Introducing inference-driven OWL ABox enrichment

Alda Canito
School of Engineering
Polytechnic of Porto
Rua Dr. António Bernardino de Almeida, 431
4200-072 Porto
alrfc@isep.ipp.pt

Paulo Maio
School of Engineering
Polytechnic of Porto
Rua Dr. António Bernardino de Almeida, 431
4200-072 Porto
pam@isep.ipp.pt

Nuno Silva
School of Engineering
Polytechnic of Porto
Rua Dr. António Bernardino de Almeida, 431
4200-072 Porto
nps@isep.ipp.pt

ABSTRACT
Publically available text-based documents (e.g. news, meeting transcripts) are a very important source of knowledge for organizations and individuals. These documents refer domain entities such as persons, places, professional positions, decisions, actions, etc. Querying these documents (instead of browsing, searching and finding) is a very relevant task for any person in general, and particularly for professionals dealing with intensive knowledge tasks. Querying text-based documents' data, however, is not supported by common technology. For that, such documents' content has to be explicitly and formally captured into knowledge base facts. Making use of automatic NLP processes for capturing such facts is a common approach, but their relatively low precision and recall give rise to data quality problems. Further, facts existing in the documents are often insufficient to answer complex queries and, therefore, it is often necessary to enrich the captured facts with facts from third-party repositories (e.g. public LOD, private IS databases). This paper describes the adopted process to identify what data is currently missing from the knowledge base repository and which is desirable to collect from external repositories. The proposed process aims to foster and is driven by OWL DL inference-based instance (ABox) classification, which is supported by the constraints of the TBox.

Keywords
Knowledge Base, Ontology, TBox, ABox, Enrichment, Inference, OWL

1. INTRODUCTION
Publically available text-based documents (e.g. news, meeting transcripts) are a very important source of knowledge for organizations and individuals. Querying the content of these documents is not technologically supported, compelling the user to search, browse and integrate information by him/herself. This is a time-consuming, tedious, error-prone, unrepeatable and unconfident process.

iiWAS2013, 2-4 December, 2013, Vienna, Austria.

In the context of the World Search project [1], a system that is able to address semantically rich and complex queries over the content of unstructured or semi-structured documents has been created [2]. Presently, a repository (further also referred as knowledge base) meeting the Linked Open Data (LOD) principles [3] and the query building and execution applications are available and functional. However, the ability to populate the repository with facts from unstructured and semi-structured documents is still an open issue. Therefore, the population of important relationships between documents and ontological instances (e.g. persons, places, professional positions, decisions, actions) mainly relies in the user.

Initially, to address this open issue, a common approach in literature has been taken: making use of automatic NLP processes for capturing and explicitly and formally "semantizing" the documents' content. While the utility of the automatic NLP processes is evident, the relatively low precision and recall of these processes referred in [4], [5] was observed, which give rise to data quality problems, including duplicates, incoherencies, inconsistencies and incompleteness. Thus, to avoid (or at least to minimize) those data quality problems, the facts generated by the NLP process are conveniently analyzed and processed manually before being integrated into the repository. Additionally, it was perceived that the NLP' acquired data is often insufficient for the purpose of applying a reasoner [6] (or a classifier) to infer new facts from the already known data (ABox) together with the terminological component (TBox) of the repository.

Later, in order to (i) automatize the aforementioned user-based process and (ii) to address the incompleteness of the data, an iterative and incremental process called Ontology-driven Data Cleansing and Enrichment (ODCE) was devised [7]. The ODCE process is triggered by the NLP process generated data and driven by the semantically rich OWL DL [8] TBox (created by domain experts) underlying the repository. It combines tasks such as (a) data cleaning, (b) ontology population and (c) enrichment for inference purposes. A brief description of the overall process is provided in section 2.

This paper focuses on the enrichment task of the ODCE process. The enrichment task seeks on third-parties repositories the missing data that will promote inference (namely classification of instances). Our approach suggests adopting the OWL DL ontological constraints (e.g. equivalent class, intersection, union, complement, disjoint) as driving vectors of the enrichment

process. I.e. the process will seek for the data that promotes the inference of specific ontological constraints. However, different ontological constraints require different enrichment processes. There are two kinds of enrichment processes: simple and composed. A simple process respects a simple and unique ontological constraint (e.g. some/all values from, has value, has self) for deciding what data is necessary and how to provide it. Composed processes reflect the OWL composed constructs of intersection, union and complement. Again, the characteristics of these ontological constructs motivate different decisions and enrichment processes. In particular, they rely on the simple enrichment processes and in other composed processes, thus giving rise to a recursive enrichment process.

Different enrichment processes have different demands in terms of data resources, reasoning needs, time, etc. Instead of recursively populating all the missing data, it is suggested enriching the repository in a controlled way by considering the efforts and resources necessary to provide it. The proposed approach suggests defining enrichment decision strategies to acquire the data, considering the following dimensions: (i) the available and required data, (ii) the characteristics of the ontological constraints and (iii) the characteristics of the data sources.

The remaining of the paper is organized as follows. Section 2 presents the overall ODCE process. Section 3 describes in detail the repository enrichment process and proposes a software development framework. Section 4 presents a real-world example making use of the proposed software development framework. Our proposal is compared to other works in section 5. Finally, section 6 summarizes the contributions and points out next research steps.

2. THE ODCE PROCESS

This section briefly describes the Ontology-driven Data Cleansing and Enrichment (ODCE) process, on which the proposed enrichment strategies are applied.

The ODCE process consists in eight steps (or tasks) combined in a iterative and incremental manner, as depicted in Figure 1.This is triggered through a NLP process (not represented in Figure 1) whose output is a knowledge base (KB_{NLP}). In this context, a knowledge base (KB) is a tuple in the form of $KB: (T, A)$, where T (TBox) is the terminological/intensional information, and A (ABox) is the assertional/concrete situation/extensional information. While this is a common definition of a Description Logics (DL) knowledge base, the distinction between TBox and ABox is not always possible or evident. The KB presented in section 3.1 contains an example of such difficulty: the individual $blue$ (ABox) is used in defining the class $BlueCar$ (TBox).

The purpose of the ODCE process is to integrate and enrich $KB_{NLP} = (T_{NLP}, A_{NLP})$ in a proper, consistent and automatic way, into the $KB_{END} = (T_{END}, A_{END})$. T_{END} covers the same domain knowledge of T_{NLP} but it is semantically enhanced by domain experts by including necessary and sufficient conditions in addition to the necessary conditions already present in T_{NLP}.

The first step (NLP Cleansing) ensures that KB_{NLP} is consistent and ready for integration. For that, it applies a set of data cleansing operations to detect and to correct inaccurate information, namely to avoid duplicated facts and/or entities. The result of this step is a minimal, clean and consistent set of facts (KB_{ADD}) to be integrated into the KB_{END} knowledge base.

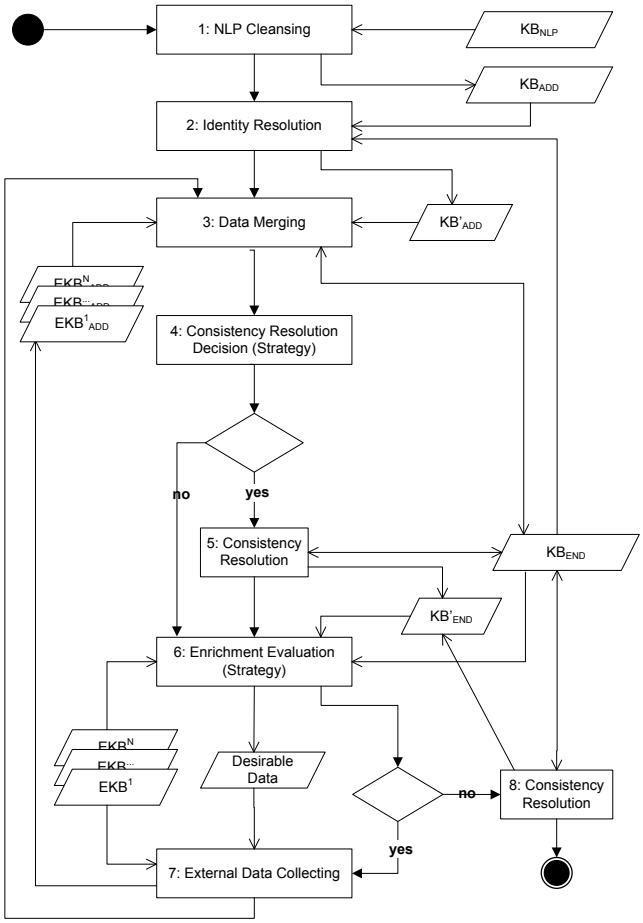

Figure 1. The ODCE process.

The second step (Identity Resolution) [9] consists in identifying univocally the entities mentioned in KB_{ADD} according to the ones existing in KB_{END}. Thus, this task is accomplished by verifying that for each entity $e \in KB_{ADD}$, an entity $e' \in KB_{END}$ exists such that e and e' are considered by a given identity function as referring to the same real/domain entity. In such cases, all references to e into KB_{ADD} are replaced by e' giving raise to KB'_{ADD}.

The third step (Data Merging) consists in taking several source knowledge bases (namely KB'_{ADD}) containing previously collected and prepared information with the unique purpose of being integrated (or merged) into the target knowledge base (KB_{END}). Therefore, it entails that a transparent data transformation process occurs between each (possible) pair of source and target knowledge bases. In this particular, a (declarative) alignment between the ontologies describing the source and target knowledge bases is required. The result of the data-merging task may leave the KB_{END} knowledge base temporarily inconsistent. It is responsibility of the next steps to resolve such inconsistencies.

The fourth step (Consistency Resolution Decision) checks if the KB_{END} knowledge base has inconsistencies resulting from the execution of the previous steps. Considering a knowledge base in OWL DL, inconsistencies can be analyzed and identified by means of a reasoner such as Pellet [6].

Based on (i) the (in) existence of inconsistencies and (ii) on the kind of inconsistencies found, it decides either:

- To resolve the inconsistencies found and, therefore, the ODCE process flows to step five before executing the enrichment task;
- To proceed immediately to the enrichment task (step 6) since the inconsistencies found are not considered as causing undesirable effects (e.g. malfunctioning, incompleteness) on such task.

Furthermore, this decision takes into consideration other issues such as the performance of the overall process, the requirements of the enrichment process, the interdependencies between the ODCE process and the running application (e.g. answering complex questions) that rely on the knowledge base.

Yet, it is important noticing that, independently of the decision made at this point, the ODCE process ensures that at the end the knowledge base is consistent. This is achieved because the last step of the process corresponds to a mandatory consistency resolution task.

The sixth step (Enrichment Evaluation) evaluates the need to enrich the KB_{END} knowledge base with new (missing) information that would foster the further (re) classification of the instances (ABox). These new facts are to be collected from available external knowledge bases (e.g. DBPedia) and added to the KB_{END}. As proposed further in section 3, the driving vector of this task is the terminological component (TBox) of KB_{END} and the underlying semantics of the OWL DL constructs (e.g. equivalent class, intersection, union, and complement).

Consequently, this step identifies the desirable information that is lacking in KB_{END}. It is worth to bear in mind that the desirable information may resolve some inconsistencies generated by the third step but may also raise other inconsistencies. Additionally, this step also instructs the ODCE process either:

- To proceed to the External Data Collecting step if it has been verified the need to enrich/collect information from the external knowledge bases;
- To proceed to the last step (Consistency Resolution) of the ODCE process.

The seventh step (external Data Collecting) extracts/collects from a set of external knowledge bases the information specified previously as desirable. The result is a set of knowledge bases (one for each external knowledge base used) containing the information to be further integrated in the KB_{END}.

At the end of this step, the process proceeds again to the Data Merging step in order to merge the collected data into the KB_{END}. After the collected data is merged new enrichment requirements may appear. As that, a new iteration of the process starts. The process has as much iterations as necessary as decided in the sixth step (Enrichment Evaluation).

At the end of the process and in order to ensure that the KB_{END} is consistent, the last step is the mandatory Consistency Resolution task. This task consists in identifying and resolving the inconsistencies caused by the added (or modified) facts on the KB_{END} knowledge base by the Data Merging task.

Steps regarding the enrichment tasks (six and seven respectively) are the most relevant and innovative ones in our proposal and the focus of this paper. Thus, they will be described in depth in the following section. The interested reader can find detailed description of the ODCE process in [7].

3. THE ENRICHMENT PROCESS

The enrichment process has two sub-tasks: (i) deciding what are the worth-to-collect missing facts and (ii) collecting the missing facts. The enrichment process is driven by the Description Logics inference process upon the KB. I.e. the goal is to enrich the KB in a way that promotes inference of new facts. The inference is performed by a description logic reasoner (or classifier) through the terminological level and the assertional level.

Assuming the knowledge bases are terminologically described (TBox) by means of OWL DL [8] ontologies, it is important to determine what and how the OWL constructs influence the inference process. Those constructs are identified and a restricted set of expressions is defined according to our needs in section 3.1. Next, in section 3.2, those expressions are exploited to systematize the actions to collect the missing facts. Section 3.3 discusses the dimensions that are considered in deciding whether to execute or not the collecting of the missing/desirable facts. Section 3.4 describes the software development framework designed accordingly.

3.1 Expressions Promoting Inference

There are two kinds of possible ABox inferences relevant for this process:

- type inference (instance checking) is the process that checks if an assertion $C(a)$ is true for every model $\mathcal{I}$ of an ABox A and a TBox T. I.e. determines if an individual a is of type C in every model $\mathcal{I}$;
- relationship inference (relationship checking) is the process that checks if an assertion $R(a,b)$ is true for any model $\mathcal{I}$ of a ABox A and a TBox T. I.e. determines if an individual a is R-related with individual b in every model $\mathcal{I}$.

It worth noticing that the relationship checking in OWL DL occurs in a restrictive context of TBox constraints and individuals. For example, considering the following knowledge base (in DL syntax):

$$Car \sqsubseteq \exists hasColour.Colour$$

$$BlueCar \equiv Car \sqcap hasColour{:}\, blue$$

$$Colour \sqsubseteq Thing$$

$$blue{:}\, Colour$$

$$car1{:}\, BlueCar$$

a DL reasoner will infer the following relationship:

$$hasColour(car1, blue)$$

The relationship inference is possible because of the type of the individual ($car1$), which is either set statically (as in the above example) or determined by the inference. Thus, as the relationship inference depends on the type inference, the remaining of the paper will focus the efforts on the type inference only.

The OWL construct that allows type inference is the EquivalentClasses, which is represented in Backus Normal Form (BNF) notation as:

EquivalentClasses :=
'EquivalentClasses' '(' ClassExpression ClassExpression { ClassExpression } ')'

This would be represented in DL syntax as:

$$ClassExpr1 \equiv ClassExpr2 \equiv \cdots \equiv ClassExprN$$

EquivalentClasses may thus be seen as a combination of two or more ClassExpression, which in turn is defined as follows:

ClassExpression :=
Class | ObjectIntersectionOf | ObjectUnionOf | ObjectComplementOf | ObjectOneOf | ObjectSomeValuesFrom | ObjectAllValuesFrom | ObjectHasValue | ObjectHasSelf | ObjectMinCardinality | ObjectMaxCardinality | ObjectExactCardinality | DataSomeValuesFrom | DataAllValuesFrom | DataHasValue | DataMinCardinality | DataMaxCardinality | DataExactCardinality

Description Logics makes two important assumptions:

- Not Unique Name Assumption (not-UNA), states that different names does not imply distinct entities. This is the opposite of UNA (typical assumption in database applications) in the sense that if two entities have different names then they are considered distinct entities. Weak UNA is a new approach that means that if two individuals are not inferred to be the same, then they will be assumed to be distinct [10]. This UNA variant is not yet supported by the DL reasoners, thus preventing its adoption in this work. Because the goal is to collect individuals from different third-party KB, we will force the adoption of UNA at the ABox level by explicitly stating "owl:differentFrom" relationships between individuals unless the identity resolution step states otherwise;
- Open World Assumption (OWA), states that something is false only if it explicitly stated. This is the opposite of Closed World Assumption (CWA) in the sense that if something is not known it is considered false. Unlike UNA, DL reasoners follow invariably OWA and there is no workaround. Consequently, we will follow OWA.

Considering the open world assumption (OWA), not all class expression combinations promote type inference. For example, while the class expression ObjectAllValuesFrom is significant for subsumption classification and consistency checking, it does not facilitate type inference. I.e. it is not possible to conclude that an individual has a certain relationship only with individuals of a specific class, as assuming open world, unknown relationships with individual of other classes may exist that deny that constraint.

All the ClassExpression combinations promoting type inference were analyzed and identified. These combinations are a sub-set of all possible ClassExpression combinations and are referred to as ClassExpressionTypeA (CETA). The meaningful interpretation of EquivalentClasses for type inference is referred to EquivalentClasses4Inference and is defined as follows:

EquivalentClasses4Inference :=
'EquivalentClasses' '(' Class CETA { CETA } ')'

CETA := CEAnd | CEOr | CEPositive | CEComplement

CEAnd := 'ObjectIntersectionOf' '(' CETA CETA { CETA } ')'

CEOr := 'ObjectUnionOf' '(' CETA CETA { CETA } ')'

Positive class expressions (CEPositive) are the class expressions which promote inference when the facts are true. I.e. it is possible to find facts that prove the class expressions (axioms):

- An instance i has **some values** of a type C related through property p if at least one $p(i,i'): i' \in C$ is found;
- An instance i **has a value** v related through property p if $p(i,v)$ is found;
- An instance i is related through property p to itself (**has self**) if $p(i,i)$ is found;
- An instance i has a **minimum** n relationships p if $|\{p(i,i')\}| \geq n$ are found;
- An instance i **is of type** C, if $C(i)$ is found. Notice that if $C(i)$ and $D(i)$ are found and C disjoint D, a consistency issue exists.

Hence, positive class expressions are defined as follows:

CEPositive :=
Class | ObjectSomeValuesFrom | ObjectHasValue | ObjectHasSelf | ObjectOneOf | ObjectMinCardinality | DataSomeValuesFrom | DataMinCardinality

On the other hand, the complement class expression has a different content, and it is composed by a single class expression of a different type, referred as ClassExpressionTypeB (CETB).

CEComplement := 'ObjectComplementOf' '(' CETB ')'

CETB := CEAnd | CEOr | CEComplement | CENegative

In this context, the negative class expressions (CENegative) are useful because they allow complement-based inference by determining false facts, namelly:

- An instance i does not have **all values** of a property p of type C if at least one $p(i,i'): i' \in D$ is found and C and D are disjoint;
- An instance i does not have **exactly n** relationships p if at least $|\{p(i,i')\}| > n$ are found;
- An instance i does not have at **maximum n** relationships p if at least $|\{p(i,i')\}| > n$ are found;
- An instance i is **not of type** C if it is found D(i) such that C is disjoint of D. Notice that if $C(i)$ and $D(i)$ are found and C is disjoint of D, a consistency issue exists.

Negative class expressions are therefore defined as follows:

CENegative :=
Class | ObjectMaxCardinality | ObjectAllValuesFrom | ObjectExactCardinality | DataMaxCardinality | DataAllValuesFrom | DataExactCardinality

Yet, the constructs mentioned in the above CEPositive and CENegative definitions are also analyzed and their interpretations are constrained to identify the meaningful inference class expressions of each one. As a result, each construct may only "accept" CETA instead of ClassExpression. For example, the construct ObjectSomeValuesFrom that is originally defined as

ObjectSomeValuesFrom := 'ObjectSomeValuesFrom' '('
ObjectPropertyExpression ClassExpression ')'

is interpreted as its definition comprehending CETA instead of ClassExpression, such that:

ObjectSomeValuesFrom := 'ObjectSomeValuesFrom' '('
ObjectPropertyExpression CETA ')'

For brevity reasons, the remainder (re) interpretations of constructs mentioned in CEPositive and CENegative are omitted.

Figure 2 depicts an UML class diagram representing the BNF expressions capturing the class expressions useful for inference (Object and Datatype restrictions are not distinguished).

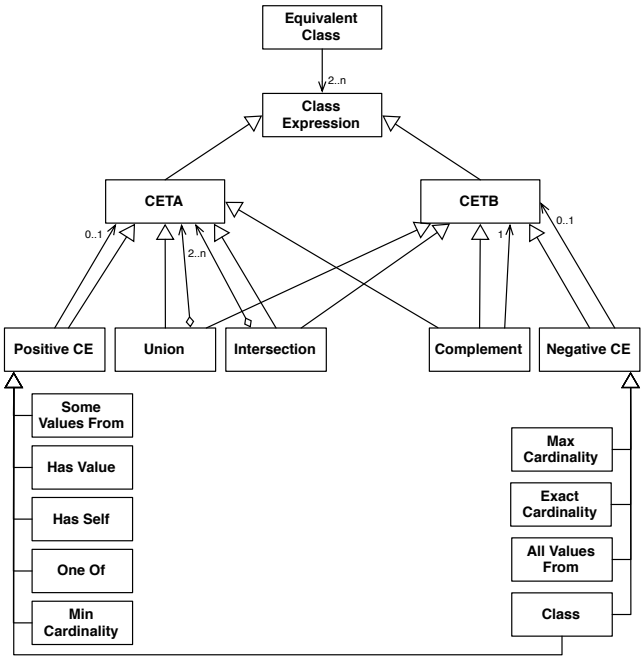

Figure 2. UML class diagram of equivalent class expression for inference.

3.2 Collecting Desirable Data

Considering that the goal is to enrich every instance (say i) added to the KB with facts that allow/promote inference of new facts, each class expression drives the efforts for searching/collecting the necessary facts related to that expression. Here, it is important to distinguish between:

- Composed class expressions, which are those related to the OWL Propositional Connectives (i.e. Intersection, Union and Complement), which in their canonical form have two or more composed class expression or simple class expression;
- Simple class expressions, related to CETA and CETB, are those that in their canonical form have a non-composed class expression, i.e. an indivisible class expression.

Accordingly, the collecting task of each single class expression depends on which construct it relies on, as systematized in Table 1 (again, Object and Datatype restrictions are not distinguished).

Table 1. OWL constructs and respective collecting actions

OWL class expressions	Collecting action
$Class$	$collect(i, Class) \rightarrow \{Facts, bSucc\}$
	$collect(i, Class, DisjointClass)$
$SomeValuesFrom(p\ C)$	$collect(i, p, C)$
$AllValuesFrom(p\ C)$	$collect(i, p, C, DisjointClass)$
$HasValue(p\ j)$	$collect(i, p, j)$
$HasSelf(p)$	$collect(i, p, i)$
$OneOf(p\ enum)$	$collectOne(i, p, enum)$
$MinCardinality(p\ y)$	$collectAtLeast(i, p, y)$
$MaxCardinality(p\ y)$	$collectAtLeast(i, p, y+1)$
$ExactCardinality(p\ y)$	$collectAtLeast(i, p, y+1)$

Each enrichment action returns a (possibly empty) set of facts ($Facts$) and a boolean value ($bSucc$) indicating the success or failure of the data search process (as represented in the first row of the table, but omitted in the subsequent rows):

- $collect(i, Class) \rightarrow \{Facts, bSucc\}$ returns a fact in the form $Class(i)$ proving that instance i is of type $Class$. Further, it will return *true* if the facts are enough to prove so or *false* otherwise;
- $collect(i, Class, DisjointClass) \rightarrow \{Facts, bSucc\}$ returns facts $C'(i)$ such that $C' \in DisjointClass$, proving that i is not of type $Class$;
- $collect(i, p, C) \rightarrow \{Facts, bSucc\}$ returns the facts in the form $p(i, j)$ such that $j \in C$, proving that instance i has p-relationships with entities of type C;
- $collect(i, p, C, DisjointClass) \rightarrow \{Facts, bSucc\}$ returns the facts in the form $p(i, j)$ such that $j \in C'$ and $C' \in DisjointClass$ proving that i has p-relationships with instances that are not of type C ;
- $collect(i, p, j) \rightarrow \{Facts, bSucc\}$ returns the facts in the form $p(i, j)$, proving that proving that i is p-related with j, whose type is irrelevant;
- $collect(i, p, i) \rightarrow \{Facts, bSucc\}$ returns the fact $p(i, i)$, proving that i is p-related to itself;
- $collectOne(i, p, set) \rightarrow \{Facts, bSucc\}$ returns the fact $p(i, j)$, proving that i is p-related to $j \in set$;
- $collectAtLeast(i, p, y) \rightarrow \{Facts, bSucc\}$ returns at least y facts in the form $p(i, j)$, proving that i is p-relate to at least y other instances;
- $collectAtLeast(i, p, y+1) \rightarrow \{Facts, bSucc\}$ returns at least $y+1$ facts in the form $p(i, j)$, proving that i is p-relate to at least $y+1$ other instances.

On the other hand, the collecting task of composed class expressions results from the "combination" of the results of two or more class expressions. For example, the equivalent expression:

$$BlueCar \equiv Car \sqcap hasColour{:}\ blue$$

is composed by a CEAnd, which in turn is composed by a Class and HasValue constructs, which will require a $collect(i, Car)$ and $collect(i, hasColour, blue)$ collecting actions respectively. The CEAnd collecting action will return true if the collecting action of all the included constructs return true, and false otherwise. Table 2 systematizes the collecting actions for the OWL propositional connectives. This collecting task is further referred to as Composed Class Expression Collecting.

Table 2. OWL Propositional Connectives and respective collecting actions

OWL Propositional Connectives	Collecting action
$IntersectionOf(C_1 \dots C_n)$	$and\big(collect(C_1) \dots collect(C_n)\big)$
$UnionOf(C_1 \dots C_n)$	$or\big(collect(C_1) \dots collect(C_n)\big)$
$ComplementOf(C)$	$not\big(collect(C)\big)$

In particular, the $not\big(collect(C)\big)$ action will return true if the $collect(C)$ action returns false. Of course that only the collecting actions related to CETB are admissible in this context: Intersection, Union, Complement, All Values From, Exact Cardinality, Max Cardinality and Class (Disjoint).

3.3 Enrichment Decision Strategies

The effort for carrying each of these actions is different depending on several dimensions and, in particular, the following ones:

- Internal criteria: the type and characteristics of the OWL constructs;
- Data sources criteria: the characteristics and number of the data sources involved;
- Context criteria: the relative importance given to a specific inferred fact in a specific domain/application.

The goal is to analyze and systematize these dimensions into a strategy-based decision framework for automatically adoption during enrichment.

3.3.1 Internal criteria

This type of criteria varies according to the previous distinction between single class expressions and composed ones. With respect to the simple class expressions one may consider, for example, (i) the cost of the collecting actions, (ii) the number of disjoint classes, (iii) the deepness of disjoint classes. Concerning the composed class expressions one may consider (i) the tree deepness (e.g. is less than y), (ii) the total number of class expressions, (iii) the minimum number of class expressions that will run the collecting action.

It is worth to notice that the enrichment of composed class expressions depends not only on the characteristics of itself, but also on the characteristics of its included class expressions. For example, the intersection class expression obliges that all the included class expressions are executed, while the union class expression requires only one to be successfully executed. The complement class expression is composed by a simple class expression but its characteristics and dimensions are very similar to those of intersection and union.

3.3.2 Data sources criteria

Regarding the second dimension, this type of criteria may take into consideration (i) the number of data sources required (e.g. more/less than y), (ii) the number of data sources available (e.g. more/less than x), (iii) the accessibility of the data source (e.g. public and/or/xor private), (iv) the precision and/or recall of each data source, (v) the cost of access/reading the data source (e.g. traffic).

3.3.3 Context criteria

Regarding the third dimension, this type of criteria may consider the pragmatics (business pertinence) of classifying an instance as being of a given type or, in turn, simply opt to not run the collecting action for (i) a concrete entity (e.g. Student) or (ii) a concrete kind of class expression (e.g. intersection).

3.4 Software Development Framework

To facilitate and drive the development of the enrichment process, the team designed a software development framework that captures the concepts and the approach described above. For that, the team have resorted to the well-known software design pattern Strategy [11]. The resulting design is captured in the UML class diagram depicted in Figure 3 and described next.

Every ontological construct (e.g. Some Values From) has an enrichment interface counterpart (e.g. Some Values From Enrichment) derived from the CE Enrichment interface. This interface defines several methods for accessing its configuration (i.e. gets) and two important methods:

- willRun(), that decides whether the enrichment process will run or not. This method relies on a specific enrichment decision strategy (CE Enrich Decision), that in turn makes use of the defined criteria parameters;
- run(), which executes the collecting action that eventually leads to new facts.

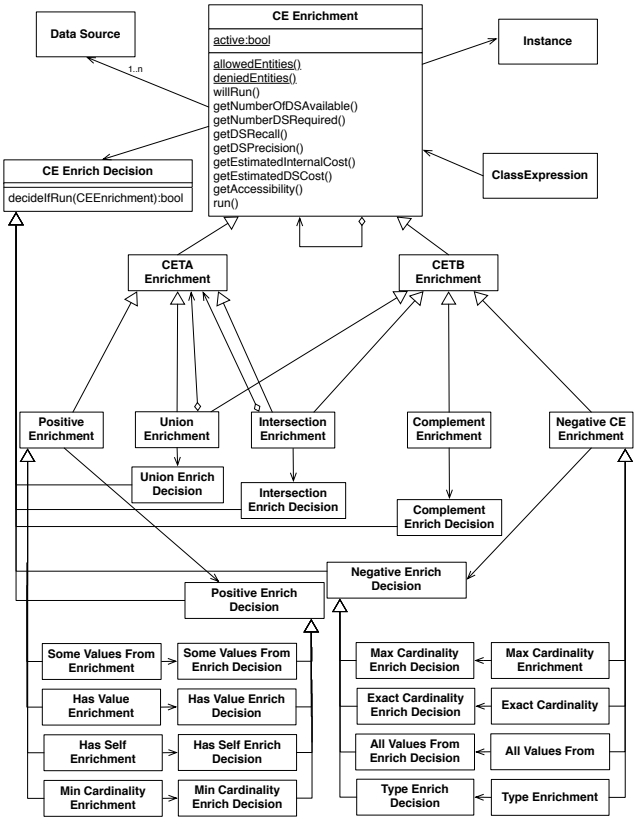

Figure 3. Class diagram depicting the enrichment actions and enrichment decision strategies.

The CE Enrich Decision strategy has a method that takes the enrichment process and, based on the available data (ABox), terminological constraints (TBox), and criteria parameters, decides whether to execute or not execute the enrichment attempt. This design has been successfully applied in the World Search system and in the ODCE process in particular.

4. A WALK-THROUGH EXAMPLE

To facilitate the understanding of the enrichment process and of the composed nature of the strategies, a short, yet real-world scenario from the World Search project is presented. This scenario considers the need to populate KB_{END} from Portuguese news and municipal meetings transcripts.

Consider a knowledge base described by the TBox defined partially through the following axioms in DL syntax and graphically depicted in Figure 4.

1. $Person \sqsubseteq \exists hasPosition.Position$
2. $PoliticalOffice \sqsubseteq Position$
3. $MinisterOffice \sqsubseteq PoliticalOffice$
4. $PrimeMinisterOffice \sqsubseteq MinisterOffice$
5. $EuroLocation \sqsubseteq Location$

6. $Party \sqsubseteq \exists hasMember.Person$
7. $isMemberOf \equiv hasMember^{-}$
8. $Party \sqsubseteq \exists isLeading.Government$
9. $\exists rules.\top \sqsubseteq Government$
10. $\top \sqsubseteq \forall rules.Location$

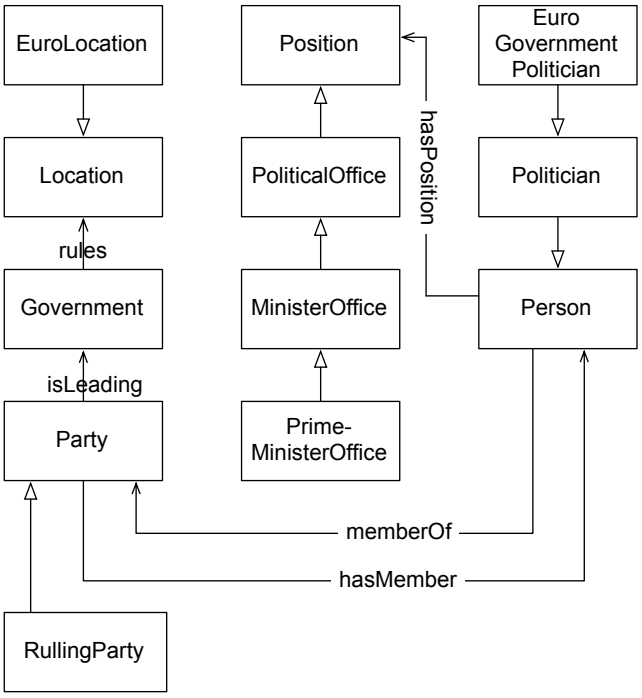

Figure 4. TBox's classes and their relations (partial)

Further, consider the following definitions of equivalent classes:

11. $RullingParty \equiv Party \sqcap \exists isLeading.Government$
12. $RullingParty \equiv Party \sqcap \exists hasMember.(\exists hasPosition.PrimeMinisterOffice)$
13. $Politician \equiv Person \sqcap \exists hasPosition.PoliticalOffice$
14. $EuroGovernmentPolitician \equiv Politician \sqcap \exists isMemberOf.(\exists rules.EuroLocation)$

Now, consider that from the earlier tasks of the ODCE process (i.e. tasks 1 to 5) the following ABox assertions were integrated in the knowledge base:

15. $Party(ps)$
16. $Person(jose\_socrates)$

Based on this knowledge base, the enrichment evaluation task of the ODCE process focus on the (newly) integrated instances to identify which is the desirable data. As that:

- The instance $Party(ps)$ serves as motivation for acquiring the facts needed by the equivalent class expression associated to the $RullingParty$ class, as it partially fills its constraints;
- The instance $Person(jose\_socrates)$ serves as motivation for acquiring the facts needed by the equivalent class expression associated to the $Politician$ class, as it partially fills its constraints. Further, because the equivalent class expression associated to $EuroGovernmentPolitician$ class depends on $Politician$ class, it is evaluated for decision too.

These constraints and the tree of constraints for each equivalent class are depicted in Figure 5. Also, consider the annotations in Figure 5 as the enrichment decision criteria of each class expression. For example, the intersection class expression enrichment labeled as "intersection enrichment 1" is constrained (i) to a maximum of one level deep (i.e. maxdeep:1) and (ii) to a maximum of two class expressions (i.e. max 2 CE).

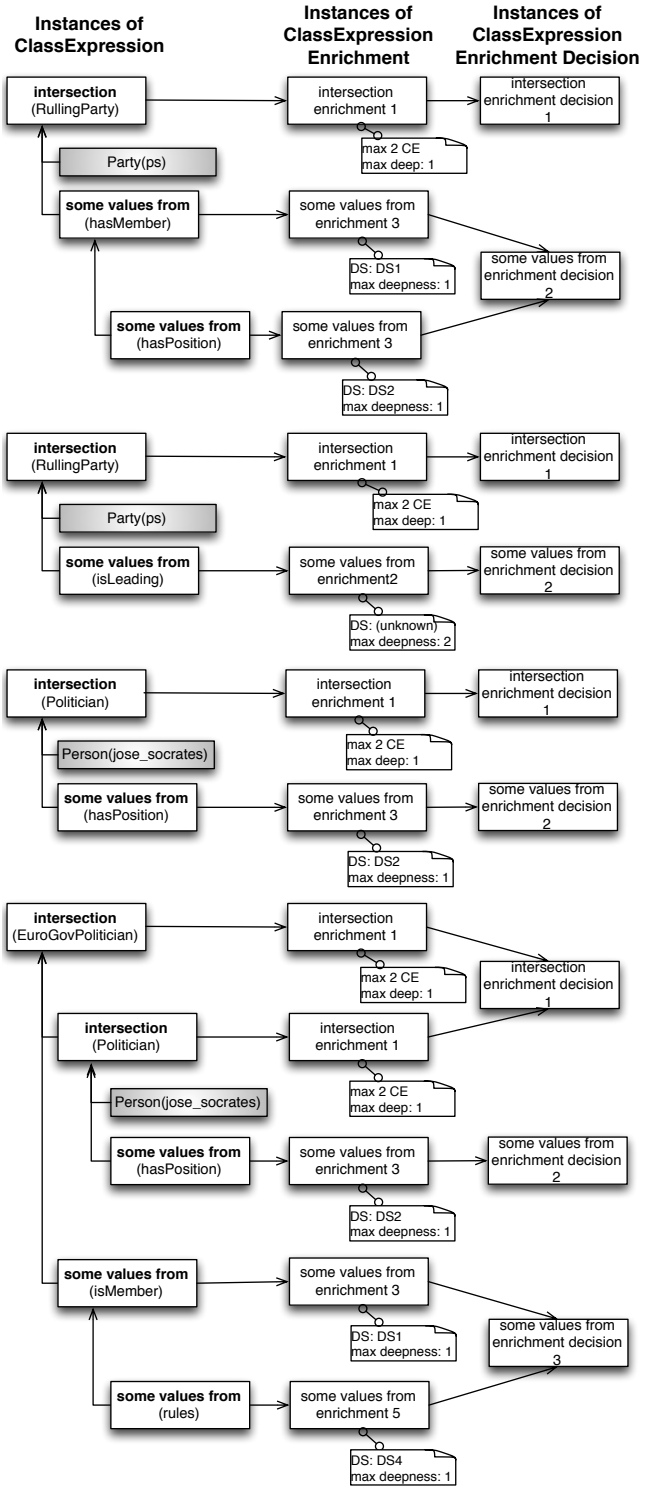

Figure 5. SDF's perspective of the example's enrichment data

Due to these constraints and the KB's content, only the $Politician$ equivalent class enrichment process will be executed. Notice that the other enrichment processes are not executed because (i) the resolution tree exceeds the specified maximum deep constraint value or (ii) it is known *a priori* that no external data sources can provide the intended information (e.g. *isLeading* relationships) since no alignments exist for that relation (between the World Search knowledge base and the external data sources).

Execution of the $Politician$ equivalent class enrichment process will tentatively enrich the instance $jose\_socrates$ with the facts that permit the inference of the equivalent class. This corresponds to execute:

$$collect(jose\_socrates, hasPosition, PoliticalOffice)$$

Consider now that the querying, cleansing and integration tasks of the ODCE process (i.e. tasks 7, 3 and 4 respectively) updates the knowledge base with the following new assertions:

17. $hasPosition(jose\_socrates, minister122)$
18. $hasPosition(jose\_socrates, prime\_minister232)$
19. $MinisterOffice(minister122)$
20. $PrimeMinisterOffice(prime\_minister232)$

As a result, a reasoner will be able to infer the axiom:

21. $Politician(jose\_socrates)$

Once this assertion is added to the KB, new enrichment attempts are made for every equivalent class definition (not depicted in Figure 5). Considering the current content of the KB, both the $RullingParty$ and the $EuroGovernmentPolitician$ equivalent classes are re-evaluated ($Politician$ has no further data that motivates its enrichment).

The re-evaluation of the $RullingParty$ strategy considers that the KB contains one instance of $Person$ that holds the position of $PrimeMinisterOffice$, which reduces the enrichment requirement of the $hasMember$ expression to one level (instead of the previous two levels), and thus, since no constraints are now violated, it decides to executed the enrichment process.

For this case, the enrichment process will collect information about $jose\_socrates$ affiliation to the 'ps' party.

$$collect(jose\_socrates, isMemberOf, ps)$$

Considering that this search effort is well succeeded, a new axiom is added to the KB:

22. $isMemberOf(jose\_socrates, ps)$

which in turn triggers the inference of:

23. $hasMember(ps, jose\_socrates)$
24. $RullingParty(ps)$

The same *rationale* is valid and applicable for the $EuroGovernmentPolitician$ class.

As demonstrated by the example, the proposed enrichment process is iterative and incremental, relying (i) on the intensional information (ABox), (ii) the terminological constraints (TBox), (iii) the enrichment criteria and (iv) the decision making specified by means of strategies.

5. RELATED WORK

Ontology enrichment is commonly used to describe the process of parsing documents to extract schemas and facts. Some approaches, like the system described in [12] aim to extract and expand an ontology using several sources of structured and unstructured data (e.g. web documents, databases) using NLP. This system addresses the problem of heterogeneity of data and attempts to solve it by providing a framework, extracting ontologies from a set of data. Much like our proposal, querying other data sources further enhances the enrichment process. However, while it is TBox-driven, unlike our work, it is focused on expanding the number of addressed concepts.

While the work described is similar to ABox abduction [13], they differ in the sense that ABox abduction tries to explain already existing facts, while the enrichment process tries to go from already existing facts to new (inferred) facts.

The system described in [14] extracts instances from various sources through a domain-independent TBox-driven approach. Like our system, the TBox drives the population process, determining which data to extract. However, unlike our system, it makes no use of the existing instances and, therefore, it does not attempt to fill potential gaps in the existing data.

In [15] the authors present a system whose goal is to mix traditional information retrieval queries and ontology-based queries. For that, documents are semantically annotated based on a very lightweight TBox through an NLP-oriented process. In this respect, this system corresponds to the NLP-based parsing step of our approach (that triggers the proposed ODCE process). Moreover, while our goal is to enrich the ABox in order to reason and infer new data enabling complex query answering, the aforementioned system intends to gather data as much and broader as possible to provide extended information.

The work presented in [16] evaluates the types of change that can occur at the terminological level of the KB. Depending on the nature of such change the system chooses an appropriate strategy predicting other changes that may be necessary to maintain consistency. In our proposal, a similar strategy-based approach is used to drive the enrichment process of the ABox instead of the evolution of the TBox.

6. CONCLUSION AND FUTURE WORK

This paper presented a KB enrichment process characterized by:

- NLP-triggered, in the sense that the data generated by the automatic NLP process will trigger and will be the motivation of the enrichment process;
- OWL-based, because the enrichment process is controlled by the ontological constraints and the ontology constructs semantics of OWL DL, including the OWA;
- Inference-driven, because the enrichment process aims to support and promote inference of new facts.

This enrichment approach is contextualized first in the Ontology-based Data Cleansing and Enrichment process, which in turn is part of the World Search project system aiming to support user performing complex queries upon LOD repositories. The designed software development framework has been successfully applied in this technological context.

The proposed enrichment process grounds on and follows the semantics of the OWL DL language thus guarantying a wide understanding and acceptance semantics. Nevertheless, the characteristics related to the class expressions should be better and more carefully analyzed in respect to the identified and others criteria in order to specify concrete decision strategies that could be used out of the box or minimally customized. For example, one

could use a conservative enrichment strategy (similar to that applied in the example), while other would prefer a more aggressive strategy (one that minimally constraints the enrichment execution).

Further, the authors are aware that there are ontologies whose equivalent class expressions (and equivalent properties expressions) are not defined as description logics axioms, but as Horn clauses that respect the conditions of DL-Safe rules [17], i.e. rules that are specified using the TBox vocabulary defined in description logics, and that are applied to only named individuals in the KB. In this sense, the inference performed upon this type of rules can profit from an enrichment process similar to the one proposed here for equivalent class expressions. The team aims to address this subject in the near future.

The contributions of the paper focused on the analysis of the enrichment actions and on the decision to whether execute or not them. On the other hand, the data source access was not addressed. In fact, for the moment every enrichment process is coded in the enrichment class instance as a SPARQL query (or other query language). This solution is not particularly flexible and adaptable to changes, both in terms of data sources and criteria. Hence, it would be important to address the process of generating data source accessors based on the ontology (TBox) alignments established between the target ontology and the source ontologies.

7. ACKNOWLEDGMENT

This work is supported by FEDER Funds through the "Programa Operacional Factores de Competitividade - COMPETE" program and by National Funds through FCT "Fundação para a Ciência e Tecnologia" under the projects: World Search (QREN11495) and AEMOS (PTDC/EIA-EIA/104752/2008).

8. REFERENCES

[1] Canito A., Maio, P. and Silva, N. 2013. An Approach for Populating and Enriching Ontology-based Repositories. *12th International Workshop on Web Semantics (WebS) at DEXA* (Prague, Czech Republic, Sept 2013).

[2] Brandão, R., Maio, P. and Silva, N. 2012. Enhancing LOD Complex Query Building with Context. (Macau, China, Dec. 2012).

[3] Elsenbroich, C., Kutz, O. and Sattler, U. 2006. A Case for Abductive Reasoning over Ontologies. (2006).

[4] Gamma, E., Helm, R., Johnson, R. and Vlissides, J. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.

[5] Kiryakov, A., Popov, B., Terziev, I., Manov, D. and Ognyanoff, D. 2004. Semantic annotation, indexing, and retrieval. *Web Semantics: Science, Services and Agents on the World Wide Web*. 2, 1 (Dec. 2004), 49–79.

[6] Linked Data - Design Issues: 2006. *http://www.w3.org/DesignIssues/LinkedData.html*.

[7] McDowell, L.K. and Cafarella, M. 2008. Ontology-driven, unsupervised instance population. *Web Semantics: Science, Services and Agents on the World Wide Web*. 6, 3 (Sep. 2008), 218–236.

[8] Motik, B., Sattler, U. and Studer, R. 2005. Query Answering for OWL-DL with rules. *Web Semant.* 3, 1 (Jul. 2005), 41–60.

[9] OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition): *http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/*. Accessed: 2013-02-26.

[10] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A. and Katz, Y. 2007. Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*. 5, 2 (2007), 51–53.

[11] Song, F., Zacharewicz, G. and Chen, D. 2013. An ontology-driven framework towards building enterprise semantic information layer. *Advanced Engineering Informatics*. 27, 1 (Jan. 2013), 38–50.

[12] Song, F., Zacharewicz, G. and Chen, D. 2013. An ontology-driven framework towards building enterprise semantic information layer. *Advanced Engineering Informatics*. 27, 1 (Jan. 2013), 38–50.

[13] Stojanovic, L., Maedche, A., Motik, B. and Stojanovic, N. 2002. User-driven Ontology Evolution Management. *13th International Conference on Knowledge Engineering and Knowledge Management* (Heidelberg, Oct. 2002), 197—212.

[14] Talburt, J.R. 2011. *Entity resolution and information quality*. Morgan Kaufmann.

[15] World Search: 2010. *http://www.microsoft.com/portugal/mldc/worldsearch/en/*.

[16] *Computational Processing of the Portuguese Language - 6th International Workshop, PROPOR 2003*.