A CONTEXT-BASED APPROACH FOR LINGUISTIC MATCHING

Youssef Bououlid Idrissi and Julie Vachon
DIRO, University of Montreal
Montreal, Quebec, H3C 3J7, Canada

Keywords: Context analysis, semantic alignment, linguistic matching, data source integration.

Abstract: As currently implemented by most data mapping systems, linguistic matching often boils down to string comparison or to a synonym look-up in a dictionary. But these solutions have proved to be inefficient for dealing with highly heterogeneous data sources. To cope with data source heterogeneity more efficiently, we introduce INDIGO, a system which computes semantic matching by taking into account data sources' context. The distinctive feature of INDIGO consists in enriching data sources with semantic information extracted from their individual development artifacts before mapping them. As explained in this article, experiments conducted on two case studies proved the relevance of this approach.

1 INTRODUCTION

Considering their omnipresence in matching solutions (Noy and Musen, 2001; Hu et al., 2006; J. Madhavan and Rahm, 2001; Euzenat and et. al, 2004), linguistic matching strategies deserve to be further developed and improved. Current strategies commonly resort to two main evaluation techniques: (1) string comparison metrics and (2) lexical similarity functions. String metrics compare concept names according to their string characteristics (e.g. number of shared characters, similar string lengths, etc.). As for lexical functions, they assess similarity between concept names rather by comparing their respective meanings. Hence, each concept name is looked up in some external dictionary (e.g. WordNet) in order to extract the set of words that are semantically related to it (e.g. by synonymy, hyponymy, etc.). But in practice, unfortunately, neither string metrics nor lexical-based functions prove to be robust matching techniques when data sources to be aligned are very heterogeneous. In particular, lexical-based functions rarely perform well since general dictionaries on which they rely do not consider domain specificities and can even muddle up matching with irrelevant synonyms or homonyms. A reference to the context is often mandatory to clear up ambiguities. Hence, this article introduces INDIGO[1], a system which implements an innovative solution based on the exploration of the data sources' *informational context*. The informational context of a data source is composed of all the available textual and formal artifacts documenting, specifying, implementing this data source. It therefore conceals precious supplementary information which can provide useful insights about the semantics of data sources' concepts.

INDIGO distinguishes two main sets of documents in the informational context (cf. (Bououlid and Vachon, 2005)). The first set, called the *descriptive context*, gathers all the available data source specification and 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. To get contextual semantic information, INDIGO focuses on the first set of documents i.e. the descriptive context[2]. Given their descriptive nature, these documents are more likely to reveal concept names semantically related to the ones found in the data sources. INDIGO explores the descriptive context to extract the words

[1]INteroperability and Data InteGratiOn
[2]The operational context is explored for other purposes, such as for elaborating complex matching (Bououlid and Vachon, 2007).

Bououlid Idrissi Y. and Vachon J. (2007).
A CONTEXT-BASED APPROACH FOR LINGUISTIC MATCHING.
In *Proceedings of the Second International Conference on Software and Data Technologies - Volume ISDM/WsEHST/DC*, pages 197-202
DOI: 10.5220/0001345601970202

found in the vicinity of some data source concept name found within a phrase in a document. These neighboring words will be useful to evaluate similarity between concepts. Extracted words are integrated and related to their corresponding concept in the data source. In INDIGO, this enhancement of a data source with information gathered from its context is called *data source enrichment*.

To show the benefits of data source enrichment, INDIGO was compared to two well-known matching approaches, that is Similarity Flooding (Melnik et al., 2002) and V-Doc (Y. Qu and Cheng, 2006). The experiment consisted in applying each matching system over the two following case studies: (1) Matching of two database schemas taken from two open-source e-commerce applications: *Java Pet Store* (Microsystems, 2005) and *eStore* (McUmber, 2003) and (2) Matching of two real data sources describing courses taught at Cornell University and at the University of Washington.

The rest of the paper is organized as follows. Section 2 surveys recent work on linguistic matching. Section 3 describes the current implementation of INDIGO's *Context Analyzer* and *Mapper* modules. Experimental results of our two case studies are presented and commented in Section 4. Concluding remarks and comments on future work are given in Section 5.

2 RELATED WORK

As pointed out in section 1, there are essentially two categories of techniques for linguistic matching: string comparison metrics and lexical-based matching methods. Concerning string metrics, the most popular are certainly the following (Euzenat and et. al, 2004): Levenstein, Needleman-Wunsch, Smith-Waterman, Jaro-Winkler and Q-Gram. As for lexical-based metrics, they were mostly developed within the framework of specific mapping systems such as ASCO (B.T. Le and Gandon, 2004) and HCONE-merge (K. Kotis and Stergiou, 2004) to serve their own application objectives. Both of these systems resort to WordNet to collect additional semantic information (sets of words) about the concept to be matched. For instance, ASCO searches for synonyms while HCONE-merge looks for hyponyms.

Among the numerous linguistic matching solutions mentioned in the literature, we had a closer look at two systems frequently cited by researchers for their performances and which were available on the net: Similarity Flooding (**SF**)(Melnik et al., 2002) and V-Doc (Y. Qu and Cheng, 2006). We selected those two applications to experimentally compare them with our own system INDIGO.

Similarity Flooding (SF) is a generic algorithm used to match different kinds of data structures called models. Models can be composed of data schemas, data instances or a combination of both. The **SF** algorithm converts both source and target models into some proprietary labeled directed graph representation (G_1 and G_2). It then applies an iterative fixed point based procedure over these two graphs to discover matches between their respective nodes.

V-Doc is the linguistic matching module of Falcon-AO (Hu et al., 2006), a matching system for ontologies. V-Doc constructs for each entity in the ontologies to be aligned, a *virtual document* consisting in a set of words extracted from the name, label and comment fields of the entity and of its neighbors within the ontology. It then compares virtual documents using the well-known TF/IDF[3] technique.

INDIGO aims at the same objectives as the above systems. From a conceptual standpoint, it belongs to the category of systems relying on multiple strategic matchers (A. Doan and Halevy, 2003; Do and Rahm, 2002). It distinguishes itself by taking into account the informational context of data sources in its alignment process.

3 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 related context documents and proceeds to their enrichment before delivering them to the *Mapper* module for their effective matching.

3.1 Context Analyzer

The role of the *Context analyzer* consists in parsing the artifacts related to a data source with the aim to extract pertinent semantic information and to enrich this data source with it. Our implementation targets two types of enrichment: (1) enhancement of a data source with complex concepts[4] and (2) enhancement

[3]Term frequency/Inverted Document frequency.

[4]This kind of enhancement will not be discussed in this article because it is not directly involved in linguistic matching (c.f. (Bououlid and Vachon, 2007) for details).

of a data source with related concept names. To be more specific about the semantics of a given concept, the descriptive context is searched for entities which may relate to this concept. The following heuristics is proposed to identify these relations: ”Two concepts that are often brought up *one near the other* are likely to be closely related.”

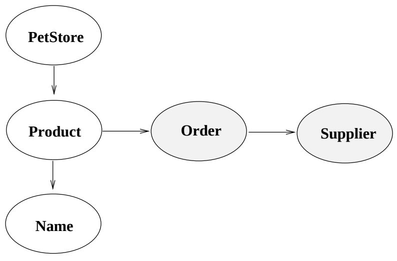

Figure 1: Graph of dependencies between concepts of the *Pet Store* application.

In the *Pet Store* application (our first case study), the database model reveals that *name* is an attribute of the *product* entity. In other respects, documentation files clearly show that *order* and *supplier* are concepts frequently appearing in the vicinity of the *product* concept. For each concept referenced by path s in the XML data source, a graph is constructed to trace presumed dependencies between concepts. We call such a graph a *concept dependency graph* (**CDG**). Let s be the XML path of a concept in the data source and let $Concepts(s)$ be the set of concepts composing s. A **CDG** for s is a graph that

1. links each $c \in Concepts(s)$ to its successor c' in s (with $c' \in Concepts(s)$). The position of a concept in the list thus obtained is called its *level*.
2. links each $c \in Concepts(s)$ to a list $elist(c)$ of enrichment concepts extracted from the data source's context. The position of a concept in $elist(c)$ is called its *rank*.

Figure 1 shows a **CDG** stressing the connection of the *product* concept with its *name* attribute and the *Pet Store* database to which it belongs. These concepts are vertically ordered on the figure. The **CDG** also shows the relation of *product* with two enriching concepts, *order* and *supplier*, which were often found close by in the data source's context. These concepts appear horizontally ordered.

The design of INDIGO's *Context Analyzer* (Figure 2) comprises two main modules, each being specialized in a specific type of information extraction. To be specific, it is composed of a *Concept Name Collector* and a *Complex Concept Extractor*, providing the necessary support for the two kinds of enhancement mentioned previously.

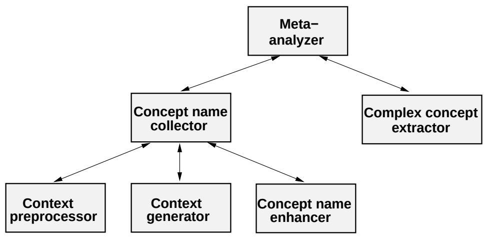

Figure 2: Architecture of the Context Analyzer.

As illustrated on Figure 2, these two analyzer modules are headed by a meta-analyzer which coordinates their respective tasks. At the lowest level, the architecture provides utility modules to pre-process, generate and enrich data sources with context information.

3.1.1 Meta-analyzer

This module explores the set of documents constituting the informational context of the data source. It classifies each of them into either the formal or informal category. Then it addresses formal documents to the *Complex Concept Extractor* and informal ones to the *Concept Name Collector*.

3.1.2 Concept Name Collector

Given a concept name of a data source, the *Collector* gathers from the descriptive context all names frequently surrounding it. The *Collector* chooses then the closest ones to enrich the concept name with them. Actually, the *Collector* executes the data source enrichment process in two main stages: a *context preprocessing* step followed by a *data source enhancement* procedure.

Context preprocessing. The *Collector* relies on two internal modules respectively named *Context Preprocessor* and *Context Generator* to generate what we called the *contextualized concept file* of a data source (abbreviated **CC-file**). This file is written in **XML**. For each concept name in the data source, it gathers a set of nouns surrounding the concept in the data source's context documents. These nouns are saved by the **CC-file** in some normalized form which prevents that a single concept appearing in different guises (e.g. *inventory* and *inventories*) be considered as two distinct nouns. Let us consider the *Pet Store* application for example, whose database contains a table named *SKU* (i.e. Stock Keeping Unit). The *Context Preprocessor* will thus parse the set of documents it was addressed and will gather all sentences containing the concept name *SKU*. Sentences are then filtered by the *Context Generator* to keep noun-type words in their normalized form. The order in which a word first appears in a sentence is of course preserved.

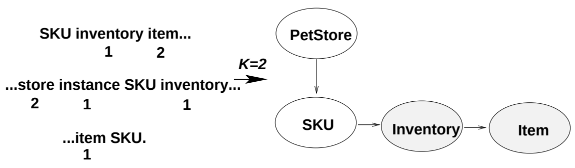

Figure 3: Enhancement of the SKU concept (generation of the corresponding part of the CDG).

Data source enhancement. The **CC-file** elaborated during the previous step is used by a third internal module of the *Collector*, called *Concept Name Enhancer*, to generate a **CDG** for each data source concept. Let us consider the enhancement of the *SKU* concept. Over each sentence of the **CC-file** relating to *SKU*, the *Concept Name Enhancer* applies a position marker which tags each name surrounding the *SKU* word with a distance value. This value indicates how many names separate the tagged name from the word *SKU*, be it on the left or on the right. The left part of Figure 3 shows the tagging process. From there on, each tagged name is collected together with its distance tag and its frequency rate. Tagged names are sorted in descending order of frequency rates and proximity to *SKU*. Best candidates are retained to enrich the *SKU* concept. A threshold value is used to limit the number of enriching concepts to be kept. In the previous example, given a threshold *K*=2, the names *item* and *inventory* are retained to enrich the *SKU* concept and are thus added to *SKU*'s **CDG**.

3.2 Mapper Module

After the enhancement of data sources with the newly extracted information, the *Mapper* module proceeds to the matching phase. Allowing for multiple matching strategies, its architecture is organized as a hierarchy of modules where internal nodes are called coordinators and where leaves represent aligners. Each aligner implements a specific matching strategy based on a particular similarity measure. For instance, a name-based aligner can be used to compute matches between concepts according to the similarity of their names. As for coordinators, they combine predictions of their respective children to elaborate their own matching proposal.

3.3 Context-based Alignment

Context-based alignment is executed as follows. First, a mapping is computed between two given data sources without considering their context. Then, for each pair of concepts a *context similarity factor* (**CSF**) is assessed by comparing their respective **CDG** (cf. Section 3.1). The **CSF** is a normalized value between 0 and 1 which is used to rate the augmentation of concepts' similarity by taking into account the similarity of their respective context.

To show how a **CSF** is calculated, let us consider the respective **CDG** of a source and of a target concept, SN and TN, as illustratred on Figure 4. SN_{ij} (resp. TN_{ij}) denotes the concept occupying level i and enrichment rank j in the **CDG** of the source concept (resp. target concept). Each row of a **CDG** denotes a context level, while each column denotes a concept ranking regarding relevance to the context (recall that rank 1 denotes the most relevant concept). The evaluation of the **CSF** between SN and TN is as follows.

First, we split up equally each list of enrichment concepts into some predefined number of subsets. Figure 5 depicts the splitting, into w boxes, of the enrichment concepts in a **CDG**. On a same level, boxes should contain almost the same number of enrichment concepts. Considering there are p_i concepts on level i and that $p_i = w * k_i + r_i$, the first r_i boxes will therefore contain $k_i + 1 = \lceil p_i/w \rceil$ concepts while the following $w - r_i$ others will have k_i of these. Let SB_i, with $i \in \{1, ..., w\}$, denote the i^{th} box in a source concept's **CDG**. $SB_{i,j}$ thus corresponds to the set of enrichment concepts of level j in box SB_i . Note that the CDG of the target concept, illustrated on Figure 4, has been splitted up exactly the same way as the **CDG** of the source concept. Then, similarity between boxes SB_{i_s} and TB_{i_t}, where $i_s, i_t \in \{1, ..., w\}$, is given by the following formula:

$$BoxSim(SB_{i_s}, TB_{i_t}) = \frac{\sum_{l=1}^{\min(n,m)} \lambda^l Inter(SB_{i_s,n-l+1}, TB_{i_t,m-l+1})}{\sum_{l=1}^{\min(n,m)} \lambda^l}$$

$Inter(SB_{i_s,n-l+1}, TB_{i_t,m-l+1})$ is a function returning 0 if $SB_{i_s,n-l+1} \cap TB_{i_t,m-l+1}$ is empty, and returning 1 otherwise. λ is a positive constant, superior to one, which is empirically settled.

Finally, the *context similarity factor* (**CSF**) evaluated between the respective **CDG** of a source and of a target concept is given by the following formula:

$$CSF = \frac{\sum_{i=1}^{w} \alpha^{w-|i-n(i)|} BoxSim(SB_i, TB_{n(i)})}{w\alpha^w}$$

where $n(i)$ is the nearest target box index to i such as $BoxSim(SB_i, TB_{n(i)}) \neq 0$.

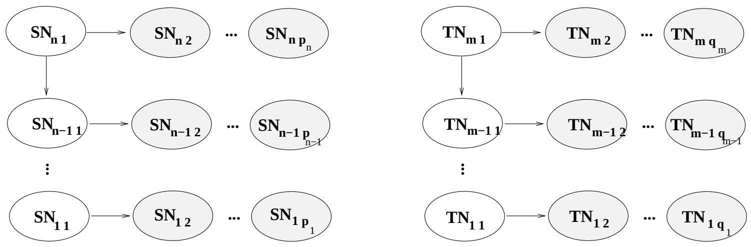

Figure 4: CDG of a source and of a target name.

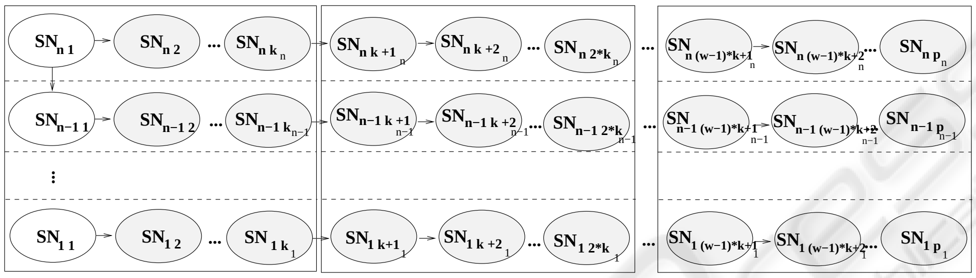

Figure 5: Splitting of a concept name's *CDG*.

4 TESTS AND EVALUATION

Two types of tests have been targeted: (1) the impact of contextualization on linguistic matching and (2) the comparison of INDIGO with the state of the art.

4.1 Impact of Contextualization

For this experiment, the *Mapper* was configured with five name-based aligners, that is one for each of the five well-known string metrics (i.e. Jaro-Winkler, Smith-Waterman, Levenshtein, Needleman- Wunch and Q-Gram) mentioned in section 2. Each aligner was individually tested on the first case study, i.e. the mapping of the *PetStore*'s data source over *eStore*'s, with and without contextualization being taken into account. Table 1 describes the results obtained when fixing the enrichment threshold to $K = 30$ and the context splitting parameter to $w = 5$. The two constants λ and α were respectively set to 4 and 2. Results show that data sources contextualizing has a positive impact on the performance of name-based aligners. In the case of the *QGram* distance, the *f-measure* of the alignment process increased by 127%.

4.2 Comparison with Other Semantic Matching Systems

INDIGO was confronted with both the Similarity Flooding algorithm and the V-Doc System. For this experiment, the *Mapper* module was configured with a single matching strategy, namely a sole aligner based on the *JaroWinkler* metric. Each system was executed over the two case studies thus computing: (1) the mapping of the *PetStore*'s data source over *eStore*'s and (2) the mapping of course data source of *Cornell university* over the one of *Washington university*. To be precise, two versions of the second case study were experimented on: a version over reduced course data sources and the original one. Corresponding test results are summarized in Table 2. In particular, the two last rows present the results obtained with INDIGO for both (1) non context-based and (2) context-based alignments.

In all alignment tests, INDIGO succeeded in improving its performance thanks to contextual data source enrichment. It did not globally surpass all the other systems but managed to get very close to the best one (i.e. SF) thanks to data source contextualization. Indeed, INDIGO's global performance was increased by 14% with contextualization, what brought it close to SF's results by less than 1.8%. Finally, it is worth noting that INDIGO did surpass all the other systems in the *PetStore/eStore* case study thanks to data source contextualization. This proves that INDIGO's context-based approach is particularly useful for matching data sources that are very heterogeneous.

Table 1: Performance variations due to data source contextualization observed in the mapping of *PetStore*'s data source over *eStore*'s using INDIGO.

	Non context-based			Context-based			Variation		
	prec.	rec.	f-m.	prec.	rec.	f-m.	prec.	rec.	f-m.
Jaro-Winkler	44.44	13.79	21.04	31.11	20.9	25	−30%	+51%	+19%
Smith-Waterman	34.78	18.39	24.05	39.62	24.14	30.00	+14%	+31%	+25%
Levenshtein	48.15	14.94	22.80	53.85	16.09	24.77	+16%	+7%	+10%
Needleman-Wunch	43.75	8.05	13.59	23.91	12.64	16.53	−45%	+57%	+22%
Q-Gram	66.67	9.20	16.17	60.53	26.44	36.80	−9%	+187%	+127%

Table 2: Comparison of INDIGO's performance with Similarity Flooding and the V-Doc matching results.

	PetStore/eStore			mini-cornell/mini-washington			cornell/washington			average		
	prec.	rec.	f-m.	prec.	rec.	f-m.	prec.	rec.	f-m.	prec.	rec.	f-m.
SF	41.66	14.92	21.97	1.0	79.41	88.52	60.0	72.22	65.54	34.22	55.51	58.67
V-Doc	66.66	2.98	5.71	65.62	61.76	63.63	17.36	46.29	25.25	49.88	37.01	31.53
Indigo(1)	44.44	13.79	21.04	83.87	78.79	81.85	34.51	73.58	46.99	54.27	55.38	49.96
Indigo(2)	31.11	20.9	25	84.38	81.82	83.08	58.06	67.92	62.61	57.85	56.88	56.9

5 CONCLUSIONS

This paper presents INDIGO, an innovative solution to linguistic matching. INDIGO relies on an architecture composed of two main modules: a *Context Analyzer* and a *Mapper* module. The *Context Analyzer* enriches data sources with semantic information extracted from artifacts belonging to their respective environments before delivering them to the *Mapper* for their effective alignment. Tests conducted over two case studies proved that data source contextualizing can improve the alignment process performance. In particular, INDIGO was compared to two renowned matching systems in the domain and outperformed them in the case of data source candidates presenting higher heterogeneity.

REFERENCES

A. Doan, P. D. and Halevy, A. (2003). Learning to match the schemas of data sources : a multistrategy approach. *Journal of Machine Learning*, 50(3):279–301.

Bououlid, I. and Vachon, J. (2005). 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)*, pages 445–448, Miami.

Bououlid, I. and Vachon, J. (2007). A context-based approach for for complex semantic matching. In *Proc. of CAiSE-Forum*, Trondheim, Norway.

B.T. Le, R. D.-K. and Gandon, F. (2004). On ontology matching problems - for building a corporate semantic web in a multi-communities organization. In *Proc. of the Int. Conf. on Enterprise Information Systems (ICEIS)*, volume 4, pages 236–243.

Do, H. and Rahm, E. (2002). Coma: a system for flexible combination of schema matching approaches. In *Proceedings of the 28th Conf. on Very Large Databases*.

Euzenat, J. and et. al (2004). State of the art on ontology alignment. part of a research project funded by the ist program of the commission of the european communities. Technical Report project number IST-2004-507482, Knowledge Web Consortium.

Hu, W., Cheng, G., Zheng, D., Zhong, X., and Qu, Y. (2006). The results of falcon-ao in the oaei 2006 campaign. In *International Workshop on Ontology Matching*.

J. Madhavan, P. B. and Rahm, E. (2001). Generic schema matching using cupid. In *In Proceedings of the 27th VLDB Conference*, pages 48–58, Roma, Italy.

K. Kotis, G. V. and Stergiou, K. (2004). Capturing semantics towards automatic coordination of domain ontologies. In *Artificial Intelligence: Methodology, Systems, and Applications*, volume 3192 of *LNCS*, pages 22–32.

McUmber, R. (2003). Developing pet store using rup and xde. Web Site.

Melnik, S., Garcia-Molina, H., and Rahm, H. (2002). Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In *Proc. International Conference on Data Engineering (ICDE'02)*, pages 117–128.

Microsystems, S. (2005). Java petstore. Web Site.

Noy, N. and Musen, M. (2001). Anchor-prompt: Using non-local context for semantic matching. In *In Proc. IJCAI 2001 workshop on ontology and information sharing*, pages 63–70, Seattle (WA US).

Y. Qu, W. H. and Cheng, G. (2006). Constructing virtual documents for ontology matching. In *Proceedings of the 15th International World Wide Web Conference*.