NORMS: an automatic tool to perform schema label normalization

Serena Sorrentino, Sonia Bergamaschi, Maciej Gawinecki

DII, University of Modena and Reggio Emilia
via Vignolese 905, 41015 Modena, Italy
firstname.lastname@unimore.it

***Abstract*— Schema matching is the problem of finding relationships among concepts across heterogeneous data sources (heterogeneous in format and structure). Schema matching systems usually exploit lexical and semantic information provided by lexical databases/thesauri to discover intra/inter semantic relationships among schema elements. However, most of them obtain poor performance on real world scenarios due to the significant presence of "non-dictionary words". Non-dictionary words include compound nouns, abbreviations and acronyms. In this paper, we present *NORMS* (NORMalizer of Schemata), a tool performing *schema label normalization* to increase the number of comparable labels extracted from schemata[1].**

I. Introduction

Schema matching is a critical task in many areas such as data integration, data warehousing, e-business and semantic web applications. Most existing schema matching systems try to find semantic relationships on the basis of the lexical and semantic information associated with the schema labels by exploiting external resources such as thesauri or lexical databases [1], [2] (e.g. WordNet (WN) [3]).

The strength of a thesaurus (like WN) is the presence of a wide network of semantic relationships among word meanings (synsets in WN) such as synonymous relationships (defined between two labels that are synonymous) and hyponym/hypernym relationships (defined between two labels where the one is more specific/general then the other). Starting from these semantic relationships, it is possible to infer a corresponding semantic network of relationships among the labels of different schemata. The weakness of a thesaurus is that it does not cover, with the same detail, different domains of knowledge and many domain dependent words, as "Non-Dictionary Words" (NDWs), may not be present in it. NDWs include abbreviations, acronyms and Compound Nouns (CNs). The results of schema matching systems (i.e. the discovered relationships) is strongly affected by the presence of NDWs in schemata. Thus, a process of *schema label normalization*—able to expand acronyms and abbreviations (in the following *abbreviations*) and to semantically enrich a thesaurus with CNs—is needed.

Without schema label normalization two kinds of errors may occur during the schema matching process: discovering a wrong relationship (i.e. *false positive relationship*) and missing a right relationship (i.e. *false negative relationship*). The first one decreases precision, while the second one decreases recall[2]. To give an intuition of the problem, let us consider the example in Figure 1, which shows the schemata of two simple data sources (a relational database and an XML file) to be matched. These schemata contain many non-dictionary CNs (e.g. "CustomerName") and abbreviations (e.g. "QTY"). Let us consider the two labels "amount" of the "PurchaseOrder" schema and "QTY" (abbreviation of "quantity") of the "PO" schema. Without expanding the second label we cannot discover that there exists a synonymous relationship between both labels; this leads to a false negative relationship. Moreover, let us consider the two labels "CustomerName" and "CLIENTADDRESS" in the sources "PurchaseOrder" and "PO" respectively. Both CNs do not have an entry in WN. For discovering a relationship between these two labels, we might consider the WN meanings associated to the single words "Customer" and "Name", "CLIENT" and "ADDRESS" separately. In this way, we discover a synonymous relationship between them, because the terms "Customer" and "CLIENT" share the same WN meaning. However, this is a false positive relationship because those CNs represent "semantically distant" concepts.

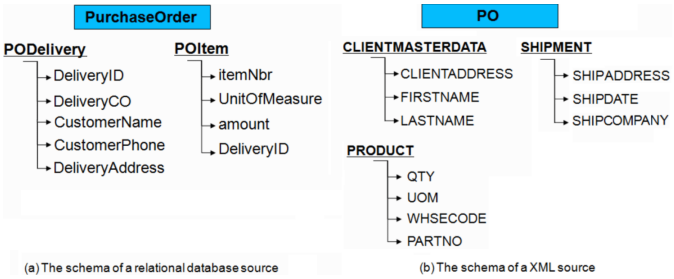

Fig. 1. Graph representation of two schemata with elements containing abbreviations and CNs.

A manual process of label normalization is laborious, time consuming and itself prone to errors. Furthermore, when the number of schemata is very large or dynamically growing, a

[1]This work was partially funded by the "Searching for a needle in mountains of data!" project funded by the Fondazione Cassa di Risparmio di Modena within the Bando di Ricerca Internazionale 2008 (http://www.dbgroup.unimo.it/keymantic).

[2]Let A be the number of relationships reported by the schema matching system over the schemata, B the number of correct relationships within A and C the total number of correct relationships in the schemata, precision is defined as B/A and recall as B/C.

manual process is not feasible. To the best of our knowledge, in literature, there are no automatic or semi-automatic tools that support designers in the label normalization process. Most schema matching systems overlook the presence of NDWs or assume they have been a-priori solved. Although, [4], [5], [6] address the issues of abbreviations and CNs, they provide partial solutions or solve the problem in a non-scalable way. Another way to overcome the problem of limited amount of useful schema information and meaningless schema labels (as in the case of presence of several NDWs) is the use of *instance-level schema matching* techniques [1], which exploit only data instances, and do not consider the schema information at all. The main drawback of these approaches is that there are several situations where data instances are not available due to security reasons or restricted authorizations.

Starting from our previous work on label normalization in the context of data integration [7], we present NORMS, a stand-alone tool performing schema label normalization. The main innovative features of NORMS are: (1) it allows to expand abbreviations and to enrich WN with CNs in an automatic way; (2) it provides a GUI that supports the designer during the normalization process and allows him/her to enhance the automatic results by correcting potential errors; (3) it provides, as additional feature, the possibility to automatically annotate the schema elements w.r.t. WN; (4) its output can be exploited by several matching applications.

In the following, we describe the NORMS architecture and we quantify the normalization quality and the amount of saved manual effort by applying NORMS on different real world data sets.

II. NORMS OVERVIEW

NORMS is composed of four main modules (see Figure 2): (1) Wrapper, (2) Schema Label Normalizer, (3) Lexical Annotator, (4) Output Generator. A modular architecture enables reuse and composition of single functionalities. In the following, we describe the input and output of each module and the interaction among them. A detailed description of the methods implemented in these modules can be found in [7], [2].

Wrapper extracts the schemata to be normalized by the NORMS tool. Using the GUI the designer can upload one or more schemata from a number of sources: relational database (SQL Server, MySQL, ODBC and Oracle), XML and OWL. Each schema is logically converted into the internal object language ODL_{I^3}[3].

Schema Label Normalizer processes schemata from the wrapper in the following phases: (1) Label Preprocessing, (2) Abbreviation Expansion and (3) Creation of New CN Meanings. Figure 3 illustrates an example of the normalization process applied on the label "DeliveryCO", showing the input and output of each phase.

Phase 1. Label Preprocessing automatically selects and tokenizes the labels to be normalized. The tokenized labels

[3]ODL_{I^3} is an extension of the standard ODMG-ODL (http://www.odbms.org) to deal with semi-structured data sources, and its semantic is based on a description logic (including OWL-DL).

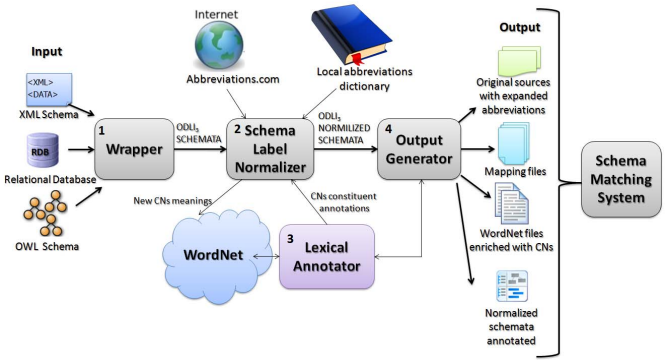

Fig. 2. NORMS architecture.

are further classified into three groups: abbreviations, CNs, and CNs containing abbreviations. For instance, the schema label "DeliveryCO", as shown in Figure 3, is tokenized in two single words "Delivery" and "CO" and classified as a CN that contains the abbreviation "CO". By using the GUI the designer can modify the automatic results. In particular he/she can: select if a label has to be normalized, correct tokens and indicate whether a label is a CN and/or contains abbreviations.

Phase 2. Abbreviation Expansion automatically performs the following steps: (1) searches for potential expansions for a given abbreviation; (2) scores the relevance of the potential expansions; (3) combines the scores and suggests the top-scored expansion. In order to identify potential expansions, NORMS exploits four *expansion resources*: Local Context (LC), Complementary Schemata (CS), Online abbreviation Dictionary (OD), and Local abbreviation Dictionary (LD). For example, for the "CO" abbreviation contained in "DeliveryCO", the local context is its schema "PurchaseOrder" and the schema "PO" is the complementary schema. LD is initially bootstrapped with standard schema abbreviations from schema design guideliness for the OTA standard[4]. The GUI allows the designer to enrich the dictionary with non-standard abbreviations that are *specific* for the schemata under normalization. As OD NORMS uses "Abbreviations.com". For instance, for the abbreviation "CO" the following expansions are identified: (a) from OD {"Company", "Colorado", and "Check Out"} (b) from LC no results, (c) from CS schemata {"Company"}. For each identified expansion exp_i, NORMS computes a combined score $sc(exp_i) \in [0, 1]$ and ranks the expansions accordingly. NORMS suggests the top-score expansion but it also allows designers to select, by using the GUI, the correct expansion among also less scored ones. As shown in Figure 3, for the abbreviation "CO" the tool suggests the expansion "Company". The score $sc(exp_i)$ is computed by combining scores from the single resources:

$$sc(exp_i) = \alpha_{LD} \cdot sc_{LD}(exp_i) + \alpha_{CS} \cdot sc_{CS}(exp_i) + \alpha_{LC} \cdot sc_{LC}(exp_i) + \alpha_{OD} \cdot sc_{OD}(exp_i)$$

[4]OpenTravel Alliance Xml schema for travel industry. Available online at http://www.opentravel.org/.

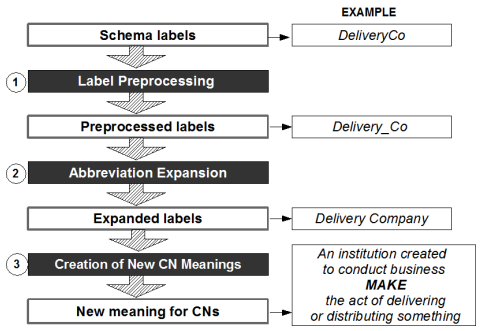

Fig. 3. Example of application of NORMS on a schema label.

where sc_{LD}, sc_{CS} and sc_{LC} are equal to 1 if exp_i is found in the given resource or 0 otherwise; and sc_{OD} is computed as described in [7]. $\alpha_{LD} + \alpha_{CS} + \alpha_{LC} + \alpha_{OD} = 1$ are the weights of resource relevance. NORMS uses as default weights $\alpha_{LD} = 0.4$, $\alpha_{LC} = 0.3$, $\alpha_{CS} = 0.2$ and $\alpha_{OD} = 0.1$. These weights were selected after an evaluation of the abbreviation expansion phase on several real schemata. The weights can also be manually set up by the designer through the GUI.

Phase 3. Creation of New CN Meanings. A CN is a word composed by more than one word called CN *constituents*: a head noun and one or more modifier (adjective or noun) which restrict the meaning of the head. It is used to denote a concept, and can be interpreted on the basis of the meanings of its constituents. This phase can be summed up into two automatic steps: (1) CN interpretation via semantic relationships; and (2) creation of a new CN meaning in WN.

CNs are interpreted to enrich WN with non-dictionary CNs. The interpretation of a CN is the task of determining the semantic relationships holding among its constituents. NORMS annotates each CN constituent with its corresponding WN synset, by using the Lexical Annotator module (described in the following) and then performs automatic CN interpretation by using the set of nine semantic relationships (CAUSE, HAVE, MAKE etc.) defined by Levi in [8]. For example, (see Figure 3) let us consider the label "DeliveryCompany", its constituents get annotated w.r.t. WN and then the CN is interpreted with the MAKE relationship: "Company" MAKE "Delivery".

At the end NORMS creates a new WN meaning for the given CN. It creates the *gloss*[5] to be associated with the CN, starting from the relationship associated with it and exploiting the glosses of the CN constituents. For instance, the glosses of the constituents "Company" and "Delivery" are joined together according to the relationship MAKE. Then, the new meaning for the target CN is inserted in the WN thesaurus by creating WN relationships with the other WN meanings. By using the GUI, the designer can correct the automatically created meaning. In particular, he/she can: specify the correct Levi relationship for the interpretation; modify the gloss; add, delete or edit the relationships between the new and other WN meanings.

[5]A gloss is the description in natural language of the meaning of a word.

Lexical Annotator uses the CWSD (Combined Word Sense Disambiguation) algorithm [9] to annotate each label with a corresponding WN synset. CWSD exploits the structure of schemata together with its element lexicon in order to discover the "right meaning" (synset in WN terminology) to be associated to each label. As described before, this module is used to perform CN interpretation but can be used also to automatically annotate the normalized schemata. This additional feature is important as some schema matching applications can take advantage also from the annotation information. The GUI allows the designer to correct the automatic annotations.

Output Generator exports all the results of NORMS in *portable* formats, so they can be directly used as input in schema matching systems or several other applications that utilize the semantic and/or lexical information associated to schema labels. Those applications include: data exchange, web interface integration, ontology alignment, data warehousing, web service integration etc. The module generates the following files: (1) *the original sources with expanded abbreviations*: a file for each source where each abbreviation has been replaced with the selected expansion; (2) *the mapping files*: a textual file for each source that contains the mapping between the original schema elements labeled with abbreviations and the new expanded labels; (3) *the WN files enriched with CNs*: WN files updated with the new created CN meanings[6]; and (4) *normalized annotated schemata*: a file containing the ODL_{I^3} normalized schema annotated by the Lexical Annotator module w.r.t. WN. By using the GUI the designer can choose to export only the files he/she needs.

III. Performance and Human Effort Evaluation

We extended the initial evaluation of our normalization method presented in [7], by testing NORMS on five real world data sets: GeneX, Mondial, Amalgam, TCP-H and PurchaseOrder[7]. We performed the evaluation starting from the default configuration where the designer does not perform any initialization tasks (i.e. no specific abbreviations manually inserted in LD and default abbreviation weights). NORMS performed well on the whole set of NDWs present in the schemata: it achieved 76% of precision and 66% of recall and then 71% of F-measure on average. However, in all data sets the recall value was affected by the presence of NDWs, such as "In Proceedings" or "sea 2" that our method is not able to normalize, as containing digits and prepositions (future work).

As the goal of NORMS is to provide an effective tool to improve the performance of semantic and lexical schema matching applications, we used the output of NORMS as input of the MOMIS data integration system [2]. The use of NORMS brought a significant improvement in the number of correct semantic relationships (synonymous and hypernym/hyponym relationships) discovered among heterogeneous

[6]Precisely, *index.noun* an *data.noun*, the files containing all the synsets and words for the noun syntactic categories in WN: http://wordnet.princeton.edu/man/wndb.5WN.html

[7]All the data sets are publicly available at http://queens.db.toronto.edu/project/clio/ and http://dbs.uni-leipzig.de/Research/coma_index.html.

schemata: from 62% to 82% in precision and from 53% to 77% in recall (and then from 57% to 79% of F-measure). Without schema label normalization, MOMIS obtained: a low precision due to the presence of a lot of false positive relationships; a very low recall as a lot of relationships between schema elements labeled with abbreviations were not discovered.

One of the major requirement for an automatic or semi-automatic tool is to reduce the amount of manual work. For this reason, we evaluated the amount of human effort that is possible to save by using NORMS. In general, human effort may be divided in *pre-processing* (i.e. system parameter configuration) and *post-processing* effort (i.e. in our case, the correction of the automatic normalization results). Starting from the default configuration, we may assume the pre-processing effort equal to zero.

To approximate how much post-processing effort is possible to save by using NORMS, we adopted the *Accuracy* measure used in [10] to estimate post-matching effort. It can be defined as $Accuracy = 1 - \frac{|a|+|c|}{|a|+|b|} = Recall * (2 - \frac{1}{Precision})$, where, in our case, a is the number of false negatives (i.e. unnormalized labels requiring normalization), b is the number of true positives (i.e. correctly normalized labels) and c is the number of false positives (i.e. incorrectly normalized labels). In this way, we obtained an average Accuracy, over the five data sets, equal to 45%. However, according with [10], Accuracy is very pessimistic if compared to F-measure (71%) and NORMS may be "more useful" than what this measure predicts: first of all, Accuracy does not address semi-automatic normalization, in which the user interacts with the GUI by adjusting the single results step by step; moreover, it assumes equal effort to remove false positives and to add false negative normalizations, although in our case, the effort to remove or to correct a false positive normalization (e.g. the selection a wrong expansion for a given abbreviation) is considerably less than the effort to add a false negative (e.g. to manually add a new CN in WN).

IV. Demonstration

We have implemented the NORMS as an Adobe AIR[8] desktop application by using Adobe Flex 4 (for the GUI) and Java (for the business logic). We chose Flex technology because it makes creation of complex and well designed GUI an easy task and it can be easily deployed as a front end of Web application for normalization (our future work). Finally, Java and Flex applications are platform-independent.

Figure 4 shows a screenshot of the GUI of NORMS. It is divided in three main parts: an *action menu* (on the top) providing the list of actions for the normalization process, a *schema panel* with an intuitive tree representation of the wrapped schemata (on the left), and a set of *tabs*, one for each normalization phase (on the right). The whole normalization process requires a low level of human interaction, and the main operations are provided through these tabs that allow a designer to run and control the process and the results.

[8]http://www.adobe.com/products/air/

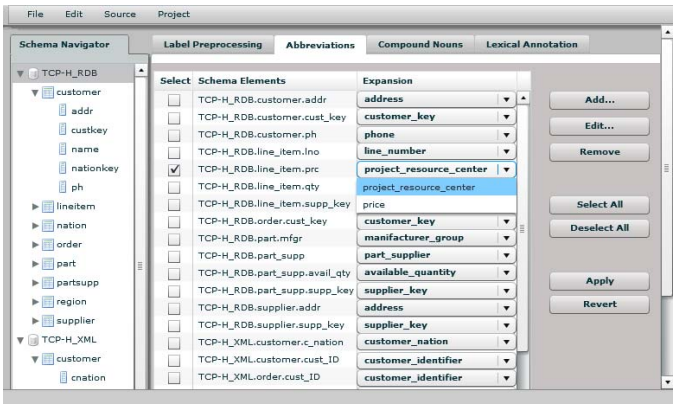

Fig. 4. A NORMS screenshot.

The goal of our demonstration is to show how NORMS can be used in a real-world scenario to improve the results of schema matching. For this reason, we will use the real-world schemata used during the evaluation (see Section III).

The demonstration will start by uploading the schemata of the sources, thus taking advantage of the several standard formats supported. Next, we will show how the designer can manually configure parameters like the expansion resource weights (see Section II). We will run the normalization process in two modes: *fully automatic* and *interactive*. The latter allows to run normalization phase by phase and control the output of each phase. We will show how using the GUI a designer can edit the outcome of the tool: add, delete and update results for each single label. For instance, in the tab "Abbreviations", shown in Figure 4, the tool allows the designer to select one of suggested expansions or insert a new one. Finally, we will demonstrate how NORMS improves the performance of schema matching, by running the MOMIS data integration system on both original and normalized schemata.

References

[1] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *VLDB J.*, vol. 10, no. 4, pp. 334–350, 2001.

[2] S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini, "Semantic Integration of Heterogeneous Information Sources," *Data & Knowledge Engineering*, vol. 36(1), pp. 215–249, 2001.

[3] G. A. Miller, "WordNet: A Lexical Database for English," *Communications of the ACM*, vol. 38, pp. 39–41, 1995.

[4] J. Madhavan, P. A. Bernstein, and E. Rahm, "Generic Schema Matching with Cupid." in *VLDB*, 2001, pp. 49–58.

[5] L. Ratinov and E. Gudes, "Abbreviation Expansion in Schema Matching and Web Integration," in *WI '04*, 2004, pp. 485–489.

[6] X. Chai, M. Sayyadian, A. Doan, A. Rosenthal, and L. Seligman, "Analyzing and revising data integration schemas to improve their matchability," *PVLDB 2008*, pp. 773–784.

[7] S. Sorrentino, S. Bergamaschi, M. Gawinecki, and L. Po, "Schema normalization for improving schema matching," in *ER*, 2009, pp. 280–293.

[8] J. N. Levi, *The Syntax and Semantics of Complex Nominals*. New York: Academic Press, 1978.

[9] S. Bergamaschi, L. Po, and S. Sorrentino, "Automatic annotation for mapping discovery in data integration systems," in *SEBD*, S. Gaglio, I. Infantino, and D. Saccà, Eds., 2008, pp. 334–341.

[10] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching." in *ICDE*, 2002, pp. 117–128.