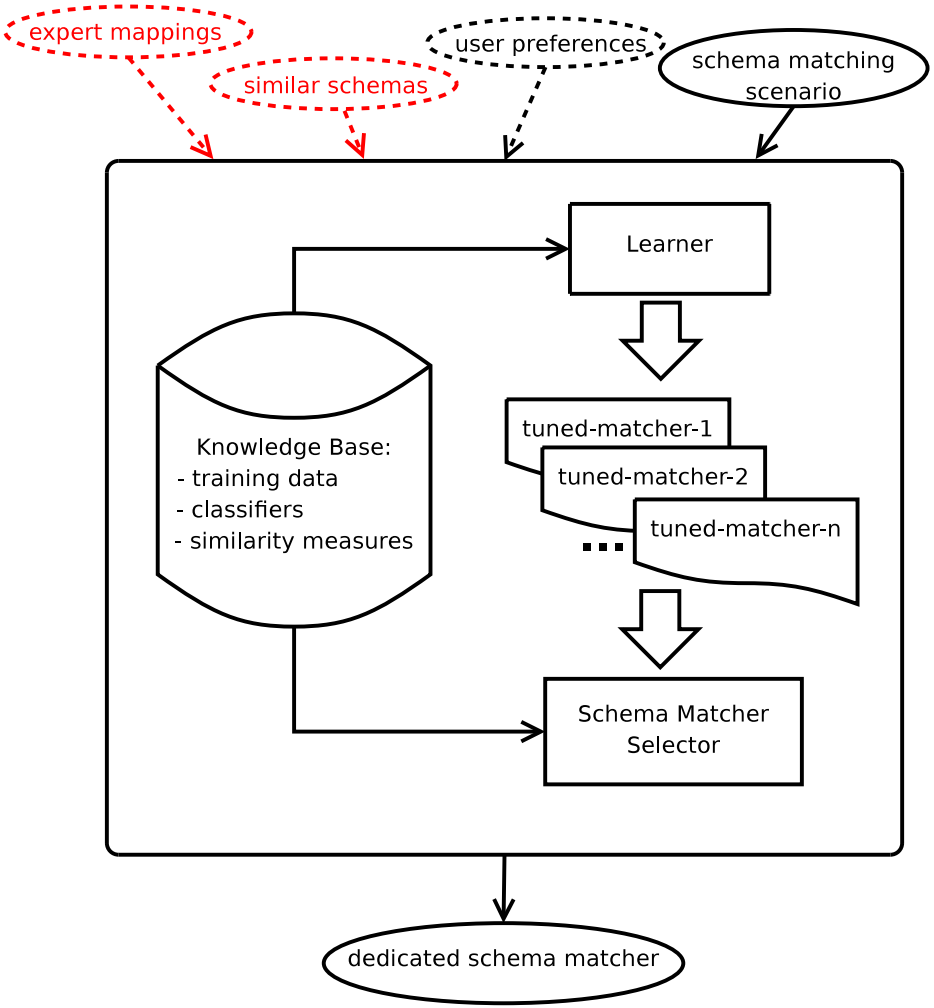
All user inputs are integrated during the learning process.



User can provide 3 **optional** inputs:

- expert mappings

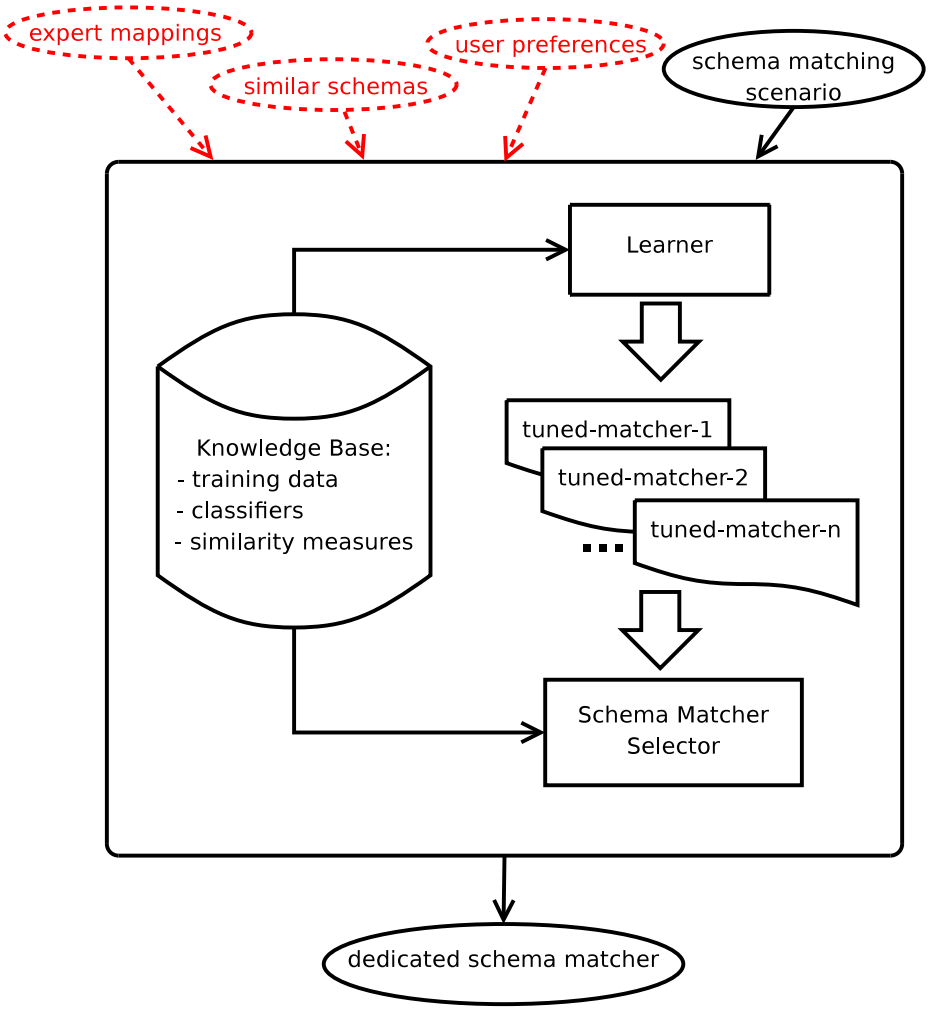
All user inputs are integrated during the learning process.



User can provide 3 **optional** inputs:

- expert mappings
- schemas which are similar (domain or schema features) than those to be matched

All user inputs are integrated during the learning process.



User can provide 3 **optional** inputs:

- expert mappings
- schemas which are similar (domain or schema features) than those to be matched
- preference between precision or recall

All user inputs are integrated during the learning process.

Similar schemas

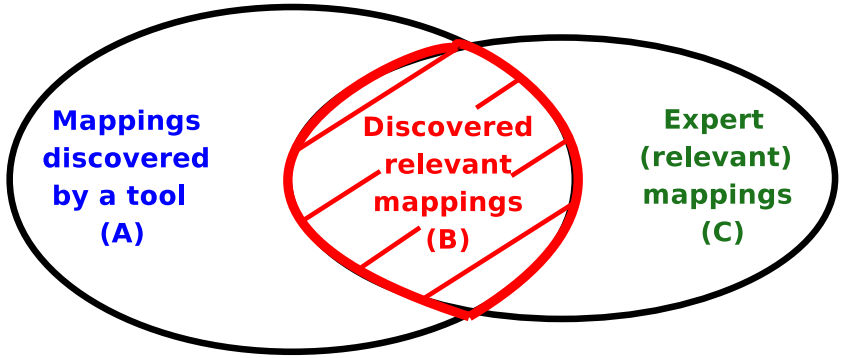
Schemas from the same domain (finance, biology, etc.) or sharing features with those to be matched can be integrated as training data.

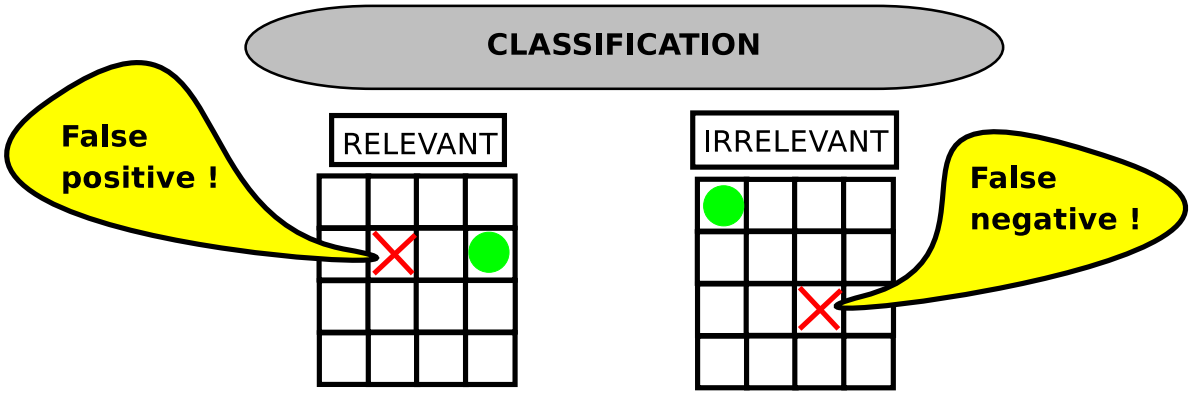
Expert mappings

Expert mappings (between the schemas to be matched) are also added to the training data. Advantages:

- this reduces the number of matching possibilities
- as a schema designer mostly keeps the same logic and methodology, the similarity measures which are efficient against the expert mappings might also be efficient with undiscovered mappings

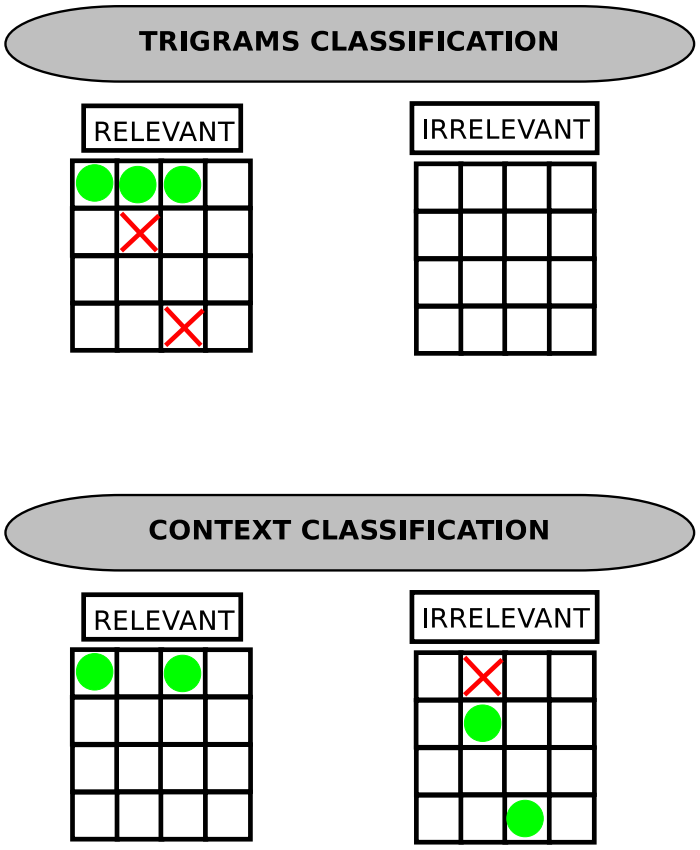
Precision / Recall



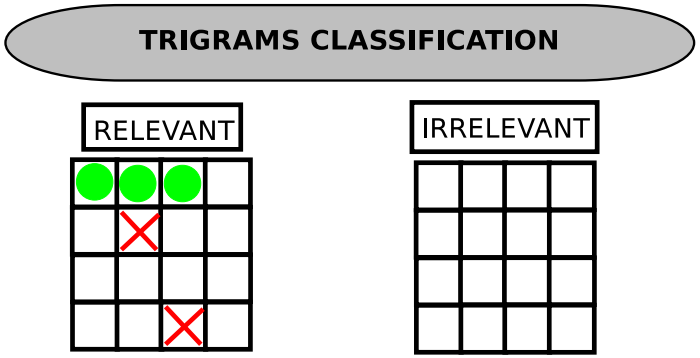


When computing the misclassification rate, we put a weight on either false positives or false negatives to respectively promote precision or recall.

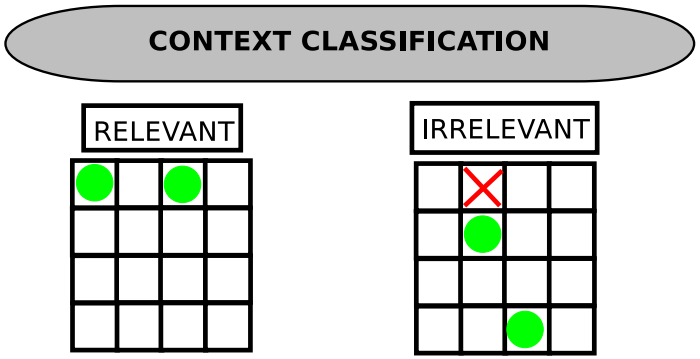
Example: a preference for recall is set to 4.



Example: a preference for recall is set to 4.

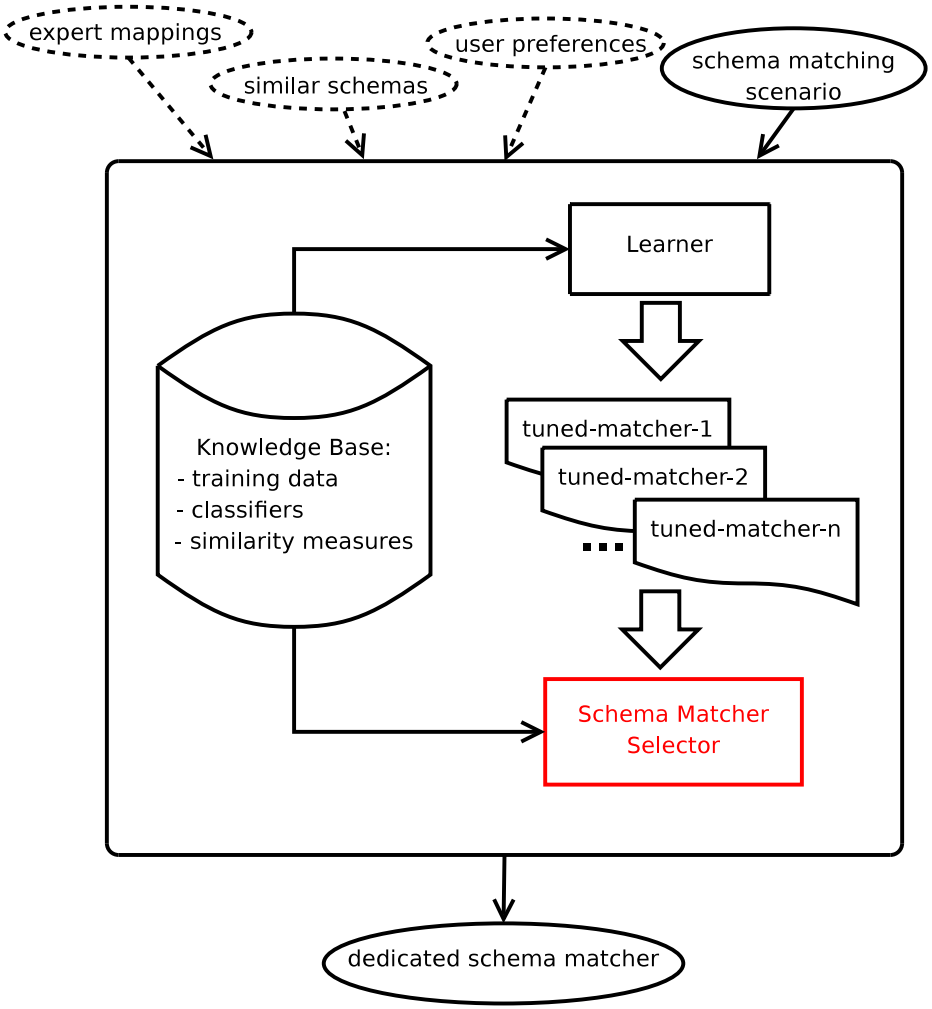


$$\Rightarrow \varepsilon = \frac{2}{5}$$



$$\Rightarrow \varepsilon = \frac{4 \times 1}{5}$$

Although the *Trigrams* measure generated more errors, it is not penalized by the misclassification of a relevant mapping.



Among all generated schema matchers, the *selector* selects the dedicated one.

Each generated schema matcher is used to match the training data (cross-validation process) and different strategies can be applied to keep the best one:

- when expert mappings/similar schemas are provided, select the schema matcher which discovers the most expert mappings on these data
- if recall (resp. precision) is promoted, keep the schema matcher which obtains the best recall (resp. precision)
- select the schema matcher which achieves the best F-measure on the training data

A tool (YAM, for Yet Another Matcher) has been implemented and we run various experiments.

Experiment report over 14 scenarios:

- **quality** $\rightarrow$ average F-measure is improved by 23% over COMA++ and Similarity Flooding
- **time performance:**

pre-match	10-20 minutes for generating a dedicated schema matcher
matching	mostly in seconds
post-match	improved (less user interactions)

Experiments Report

1. Protocol
2. Experiments
3. Summary

All our approaches have been compared with COMA++ and Similarity Flooding (only schema matching tools available) on two aspects:

- quality (precision, recall and F-measure)
- time performance

Real-world schema matching scenarios:

- web forms (from websites such as betting, finance, etc.)
- from the literature (Thalia, travel UIUC repository)
- domain specific (biology, business order)
- web services (currency, sms)

Experiments Report

We have empirically set up the number of training data for each classifier

Classifiers need more or less training data to achieve good results.

# training scenarios	Classifiers
20 and less	SLog, ADT, CR
20 to 30	J48, J48graft
30 to 50	NNge, JRip, DecTable, BayesNet, VP, FT
50 and more	VFI, IB1, IBk, SMO, NBTree, MLP

Table: Number of training scenarios for each classifier (deduced from 11000 experiments) is automatically selected by our approach

Experiments Report

Providing 5% of expert mappings improves F-measure up to 40%

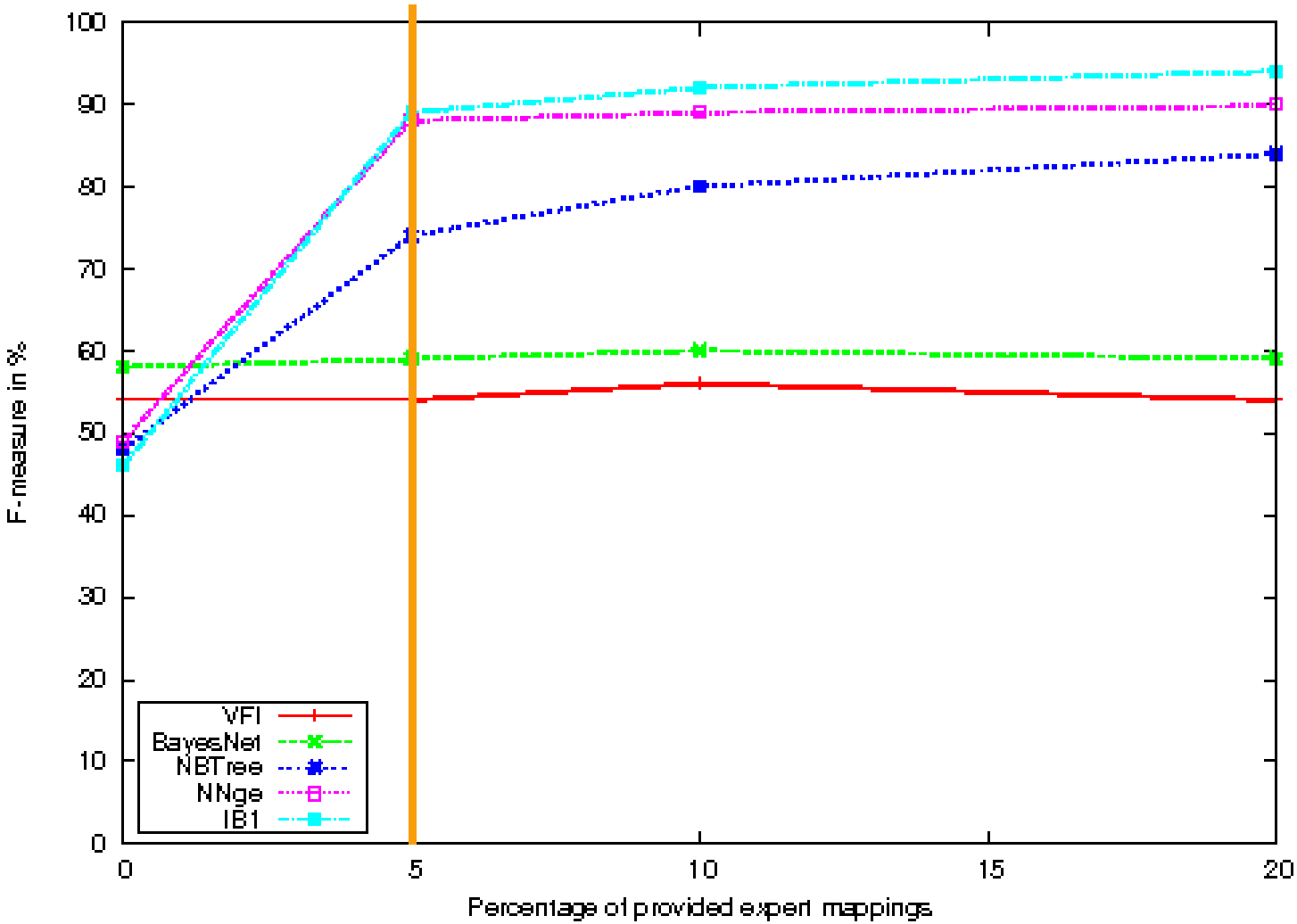


Figure: Evolution of F-measure when providing expert mappings (out of 200 scenarios)

In the next plot, we measure the post-match effort in terms of user interactions to manually achieve a 100% F-measure.

Two steps:

- all discovered mappings are (in)validated by the user
- the user manually tests all mapping possibilities for each schema element which has not been matched

This post-match effort is a worst case situation.

Experiments Report

YAM tuned in favour of recall reduces at most the post-match effort

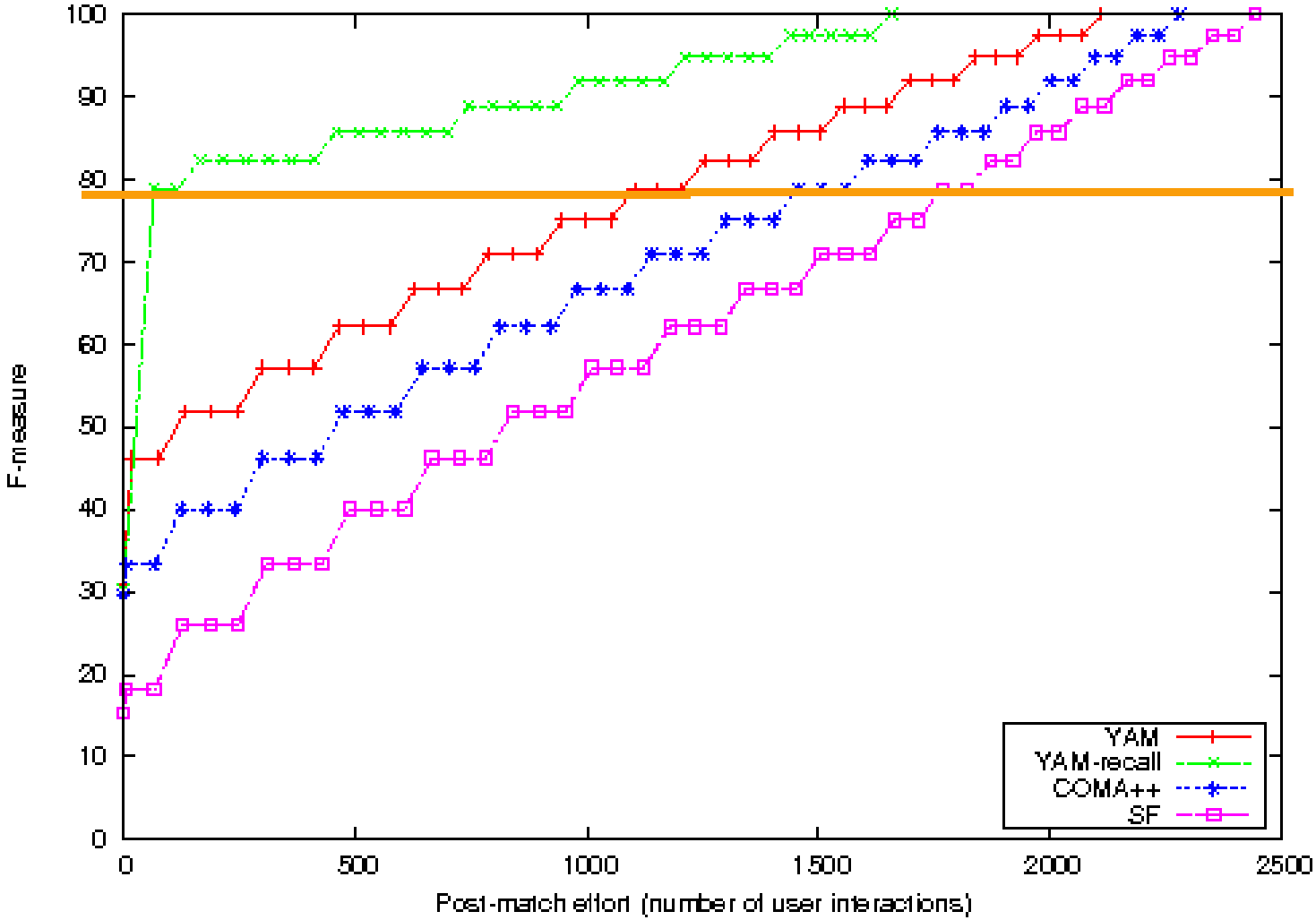


Figure: Evaluation of post-match effort for *SMS* scenario

Experiments Report

YAM (without any user inputs) obtains the best F-measure

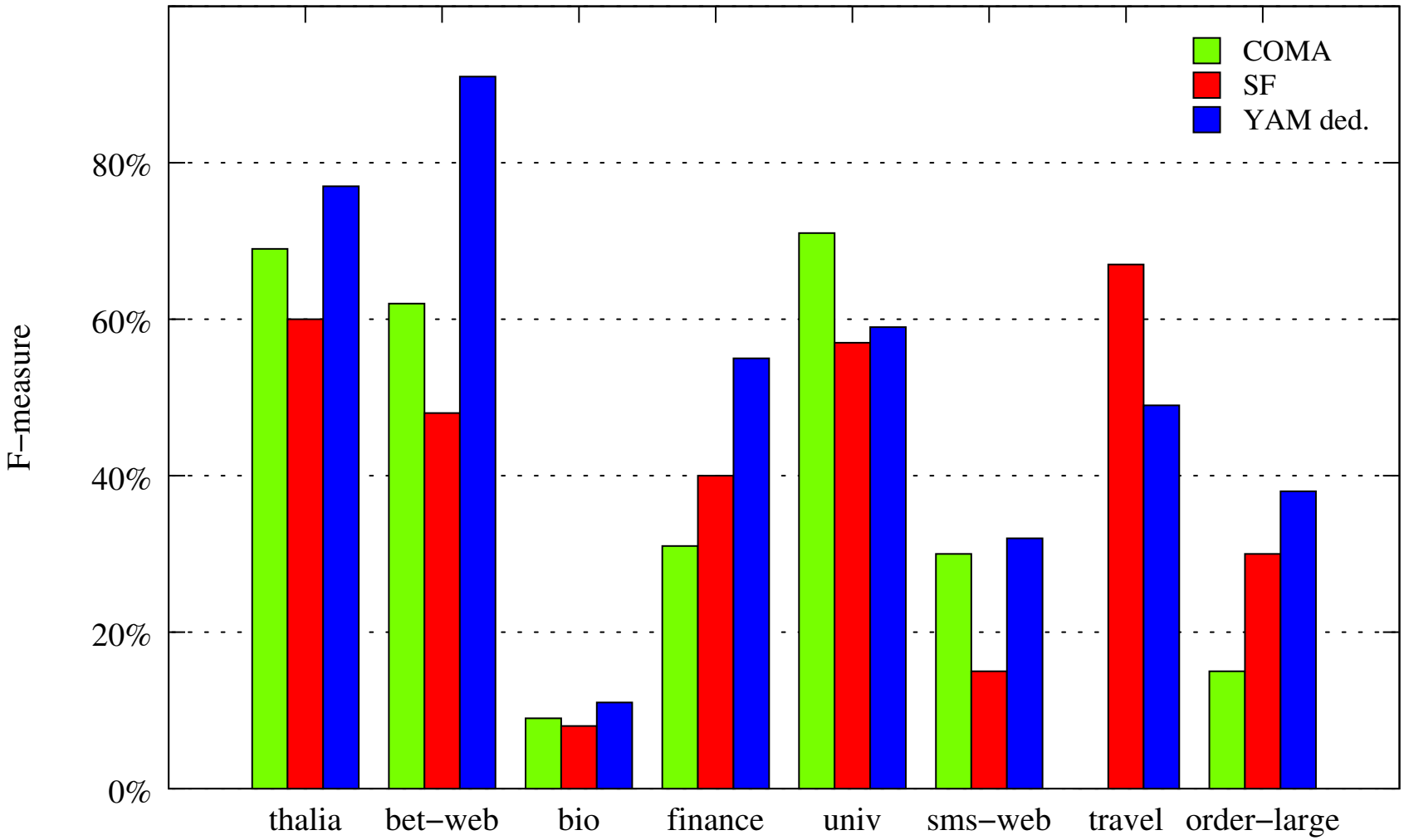


Figure: Comparing quality of schema matching tools over 8 scenarios

Experiments Report
Summary of our Results (1/2)

	Precision	Recall	F-measure
YAM	81%	65%	72%
COMA++	66%	38%	48%
SF	61%	43%	50%
YAM-recall	68%	78%	73%
YAM-similar-schemas	80%	72%	76%
YAM-expert-mappings (5%)	88%	90%	89%

Table: Average quality results over 14 scenarios

The more inputs (expert mappings, similar schemas or preference for recall/precision) the user provides, the most efficient the dedicated schema matcher will be.

Lessons learned:

- we have shown that aggregation functions can be replaced by other methods to combine similarity measures
- we have demonstrated a strong need for a schema matcher factory
- time performance during pre-match and matching is rarely important
- achieving high quality results enables better time performance during post-match

Conclusion and Perspectives

Conclusion and Perspectives

By first proposing a new method to combine similarity measures, we have finally generalized our approach. Users spend some time during pre-match to strongly improve time performance during post-match.

To reduce post-match effort:

- an automatic generation and selection of a tuned schema matcher
- a tight integration of user inputs
- a measure for computing this post-match effort

Conclusion and Perspectives

Short-term perspectives:

- Extending to ontologies
- Discovering complex mappings
- Reusing dedicated schema matchers

Long-term perspectives:

- Connecting large scale networks
- Uncertainty

Selected Publications

[OTM, 2008] Fabien Duchateau, Zohra Bellahsene, and Remi Coletta. A flexible approach for planning schema matching algorithms. In *OTM*, volume 1, pages 249–264, 2008.

[VLDB, 2007] Fabien Duchateau, Zohra Bellahsene, and Ela Hunt. Xbenchmatch: a benchmark for xml schema matching tools. In *VLDB*, pages 1318–1321, 2007.

[ISI, 2008] Fabien Duchateau, Zohra Bellahsene, and Mathieu Roche. Improving quality and performance of schema matching in large scale. *Ingénierie des Systèmes d'Information*, 13(5):59–82, 2008.

[JWS, under revision] Fabien Duchateau, Remi Coletta, and Zohra Bellahsene. Learning self-tuned plans for matching data sources. Submitted to *JWS*, 2009.

[CIKM, 2009] Fabien Duchateau, Remi Coletta, Zohra Bellahsene, and Renée J. Miller. Yet another matcher. In *CIKM*, 2009.

Given a schema matching scenario and according to user inputs, the idea is to generate the most appropriate schema matcher (i.e., the **dedicated** schema matcher).

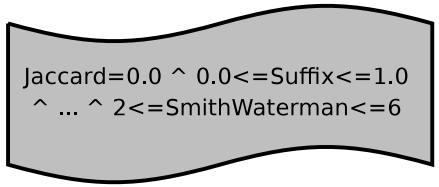


Figure: Rule-based schema matcher

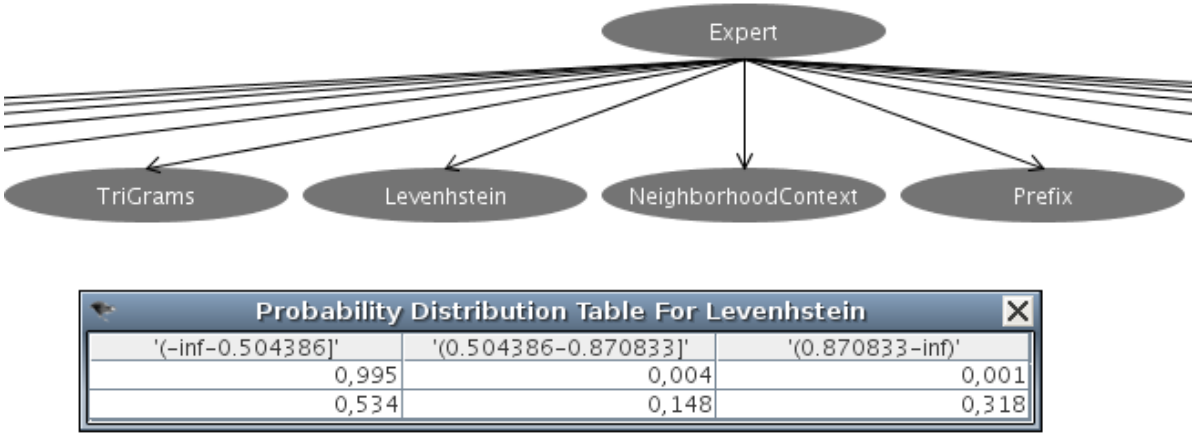


Figure: Bayes network schema matcher