Automatic Normalization and Annotation for Discovering Semantic Mappings

Sonia Bergamaschi, Domenico Beneventano, Laura Po, Serena Sorrentino

Department of Information Engineering
University of Modena and Reggio Emilia, Italy
name.surname@unimore.it

Abstract. Normalization and lexical annotation methods, developed in the context of matching systems, have proven to be effective for the discovery of lexical relationships among schemata. We will show how these methods are applicable and effective in the context of Semantic Resource Framework to mine the semantics of a web service interface and to discover mappings between them.

Key words: lexical relationships, probabilistic annotation, word sense disambiguation, label normalization, semantic resource framework

1 Introduction

This chapter will discuss the applicability of *normalization* and *lexical annotation* methods, developed in the field of schema matching, in the context of web service interfaces. The lexical annotation of a schema element is the explicit assignment of its meanings w.r.t. a lexical resource. Normalization (also called *linguistic normalization* [14]) is the reduction of the label of a schema element to some standardized form that can be easily recognized.

Starting from our previous works in the context of data integration [5, 21, 26], we propose to apply normalization and annotation methods to mine the semantics of a service, exposed through its interface and to discover connection patterns among web services.

In Natural Language Processing, Word Sense Disambiguation (WSD) is the process of identifying which sense of a word (i.e. meaning) is used in a sentence, when the word has multiple meanings (polysemy). We describe our probabilistic lexical annotation method, which automatically associates one or more meanings to schema elements w.r.t. the lexical resource WordNet (WN) [13], by exploiting a Word Sense Disambiguation (WSD) algorithm, called PWSD (Probabilistic Word Sense Disambiguation) [21]. The accuracy of lexical annotation is affected by labels which are non-dictionary words, such as Compound Nouns (CNs), acronyms and abbreviations which are very frequent on real-world schemata and web service interfaces. We addressed this problem by devising a method to normalize schema labels which is able to semi-automatically expand abbreviations and to *properly* lexically annotate CNs by creating new WN meanings.

Starting from the lexical annotation of schema elements, we can discover lexical relationships between them, on the basis of the relationships defined in WN between

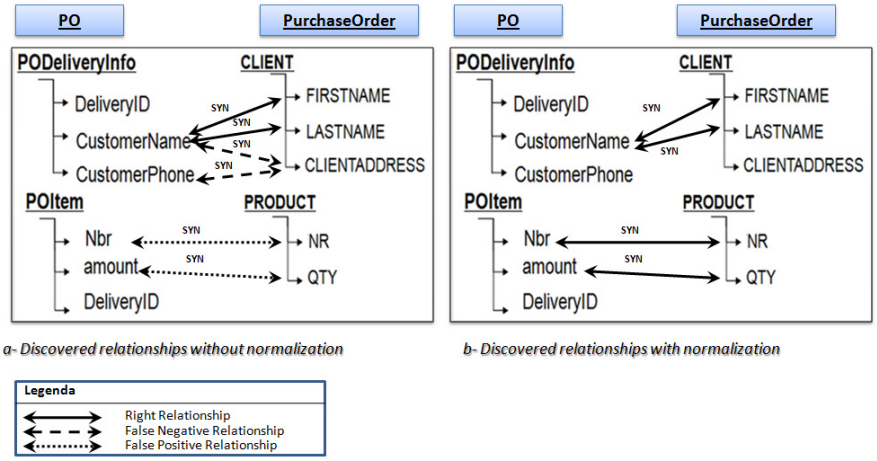

Fig. 1. Example of discovered lexical relationships without (a) and with (b) normalization.

their meanings (synsets in WN terminology). Traditional schema matching methods based on string distance techniques [10] do not permit to automatically discover that there exists, for example, a synonym relationship between the two schema elements "amount" and "quantity", as their labels share only few characters. Instead, by using our method, we are able to: (1) automatically annotate these schema elements with the corresponding WN meanings; (2) discover a synonym relationship among them, as they share the same meaning in WN (i.e. the synset "how much there is or how many there are of something that you can quantify").

Moreover, our normalization method improves the quality of semantic mappings by reducing the number of discovered *false positive/false negative relationships*. Figure 1 shows two schemata that need to be mapped/integrated, and compares the relationships discovered with and without normalization. Let us consider, for example, the two schema elements "CustomerName" and "CLIENTADDRESS", respectively, in the source "PurchaceOrder" and "PO", shown in Figure 1(a). If we annotate separately the terms "Customer" and "Name", and "CLIENT" and "ADDRESS", then we might assume a SYN relationship between them, because the terms "Customer" and "CLIENT" share the same WN meaning. In this way, a false positive relationship is discovered because these two CNs represent "semantically distant" schema elements.

Furthermore, if we consider the two corresponding schema labels "amount" and "QTY" (abbreviation for "quantity"), without abbreviation expansion we cannot discover that there exists a SYN relationship between the elements "amount" and "QTY".

In this chapter, we describe the normalization and annotation methods w.r.t. a generic object schema, which may be either a set of data sources or a set of web services (section 2). In Section 3, an example of application of the methods on the Semantic Resource Framework model is shown. Some related works are described in Section 4. Finally, in Section 5, we make some concluding remarks.

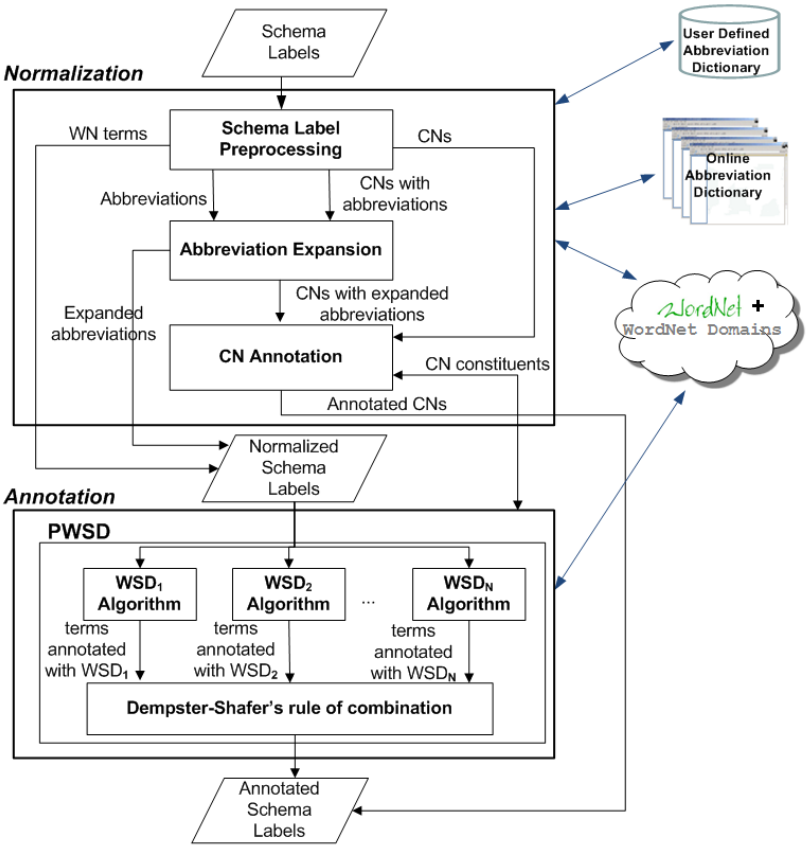

Fig. 2. Overview of the schema label Normalization and Annotation methods.

2 Normalization and Annotation of a Conceptual Schema

To describe the normalization and annotation methods, we use a generic Conceptual Modeling Language (CML), which contains common aspects of most semantic data models, UML, ontology languages, such as OWL, and description logics [1].

In the sequel, we use S to denote a $schema$ prescribed by the generic CML. Specifically, the language allows the representation of $classes$ (unary predicates over individuals), and $attributes$, which can be $simple$ (binary predicates relating individuals with values such as integers and strings) or $complex$ (binary predicates relating individuals).

Let $S = \{C_1, C_2, ..., C_n\}$ be a finite set of classes, where each class, $C = \{A_1, A_2, ..., A_n\}$, is described by a finite set of attributes. A simple attribute in the class $Hotel$, representing an hotel reservation web site, may be $name$, while a complex attribute may be $address$; $address$ refers to a class $Address$ in the same schema that combines the information of $street$, $number$, $city$, $zipcode$, and $country$.

Attributes may be subject to constraints such as, cardinality constraints. Such constraints do not influence our annotation and normalization methods and therefore are not taken into account in our generic CML.

Classes are organized in a familiar is-a hierarchy. In the previous example, the class $Hotel$ may be defined as a sub-class of $Service$.

Normalization and annotation methods regard classes and attributes of a schema, referred to as *schema elements* from now on. Each schema element has a name, referred to as *label* from now on.

Definition 1 (Lexical Annotation). *The lexical annotation of a schema element is the explicit assignment of its meanings w.r.t. a lexical resource.*

We define the lexical annotation as the connection of a schema element with its meanings defined in a lexical resource. However from now on, we will also make reference to the annotation of the label of a schema element.

The lexical resource we employ in our methods is WN. WN groups English words into sets of synonyms, called synsets. Each synset represents a distinct concept and is further clarified with a short defining gloss (i.e. a definition and optional example sentences). WN records the various semantic relationships between these synonym sets. These relationships vary based on the type of word; for the syntactical category of nouns, they include:

- hypernyms: Y is a hypernym of X iff every X is a (kind of) Y (person is a hypernym of student, because every student is a member of the larger category of persons);
- hyponyms: Y is a hyponym of X iff every Y is a (kind of) X (student is a hyponym of person);
- holonym: Y is a holonym of X iff X is a part of Y (building is a holonym of window);
- meronym: Y is a meronym of X iff Y is a part of X (window is a meronym of building).

We focus our work on the development of an automatic lexical annotation method able to select the synsets that cover the meaning of a schema element.

Thanks to the WN network of relationships, after the application of the annotation method we can *discover lexical relationships among schema elements*.

Lexical relationships are defined between classes and attributes, and are specified by considering class/attribute labels. Formally, a lexical relationship can be defined as:

Definition 2 (Lexical relationship) *Let t and s be two schema elements and $t_{\#i}$ and $s_{\#j}$ annotations assigned to t and s respectively. A lexical relationship is defined as the triple $< t_i, t_j, R >$ where R defines the type of the relationship between t_i and t_j. The types of lexical relationship are:*

- *SYN (Synonym-of): defined between two elements whose meanings are synonymous (they correspond to a WN synonym set), formally*

$$t \textit{ SYN } s \textit{ iff } \exists\, t_{\#i} \equiv s_{\#j}$$

– *BT (Broader Term): defined between two elements where the meaning of the first is more general than the meaning of the second (the opposite of BT is NT, Narrower Term), (it corresponds to a WN hypernymy/hyponymy relationship)), formally*

$$t \; BT \; s \; iff \; \exists \; t_{\#i} \; hypernym \; of \; s_{\#j}$$

– *RT (Related Term): defined between two elements whose meanings are related in a meronymy hierarchy (it corresponds to a WN meronymy relationship, i.e. part-of relationship), formally*

$$t \; RT \; s \; iff \; \exists \; t_{\#i} \; meronym \; of \; s_{\#j}$$

Let us suppose we want to automatically lexically annotate (annotate in the following) the schema element "address". In WN the noun "address" has eight different meanings, including very similar ones such as "written directions for finding some location; written on letters or packages that are to be delivered to that location" or "a sign in front of a house or business carrying the conventional form by which its location is described". We might generate a single (forced) annotation for each word (as done by many WSD approaches proposed in the literature). However, in many cases, choosing a single annotation would be difficult even for a human annotator: generating more than one probabilistic annotation comes to be a good solution that avoids loosing semantic information.

Uncertainty is an intrinsic feature of automatic and semi-automatic annotation methods and provides a quantitative indication of the quality of the result. In our method, uncertainty is qualified as probability values related to annotations w.r.t. WN.

The strength of a thesaurus like WN is the presence of a wide network of semantic relationships among meanings. Its main weakness is that it does not cover different domains of knowledge with the same level of detail and that many domain dependent terms (called *non-dictionary words*) may not be present in it. Non-dictionary words include CNs, abbreviations and acronyms (from now on, these last two will be referred to simply as "abbreviations"). The result of automatic annotation is strongly affected by the presence of these non-dictionary words in schemata, thus *label normalization* is needed. With label normalization, we mean the process of abbreviation expansion and CN annotation through the creation of new WN meanings.

Definition 3 *(Abbreviation expansion). Let AB be an abbreviation (or short form), abbreviation expansion is the task of finding a relevant expansion (long form) for the given abbreviation AB*[1].

Definition 4 *(CN annotation). Let CN be a non-dictionary compound noun constituted of more words (its constituents). The annotation of a CN is the task of creating a new WN synset starting from the annotations of its constituents.*

[1] The long form that is extracted through abbreviation expansion may not be an entry in WN. This issue remains an open problem. For the moment, we limit ourselves to examine long forms which have an entry in WN (e.g. the long form "Number" for the abbreviation "Nbr") or that correspond to CNs (e.g. the long form "Purchase Order" for the abbreviation "PO").

In the following, we describe normalization, annotation and relationship discovery in detail. In the end of this section, we show some results in term of performance of the methods.

2.1 Normalization

As shown in Figure 2, the schema label normalization method [26] consists of three steps: (1) schema label preprocessing, (2) abbreviation expansion and (3) CN annotation. The input of schema label preprocessing is the set of schema element labels. During this phase, we automatically select the labels to be normalized. The output of this module are the tokenized labels classified into four groups (as shown in Figure 2): *WN terms* (i.e. labels having an entry in WN which do not need normalization, e.g. "Airport") *abbreviations* (e.g. "FLTNO"), *CNs* (e.g. "DepartureAirport"), and *CNs containing abbreviations* (e.g. "ARRAirport").

The abbreviation expansion step is applied on all the schema labels classified as abbreviations or CNs with abbreviations. During this step, each abbreviation is expanded with the most relevant long form by using the knowledge provided by the schema and abbreviation dictionaries. Our method exploits the online abbreviation dictionary Abbreviations.com[2], particularly useful for expanding domain standard abbreviations, and a user-defined dictionary. Since real-world schemata often use application-specific codes (e.g. "X09CCDE") that will not appear in any public dictionary, the designer may enrich the user-defined dictionary with such abbreviations. The user-defined dictionary is initially bootstrapped with schema standard abbreviations and for our example by using the OTA standard[3].

The CN annotation is applied on all the schema labels classified as CNs or CNs with expanded abbreviations. In particular, we focus on a category of CNs called *endocentric*. Endocentric CNs consist of a head (i.e. the categorical part that contains the basic meaning of the whole CN) and one or more modifiers, which restrict the meaning of the head. An endocentric CN exhibits a *modifier-head structure*, where the head noun occurs always after the modifiers. Endocentric CNs are often not included in dictionaries, but they can be interpreted by using the knowledge about their constituents.

This step can be summed up into three sub-steps: (1) CN constituent disambiguation; (2) CN interpretation via semantic relationships; and (3) creation of a new CN synset in WN.

During the first sub-step the constituents of a CN are automatically annotated w.r.t. WN by applying the PWSD algorithm [21] (described in the following) which assigns a set of probabilistic annotations to each constituent. Then, starting from these annotations we perform *CN interpretation*. The interpretation of a CN is the task of determining the semantic relationships holding among its constituents. In particular, we perform automatic CN interpretation by using the set of nine semantic relationships defined by Levi in [16]: CAUSE ("flu virus"), HAVE ("college town"), MAKE ("honey bee"), USE ("water wheel"), BE ("chocolate bar"), IN ("mountain lodge"), FOR ("headache pills"),

[2] `http:\\www.abbreviations.com`

[3] OpenTravel Alliance XML schema for the travel industry. Available online at `http://www.opentravel.org/`.

FROM ("bacon grease"), and ABOUT ("adventure story"). At the end, we establish a new WN synset for the CN: first, we derive the gloss starting from the discovered Levi relationship and by exploiting the glosses of the CN constituents; then, the new meaning for the CN is inserted in the WN thesaurus by automatically creating a hypernym (and the opposite hyponym) relationship between the new synset and the synset of the CN head, and a generic (*Related term*) (which corresponds to the WN relationships *member meronym*, *part meronym*, *substance meronym*) between the new synset and the synset of the modifier.

However, the insertion of these two relationships is not sufficient; it is also necessary to discover the relationships of the new inserted meaning w.r.t. the other WN synsets. To this end, we use the WNEditor tool [4] to create/manage the new synset and to set relationships between it and the existing WN ones. As the final goal of our method is to produce a set of probabilistic annotations for each schema elements, we compute the probability associated to the new synset as the product of the probability values of the individual constituent annotations[4].

2.2 Probabilistic Lexical Annotation

As shown in Figure 2, the output of the normalization method will be the input of the annotation method. Lexical annotation is performed by PWSD, an automatic algorithm that combines several WSD algorithms. In this way, the process is not affected by the effectiveness of a single WSD algorithm in a particular context or application domain. PWSD satisfies three important constraints: (1) it is an automatic technique (only a few configuration settings are required), (2) it is flexible (i.e., it can combine any set of WSD algorithms[5]), and (3) the output of the method does not commit to an exact synset for a term under consideration, but to a set of possible senses that represent the term. We use the Dempster-Shafer theory of evidence [24, 20] to combine annotation outputs obtained by the WSD algorithms. By using the Dempster-Shafer's theory of evidence, PWSD associates a probability value to each sense selected to disambiguate a term; this value shows the uncertainty of the disambiguation process.

Given a schema element t, the PWSD algorithm associates a set of probabilistic annotations to t:

$$PM(t) = \{< t_{\#i}, P(t_{\#i}) >, ..., < t_{\#n}, P(t_{\#n}) >\}$$

Eventually, to minimize the introduction of errors, probabilistic annotations with a probability value under a certain threshold can be filtered.

2.3 Probabilistic Lexical Relationship Discovery

Once we have obtained annotations for schema elements, we can use the probability distributions over the set of possible meanings (i.e. the output of PWSD) to infer prob-

[4] We assume that the probabilities being combined are independent. This assumption is not usually hold, however, factoring out dependencies in WSD context is extremely difficult as they are usually hidden [22].

[5] At present, we combine five WSD algorithms.

abilistic lexical relationships among the object and attribute terms. As stated in Definition 2, the lexical relationships SYN (Synonym-of), BT (Broader Term)/ NT (Narrower Term) and RT (Related Term) are defined on the basis of the semantic relationships defined in WN among the meanings of two schema elements. To each lexical relationship, it is assigned a probability value that depends on the probability value of the meanings under consideration for the schema elements and it is determined by the formula of the joint probability.

More formally, given two schema elements t and s with the related probabilistic annotations, $PM(t)$ and $PM(s)$, a probabilistic lexical relationship $LexRel$ between t and s with probability P, denoted by

$$< t, LexRel, s, P >$$

is defined iff

1. $\exists < t_{\#i}, P(t_{\#i}) > \in PM(t), \exists < s_{\#j}, P(s_{\#j}) > \in PM(s)$, and $P = P(t_{\#i}) * P(s_{\#j})$
2. and one of the following conditions holds
   (a) $t_{\#i} \equiv s_{\#j}$ and $LexRel = SYN$
   (b) $t_{\#i}$ hypernym of $s_{\#j}$ and $LexRel = BT$
   (c) $t_{\#i}$ meronym of $s_{\#j}$ and $LexRel = RT$

2.4 Experimental Evaluation

In [21], our normalization and annotation methods have been evaluated in order to measure and qualify their performance. They have been integrated within the MOMIS (Mediator EnvirOment for Multiple Information Sources) data integration system [5][6], and have been evaluated on two test cases; the first is a set of three ontologies from the benchmark OAEI 2008[7]; the second is composed of two relational schemata of the well-known Amalgam integration benchmark for bibliographic data[8]. Even if these data sources represent different scenarios w.r.t. Semantic Resource Framework, our previous evaluations can be used to give an idea about the quality of the results obtained by our methods.

To assess the quality of our method, gold standards were created for each normalization step as well as for the lexical annotation and the lexical relationship discovery methods. The gold standards were manually generated by a human expert. Then, we compared the gold standard with the result obtained by using our methods. For each experimental phase, we determined: the true positives, i.e. correct results (TP), as well as the false positives (FP) and the false negatives (FN).

Based on the cardinalities of the TP, FP, and FN sets, the following quality measures are computed:

- *Precision*= $\frac{|TP|}{|TP|+|FP|}$

[6] See `http://www.dbgroup.unimore.it` for references about the MOMIS project.
[7] 101, 205 209 ontologies available at `http://oaei.ontologymatching.org/2008/benchmarks/`
[8] See `http://dblab.cs.toronto.edu/~miller/amalgam/`.

	Precision	Recall	F-Measure
Lexical annotation without normalization	0.63	0.43	0.51
Lexical annotation with normalization	0.62	0.73	0.67
Discovered lexical relationships without normalization	0.49	0.29	0.36
Discovered lexical relationships with normalization	0.81	0.74	0.77

Table 1. Average performance of the lexical annotation and lexical relationship discovery methods with and without normalization.

- *Recall*= $\frac{|TP|}{|FN|+|TP|}$
- *F-Measure*= $2 * \frac{Precision*Recall}{Precision+Recall}$

Table 1 shows the average performance of lexical annotation and lexical relationship discovery with and without the normalization method. The experimental results show how the effectiveness of automatic lexical annotation and, as a consequence, the quality of the discovered lexical relationships are improved by the normalization method.

3 Towards Annotated Services in SRF

In this section, we describe an application of our normalization and annotation methods to the Semantic Resource Framework (SRF) described in a previous chapter of this book [6]. SRF is a multi-level (conceptual, logical, and physical level) description of data sources for searching computing applications. It extends the Service Mart model presented in [8] by making such model more expressive and more fitting to the web service description requirements. SRF represents a first step for adding more semantics to web service description.

By using our method, it is possible to enrich the semantics of SRF descriptions by annotating them w.r.t. the lexical resource WN.

Our normalization and annotation methods find application at the *conceptual level* of a Service Mart. The conceptual level includes the object's name and the collection of the object's attributes; all the attributes are typed: they can be atomic (single valued) or part of a repeating group (multi-valued). Moreover, the discovered lexical relationships may suggest useful information at the *logical level* to derive *connection patterns* between Service Marts.

We can easily apply our method by considering the classes and the attributes of a Service Mart [6], as the classes and attributes of a generic object schema as defined in Section 2.

Let us suppose we have a flight booking Service Mart having the following conceptual description:

Booking(CustomerName, BookingNR, FlightNumber, Airline, DepartureDatetime, DepartureAirport, ArrivalDatetime, ArrivalAirport) (1)

For the service attribute names that do not have an entry in WN, we apply our normalization method.

Attribute	Lex. Annotation	Prob.
Airline	$Airline_{\#2}$	0.89
ArrivalAirport	$\underline{Airport}_{\#1}$ FOR $Arrival_{\#2}$	0.9
DepartureAirport	$\underline{Airport}_{\#1}$ FOR $Departure_{\#1}$	0.9
BookingNumber	$Booking_{\#2}$ HAVE $\underline{Number}_{\#4}$	0.8
FlightNumber	$Flight_{\#9}$ HAVE $\underline{Number}_{\#4}$	0.62
	$Flight_{\#2}$ HAVE $\underline{Number}_{\#4}$	0.67

Table 2. Annotations of some attributes of the "Booking" Service Mart (for the CNs the word representing the head is underlined).

WN synset	WN gloss
$Airline_{\#2}$	*a commercial enterprise that provides scheduled flights for passenger*
$Airport_{\#1}$	*an airfield equipped with control tower and hangers as well as accommodations for passengers and cargo*
$Arrival_{\#1}$	*accomplishment of an objective*
$Arrival_{\#2}$	*the act of arriving at a certain place*
$Departure_{\#1}$	*act of departing*
$Booking_{\#2}$	*the act of reserving (a place or passage) or engaging the services of (a person or group)*
$Number_{\#4}$	*a numeral or string of numerals that is used for identification*
$Flight_{\#2}$	*an instance of traveling by air*
$Flight_{\#9}$	*a scheduled trip by plane between designated airports*

Table 3. WordNet glosses.

3.1 Normalization

The normalization method is divided in three steps: preprocessing, abbreviation expansion, and CN annotation. In the first step, the method recognizes as non-dictionary words the following labels:

CustomerName, BookingNR, FlightNumber, DepartureDatetime, DepartureAirport, ArrivalDateTime, ArrivalAirport

As a consequence, these labels are first tokenized (e.g. "CustomerName" as "Customer" and "Name") and then classified as CNs (i.e. "CustomerName, FlightNumber, DepartureDateTime, DepartureAirport, ArrivalDateTime, and ArrivalAirport") and as CNs containing abbreviations (i.e. "Booking NR").

The second step of normalization is focused on expanding abbreviations: the method automatically expands the previously identified CN containing abbreviations "BookingNR" as "BookingNumber".

The last step of the normalization deals with the CNs. During this step the CNs are interpreted. Let us consider the label "BookingNumber";it is composed by two con-

Attribute	Lex. Annotation	Prob.
Airport	$Airport_{\#1}$	1.0
FlightStatus.FLTNO	$Flight_{\#2}$ HAVE $Number_{\#4}$	0.89
FlightStatus.Airline	$Airline_{\#2}$	0.89
FlightStatus.ARRAirport	$Airport_{\#1}$ FOR $Arrival_{\#2}$	0.95

Table 4. The most relevant annotations of a subset of attributes of the "Flights" Service Mart.

stituents: "Booking" and "Number" which are automatically annotated by PWSD with, respectively, the synset $Booking_{\#2}$ with probability 0.89, and with $Number_{\#4}$ with probability 0.89, as shown in Tables 2 and 3. Then the *HAVE* semantic relationship is automatically selected and a new WN meaning for the CN is created and inserted in the WN noun hierarchy: we associate the new term "Booking Number" with a gloss given by union of the glosses of "Booking" and "Number" connected by the relationship *HAVE* (i.e. gloss of $Booking_{\#2}$ "HAVE" gloss of $Number_{\#4}$); moreover, we create a hypernym/hyponym relationship between the new synset for "Booking Number" and the synset of "Number", and a Related Term relationship between the new synset and the synset of the modifier "Booking". The probability value associated to the new synset will be the product of the probabilities of the individual annotations $Booking_{\#2}$ and $Number_{\#4}$, i.e. 0.8.

Note that, on this example, only the attribute "Airline" is a WN term, whereas the others are CNs not present.

3.2 Probabilistic Lexical Annotation

After normalization, we perform the probabilistic lexical annotation of all the labels except for the CNs that have been annotated by the normalization method.

For annotating the attribute "Airline", WSD1 selects $Airline_{\#2}$ with a probability of 0.65, WSD2 provides $Airline_{\#2}$ with a probability of 0.7 and WSD3 selects $Airline_{\#1}$ with a probability of 0.6. The Dempster-Shafer's rule of combination applied on these outputs returns the following annotations:

$$PM(Airline) = \{< Airline_{\#1}, 0.11 >, < Airline_{\#2}, 0.89 >\}.$$

By applying a threshold of 0.2 , the annotation $Airline_{\#1}$ is discarded. In the end, "Airline" is thus annotated by $Airline_{\#2}$ with a probability value of 0.89 (see Table 2).

3.3 Probabilistic Lexical Relationship Discovery

After lexical annotation, we can use the probability distributions over the set of possible meanings (i.e. the output of the PWSD) to infer probabilistic lexical relationships among the attributes of the two Service Marts that share the same application domains.

Let us assume, for example, another Service Mart about the scheduled flights departing from an airport, to explain the relationship discovery task:

"Booking"	**Lex. Rel.**	"Flights"	**Prob.**
Airline	SYN	*FlightStatus.AirLine*	0.79
FlightNumber	SYN	*FlightStatus.FLTNO*	0.60
ArrivalAirport	SYN	*FlightStatus.ARRAirport*	0.85
ArrivalAirport	NT	*Airport*	0.9
DepartureAirport	NT	*Airport*	0.9
BookingNR	RT	*FlightStatus.FLTNO*	0.71

Table 5. Lexical relationships between "Flights" and "Booking" Service Marts.

$$\textbf{Flights(Airport, FlightStatus(FLTNO, Airline, ARRAirport, ScheduledDPTDateTime, EstimatedDPTDateTime))} \quad (2)$$

The normalization and annotation methods applied on the conceptual description of "Flights" retrieve the annotations shown in Table 4. From the annotations of "Booking" and "Flights", we discover a set of lexical relationships between their attributes (as shown in Table 5).

Let us consider for example, the attribute "ARRAirport" (expanded to "ArrivalAirport") in "Flights"; it is split into its constituents, "Arrival" and "Airport". The constituents are disambiguated as $Arrival_{\#2}$ and $Airport_{\#1}$. Then, the semantic relationship "FOR" between the meanings of the head and the modifier is selected. When we compare the annotation of *Arrival Airport* in "Booking" and the annotation of *FlightStatus.ARRAirport* in "Flights", we discover a SYN relationship as the two elements share the same meaning ($Airport_{\#1}$ FOR $Arrival_{\#2}$.). On the other hand, when we examine the annotations of *Arrival Airport* in the "Booking" description and *Airport* in the "Flights" description, we discover an NT relationship as the new meaning of *Arrival Airport* is an NT of $Airport_{\#1}$.

At the logical level of Service Marts, lexical relationships may suggest useful information to build a *connection pattern* between them. Every connection pattern has a conceptual name and a logical specification, consisting of a sequence of simple comparison predicates between pairs of attributes or sub-attributes of the two services [8].

For example, the SYN relationships discovered between "Flights" and "Booking" (showed in Table 5) might be used to determine a connection between these two Service Marts. This connection could be exploited on previously defined access patterns where the "Airline", "FlightNumber", and "ArrivalAirport" attributes are defined as output parameters in the "Booking" access pattern, while "FlightStatus.AirLine", "FlightStatus.FLTNO", and "FlightStatus.ARRAirport" are input parameters in the "Flights" access pattern.

As a first example, let us suppose we want to define a simple connection between "Flights" and "Booking", checking just the existence of scheduled flights at the arrival airport of a booking: in this case, the lexical relationship *ArrivalAirport SYN FlightStatus.ARRAirport* (as shown on Table 5) helps in the identification of which attribute has to be connected to "ArrivalAirport".

As a consequence, the following connection pattern can be define:

ExistsArrivalAirport(Booking,Flights):[(ArrivalAirport=ARRAirport)]

This means that "Flights" and "Booking" are connected via the connection pattern "ExistsArrivalAirport", which uses a join on arrival airports. The interpretation of joins within connection patterns is existential: if the arrival airport in the "Booking" Service Mart is equal to the ARRAirport of any scheduled flights in the "Flights" description, the predicate is satisfied, and the two instances of "Booking" and "Flights" are composed to form an instance of the result.

Suppose now, we want to define another connection between "Flights" and "Booking" that controls the departure airport in addition to the arrival airport. In this case, selecting the attribute to be connected to "DepartureAirport" is less intuitive. The set of lexical relationships discovered comes to be an important help in this selection. In fact, as shown in Table 5, we found the relationships *DepartureAirport NT Airport* that has a hight probability value (i.e. 0.9). We can, thus, write the connection pattern:

ExistsLink(Booking,Flight):[(ArrivalAirport=ARRAirport) and (DepartureAirport = Airport)]

4 Related Work

Works related to the issues discussed in this chapter are in the area of schema matching, including probabilistic matching and WSD techniques.

The problem of linguistic normalization has received much attention in different areas such as machine translation, information extraction and information retrieval. As observed, the presence of non-dictionary words in schema element labels (including CNs and abbreviations) may affect the quality of *schema elements matching* and requires additional techniques to deal with [11]. Surprisingly, current schema integration systems either do not consider the problem of abbreviation expansion at all or solve it in a non-scalable way by including a *user-defined abbreviation dictionary* or by using only simple *string comparison techniques* [17, 2]. Dealing with short forms using a user-defined dictionary only suffers from the lack of scalability: (a) the dictionary cannot handle ad hoc abbreviations; (b) same abbreviations can have different expansions depending on the domain, thus an intervention of a schema/domain expert is still required; and (c) the dictionary evolves over time and it is necessary to maintain the table of abbreviations. Some works have tried to address the limitations of the user-defined dictionary approach by using simple string comparison techniques (e.g. the Similarity Flooding [18]). Syntactical methods are able to detect matches by comparing prefixes and suffixes of literals, however, they are not able to bring to the surface the semantics of abbreviations, thus, in contrast w.r.t. our method, they cannot detect a match between synonyms like "QTY" (short form of quantity) and "amount". Similarly to the abbreviation expansion problem, few papers address the problem of CN interpretation in schema matching area. The CN interpretation is manually executed or relies on a set of manually created rules in most of the work [12, 27]. Other schema and ontology matching tools do not interpret nor normalize CNs but they treat the constituents of a CN in isolation [15, 27, 25]. This oversimplification leads to the discovery of false positive relationships, thus worsens the matching results.

Several language-based methods have been experimented in the context of ontology matching and data integration (H-MATCH [9], CUPID [17]). Some methods rely on string-based techniques only. Other methods make use of external resources, such as dictionaries, to find similarities between terms, but in most of the cases, without performing any disambiguation on the terms. Unlike these methods, our approach is based first of all on the lexical annotation of ontology/schema elements. It is only after this phase that the similarity between elements is computed, thus overcoming the limitation of methods that cannot recognize the meaning of the elements. To the best of our knowledge, [3] is the first work that introduces WSD techniques in an integration process. One of its main limitations is that it does not make use of normalization techniques to process CNs, and this is reflected in a low coverage of the method. In the area of NLP, where WSD is a challenging topic, combination methods have been shown to be an effective way of improving WSD performance, in particular it has been showed that combination systems outperform the behavior of the individual algorithms [7, 22].

Modeling uncertainty in probabilistic schema matching has been an active area of research for some years [19]. Our method takes inspiration from [23], where the concept of probabilistic schema mapping is introduced and an algorithm for uncertain query answering is presented. The authors start from initial probabilistic schema mappings, and without dealing with the generation of probabilistic mappings, propose a probabilistic query answering method. The paper describes the requirements of a data integration system to support uncertainty: uncertain schema mappings, uncertain data and uncertain queries.

5 Conclusion

Lexical annotation (i.e. the explicit assignment of meanings to a schema elements w.r.t. a lexical reference) is an effective methodology in the discovery of lexical relationships between schema elements. Normalization helps to improve the performance of lexical annotation by increasing the number of annotable elements.

Starting from our previous works in the context of data integration, in this chapter, we presented how normalization and annotation methods work on generic object schema. Then, we shown how the methods might be applied in the context of search computing in order to enrich the semantics of SRF. We provided an application example showing the effectiveness of our methods: they can profitably be used in SRF to annotate the conceptual level of a service and to identify connection patterns between service descriptions belonging to the same application domain.

References

1. Y. An, A. Borgida, and J. Mylopoulos. Discovering the semantics of relational tables through mappings. 4244:1–32, 2006.
2. D. Aumueller, H. H. Do, S. Massmann, and E. Rahm. Schema and ontology matching with COMA++. In *SIGMOD'05*, pages 906–908, New York, NY, USA, 2005. ACM.
3. M. Banek, B. Vrdoljak, and A. M. Tjoa. Word sense disambiguation as the primary step of ontology integration. In S. S. Bhowmick, J. Küng, and R. Wagner, editors, *DEXA*, volume 5181 of *Lecture Notes in Computer Science*, pages 65–72. Springer, 2008.

4. R. Benassi, S. Bergamaschi, A. Fergnani, and D. Miselli. Extending a Lexicon Ontology for Intelligent Information Integration. In R. L. de Mántaras and L. Saitta, editors, *ECAI*, pages 278–282. IOS Press, 2004.
5. S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini. Semantic Integration of Heterogeneous Information Sources. *Data & Knowledge Engineering, Special Issue on Intelligent Information Integration*, 36(1):215–249, 2001.
6. M. Brambilla, A. Campi, S. Ceri, and S. Quarteroni. Semantic Resource Framework. In *New Trends on Search Computing. Stefano Ceri, Marco Brambilla (eds.). Springer LNCS, Vol. 6585, April 2011*.
7. S. Brody, R. Navigli, and M. Lapata. Ensemble Methods for Unsupervised WSD. In *ACL*. The Association for Computer Linguistics, 2006.
8. A. Campi, S. Ceri, A. Maesani, and S. Ronchi. Designing service marts for engineering search computing applications. In B. Benatallah, F. Casati, G. Kappel, and G. Rossi, editors, *ICWE*, volume 6189 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2010.
9. S. Castano, A. Ferrara, and S. Montanelli. Matching ontologies in open networked systems: Techniques and applications. *J. Data Semantics V*, pages 25–63, 2006.
10. W. W. Cohen, P. D. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In S. Kambhampati and C. A. Knoblock, editors, *IIWeb*, pages 73–78, 2003.
11. H. H. Do. *Schema Matching and Mapping-based Data Integration: Architecture, Approaches and Evaluation*. VDM Verlag, 2007.
12. D. W. Embley, D. Jackman, and L. Xu. Multifaceted Exploitation of Metadata for Attribute Match Discovery in Information Integration. In *Workshop on Information Integration on the Web*, pages 110–117, 2001.
13. G. A. M. et al. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244, 1990.
14. J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.
15. B. T. Le, R. Dieng-Kuntz, and F. Gandon. On ontology matching problems - for building a corporate semantic web in a multi-communities organization. In *ICEIS (4)*, pages 236–243, 2004.
16. J. N. Levi. *The Syntax and Semantics of Complex Nominals*. Academic Press, New York, 1978.
17. J. Madhavan, P. A. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. In *VLDB*, pages 49–58, 2001.
18. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In *ICDE*, pages 117–128, 2002.
19. M. Nagy, M. Vargas-Vera, and E. Motta. Dssim-ontology mapping with uncertainty. In P. Shvaiko, J. Euzenat, N. F. Noy, H. Stuckenschmidt, V. R. Benjamins, and M. Uschold, editors, *Ontology Matching*, volume 225 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.
20. S. Parsons and A. Hunter. A review of uncertainty handling formalisms. In A. Hunter and S. Parsons, editors, *Applications of Uncertainty Formalisms*, volume 1455 of *Lecture Notes in Computer Science*, pages 8–37. Springer, 1998.
21. L. Po and S. Sorrentino. Automatic generation of probabilistic relationships for improving schema matching. *Information Systems*, 36(2):192 – 208, 2011. Special Issue: Semantic Integration of Data, Multimedia, and Services.
22. J. Preiss. Probabilistic word sense disambiguation. *Computer Speech & Language*, 18(3):319–337, 2004.
23. A. D. Sarma, X. Dong, and A. Y. Halevy. Bootstrapping pay-as-you-go data integration systems. In J. T.-L. Wang, editor, *SIGMOD Conference*, pages 861–874. ACM, 2008.

24. G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.
25. P. Shvaiko, F. Giunchiglia, and M. Yatskevich. Semantic matching with s-match. *Semantic Web Information Management: a Model-Based Perspective*, XX:183–202, 2010.
26. S. Sorrentino, S. Bergamaschi, M. Gawinecki, and L. Po. Schema label normalization for improving schema matching. *Data & Knowledge Engineering*, 69(12):1254 – 1273, 2010. Special issue on 28th International Conference on Conceptual Modeling (ER 2009).
27. X. Su and J. A. Gulla. Semantic Enrichment for Ontology Mapping. In F. Meziane and E. Métais, editors, *NLDB*, volume 3136 of *Lecture Notes in Computer Science*, pages 217–228. Springer, 2004.