# A Context-based Approach for Complex Semantic Matching

Youssef Bououlid Idrissi and Julie Vachon

DIRO, University of Montreal, Montreal, QC, Canada
{bououlii,vachon}@iro.umontreal.ca

**Abstract.** Semantic matching[1] is a fundamental step in implementing data sharing applications. Most systems automating this task however limit themselves to finding simple (*one-to-one*) matching. In fact, complex (*many-to-many*) matching raises a far more difficult problem as the search space of concept combinations is often very large. This article presents INDIGO, a system discovering complex matching in two steps. First, it semantically enriches data sources with complex concepts extracted from their development artifacts. It then proceeds to the alignment of data sources thus enhanced.

**Key words:** complex matching, semantic matching, context analysis

## 1 Introduction

Semantic matching consists in finding semantic correspondences between heterogenous sources. When done manually, this task can prove to be very tedious and error prone [3]. To date, many systems [1, 4] have addressed the automation of this stage. However, most solutions confine themselves to simple matching (*one-to-one*) although complex matching (*many-to-many*) is frequently required in practice. Linking concept *address* to the concatenation of concepts *street + city* is a typical example of a complex match. The little work addressing complex matching can be explained by the greater complexity of finding complex matches than of discovering simple ones. To cope with this challenging problem, this article introduces the solution implemented by our matching system INDIGO[2]. INDIGO avoids searching such large spaces of possible concept combinations. It rather implements an innovative solution based on the exploration of the data sources' *informational context*, which can indeed contain very useful semantic hints about concept combinations. The informational context of a data source is composed of all the available textual and formal artifacts documenting, specifying, implementing this data source. INDIGO distinguishes two main sets of documents in the informational context (cf. [2] for details). The first set, called the descriptive context, gathers all the available data source specification and

[1] also called *semantic alignment*, or simply *matching*
[2] INteroperabilty and Data InteGratiOn

documentation files produced during the different development stages. The second set is called the operational context. It is composed of formal artifacts such as programs, forms or XML files. In formal settings, significant combinations of concepts are more easily located (e.g. they can be found in formulas, function declarations, etc.). INDIGO thus favors the exploration of the operational context to identify relevant concept combinations that can form new *complex concepts*. Complex concepts are added to data sources as new candidates for the matching phase. As an experiment, INDIGO was used for the semantic matching of two database schemas taken from two open-source e-commerce applications, *Java Pet Store* [5] and *eStore* [6], provided with all their source code files.

## 2 INDIGO's architecture

To handle both context analysis and semantic matching, INDIGO presents an architecture composed of two main modules: a *Context Analyzer* and a *Mapper* module. The *Context Analyzer* module takes the data sources to be matched along with their related contexts and proceeds to their enrichment before delivering them to the *Mapper* module for their effective matching.

### 2.1 *Context Analyzer*

The *Context Analyzer* comprises two main modules, each one being specialized in a specific type of concept extraction. 1) The *Concept name collector* explores the descriptive context of a data source to find (simple) concept names related to the ones found in the data source's schema. 2) The *Complex concept extractor* analyzes the operational context to extract complex concepts. The left side of Figure 1 shows the current architecture of our *Context Analyzer*. Modules are either *basic* or *meta* analyzers. The analysis performed is based on heuristic rules in all cases.

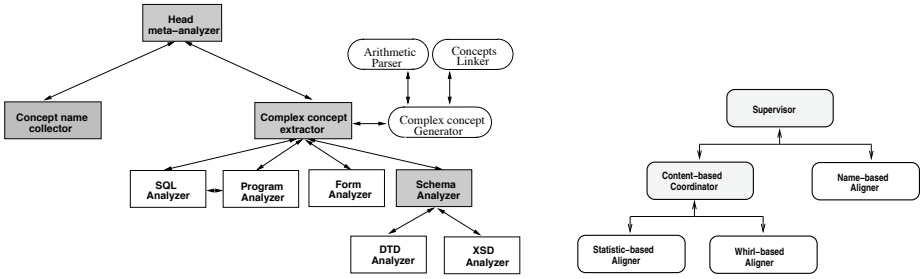


**Fig. 1.** The *Context Analyzer* and the *Mapper* modules

**Basic analyzers.** Basic analyzers (depicted by white boxes on Fig 1) are responsible for the effective mining of complex concepts. The extraction is performed as dictated by heuristic rules having the following shape $ruleName$ ::

$SP_1||SP_2...||SP_n \rightarrow extractionAction$. The left part is a disjunction of syntactic patterns (noted SP) that basic analyzers try to match[3] when parsing a document. A SP is a regular expression which can contain pattern variables *name*, *type*, $exp_1$, $exp_2$, ... $exp_n$ (e.g. for an accessor method in a Java program: $SP_i$ = {`public` *type* `get`*name* * `return` $exp_1$}). When a basic analyzer recognizes one of the SPs appearing on the left-hand side of a rule, pattern variables are assigned values (by pattern matching) and the corresponding right-hand side action of the rule is executed. This action builds a complex concept $< name, type, concept\_combinaison >$ using the pattern variables' values. Each basic analyzer applies its own set of heuristic extraction rules over each of the artifacts it is assigned. Our current basic analyzers deal with forms, programs, SQL requests, DTD and XSD schemas.

**Meta-analyzers.** Each meta-analyzer is in charge of a set of artifacts composing the informational context. Its role essentially consists in classifying these artifacts and assigning each of them to a relevant child. To do so, it applies heuristics like checking file name extensions or parsing file internal structures. The meta-analyzer module at the head of the *Context Analyzer* is in charge of the *Concept name collector* and the *Complex concept extractor* coordination. It enhances data sources with the simple and complex concepts respectively delivered by these two modules. For complex concepts, this enrichment step not only requires the name of the enriching concepts but also the values[4] associated to them. Regarding the *Complex concept extractor*, let's mention it coordinates the actions of the basic analyzers responsible for the complex concepts extraction. In addition, it relies on an internal module, called *complex concept generator*, that validates discovered concept combinations and generates complex concepts by replacing expressions (coming from source code) by appropriate concepts of the data source.

### 2.2 *Mapper* module

As shown on the right side of Figure 1, the current architecture of the *Mapper* is composed of several matching modules hierarchically organized. Each aligner supports a given matching strategy and is responsible of generating a mapping between data sources. On top, each coordinator supervises a given set of aligners and combines their returned results. The current implementation of the *Mapper* comprises the three following aligners: (1) The *Name-based aligner* proposes matches between concepts having similar names with regards to the *JaroWinkler* lexical similarity metric. (2) The *Whirl-based aligner* uses an adapted version of the so-called *WHIRL* technique developed by Cohen and Hirsh [7] to match concepts having similar instances. Finally, (3) the *Statistic-based aligner* compares concepts' content which is represented, in this case, by a normalized vector describing seven characteristics (e.g. *minimum value*, *maximum*, *variance*, etc.).

[3] N.b. This kind of text matching is called "pattern matching"

[4] These values are assessed by querying the database using SQL SELECT statements.

## 3 Experimental results

INDIGO's *Context Analyzer* has been applied to match the data sources of the *eStore* and *PetStore* applications. Two measures, respectively called *significance* and *relevance*, were defined to evaluate its performance. The significance measure indicates the percentage of extracted complex concepts presenting a semantically sound combination of concepts (e.g. *concat(first_name, last_name)*). The relevance measure is used to compute the proportion of significant complex concepts which effectively appear in the final mapping of the two data sources. The *Context Analyzer* has globally discovered 31 complex concepts of which 87% were significant. Of course, not all of them were relevant. The manual examination of the data sources revealed that *eStore*'s data source only contained two complex concepts while *PetStore*'s contained none. INDIGO nevertheless succeeded in discovering both relevant complex concepts of *eStore*. After their enhancement with complex concepts, the *PetStore*'s and the *eStore*'s data sources were matched by the *Mapper* module which was able to discover all complex matches.

## 4 Conclusion

We proposed INDIGO, an innovative solution for the discovery of complex matches between data sources. Avoiding to search the unbounded space of possible concept combinations, INDIGO discovers complex concepts by searching operational context artifacts of data sources. Newly discovered complex concepts are added to data sources as new matching candidates for complex matching. INDIGO implements a *Context analyzer* and a *Mapper* module both offering a flexible and extensible hierarchical architecture. Specialized analyzers and aligners can be added to allow the application of new mining and matching strategies. Extensibility and adaptability are undoubtedly appreciable qualities of INDIGO. Our first experiments with INDIGO showed the pertinence of this approach and let's hope for promising futur results.

## References

1. Rahm, E., P.A. Bernstein: A Survey of Approaches to Automatic Schema Matching. VLDB Journal Vol. 10, Issue. 4 (2001) 334–350
2. Bououlid, I.Y., Vachon, J.: Context Analysis for Semantic Mapping of Data Sources Using a Multi-Strategy Machine Learning Approach. In Proc. of the International Conf. on Enterprise Information Systems (ICEIS05), Miami, USA (2005) 445–448.
3. Li, W.S., Clifton, C.: Semantic Integration in Heterogeneous Databases Using Neural Networks. In Proc. of the 20th Conf. on Very Large Databases (VLDB) (1994) 1–12.
4. Euzenat, J., et al: State of the Art on Ontology Alignment. Part of a research project funded by the IST Program of the Commission of the European Communities, project number IST-2004-507482. *Knowledge Web Consortium* (2004).
5. Sun Microsystems (http://java.sun.com/developer/releases/petstore/)(2005).
6. McUmber, R.: Developing pet store using rup and xde. *Web Site* (2003).
7. Cohen, W., Hirsh, H.:Joins that Generalize: Text Classification using Whirl. In Proc. of the Fourth Int. Conf. on Knowledge Discovery and Data Mining (1998).