Ontology Mediated Information Extraction in Financial Domain with Mastro System-T

Domenico Lembo
lembo@diag.uniroma1.it
Dip. di Ingegneria Informatica, Automatica e Gestionale
Sapienza Università di Roma
Rome, Italy

Yunyao Li
yunyaoli@us.ibm.com
IBM Almaden Research Center
San Jose, California

Lucian Popa
lpopa@us.ibm.com
IBM Almaden Research Center
San Jose, California

Federico Maria Scafoglieri*
scafoglieri@diag.uniroma1.it
Dip. di Ingegneria Informatica, Automatica e Gestionale
Sapienza Università di Roma
Rome, Italy

ABSTRACT

Information extraction (IE) refers to the task of turning text documents into a structured form, in order to make the information contained therein automatically processable. Ontology Mediated Information Extraction (OMIE) is a new paradigm for IE that seeks to exploit the semantic knowledge expressed in ontologies to improve query answering over unstructured data (properly raw text). In this paper we present Mastro System-T, an OMIE tool born from a joint collaboration between the University of Rome "La Sapienza" and IBM Research Almaden and its first application in a financial domain, namely to facilitate the access to and the sharing of data extracted from the EDGAR system.

CCS CONCEPTS

• **Information systems → Information extraction**; • **Theory of computation →** *Automated reasoning*.

KEYWORDS

Ontology, Information Extraction, Ontology Based Data Access, Ontology Mediated Information Extraction, Financial Domain

ACM Reference Format:
Domenico Lembo, Yunyao Li, Lucian Popa, and Federico Maria Scafoglieri. 2020. Ontology Mediated Information Extraction in Financial Domain with Mastro System-T. In *Data Science for Macro-Modeling with Financial and Economic Datasets (DSMM'20), June 14, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3401832.3402681

*Work done while at IBM Almaden - Research. Work supported in part by European Research Council under the European Union's Horizon 2020 Programme through the ERC Advanced Grant WhiteMech (No. 834228).

1 INTRODUCTION

In the data-driven era the spread of textual information has reached an unprecedented level. Thus, it is no surprising that in recent years, both public and private organizations required more and more new technologies to process large amounts of unstructured data in order to extract key information from them. Solutions to this task are relevant to a large number of applications, including, e.g., enterprise information integration, E-Commerce, medical information management, or digital financial services.

Information Extraction (IE) deals with this problem by proposing solutions ranging from Machine Learning assisted ones [7] to rule-based approaches [21], which usually produce as output some form of structured database [3].

Ontologies (a.k.a. Knowledge Bases), have become the de facto standard for knowledge sharing in the Semantic Web [16]. For this reason they have attracted the attention of the IE community [20]. However ontologies in IE, even in the financial domain, have been so far considered only in very general terms: ontology-based IE proposals aim at structuring information according to the conceptualization provided by an ontology, or at producing an ontology as the output of the extraction process [8, 11, 22]. Nonetheless, how to exploit the reasoning abilities offered by an ontology to improve the extraction phase is still largely unexplored. The Ontology Mediated Information Extraction (OMIE) approach proposed in [23] is a recent contribution that instead goes in this direction. OMIE is a declarative, rule-based and ontology-aided paradigm for the extraction of information from text, constructed on two well-known frameworks, i.e., Ontology-based data access (OBDA), which studies query answering over ontologies posed on top of relational databases [19, 25], and document spanners, a foundational framework for IE [6]. OMIE foresees to extract information by exploiting of the reasoning services allowed by the ontology. Another salient feature of OMIE is the possibility to extract data at query-time, without having to materialize in advance the instances of the target ontology. An OMIE system is composed by two main ingredients, the ontology, written in a lightweight formal language in order to guarantee tractability of reasoning, and a set of so-called extraction assertions. The latter ones act as mediators between the extractors, written in a rule-based language, and the facts that (virtually) populate the ontology. A user interacts only with the ontology, and when she asks a query, this is rewritten, in a way totally transparent to her, into a query-tailored set of extractors compiling ontology reasoning, whose evaluation over the underlying documents returns all inferred query answers.

In this paper we present an OMIE tool called Mastro System-T, whose foundations are based on System-T[1], a rule-based IE system by IBM [2], and Mastro[2], a state-of-the-art OBDA engine from Sapienza University and O.B.D.A. Systems [5]. Our tool smoothly integrates the extraction abilities of System-T and the reasoning services of Mastro by leveraging the modular nature of the OMIE framework and the possibility that it allows of answering queries by rewriting. By virtue of these characteristics, indeed, Mastro System-T can demand the actual execution of information extractors to a standard System-T installation, whereas at its core it manages extraction assertions and implements a mechanism to couple them with the query rewriting functionalities of Mastro.

To prove the usefulness of our tool, we performed some experiments on a set of real-world financial text documents, from a context in which we believe our approach can bring several benefits. The Electronic Data Gathering, Analysis and Retrieval system (EDGAR)[3] is a U.S. government open repository of data about companies who are required by law to file information with the Securities and Exchange Commission (SEC). EDGAR, among its objectives, aims to increase transparency of the securities markets and to favor the sharing of public company's financial information. In this respect, ontologies can play a crucial role, for their ability of supporting data interpretation and sharing. At the same time, EDGAR contains terabyte of data, mostly unstructured and subject to continuous changes (about 3,000 filings per day). For these reasons it is unfeasible to process them manually. Thus, an approach enabling automatic extraction of information in structured form, in front to on-the-fly users' requests and exploiting ontology reasoning abilities to retrieve complete answers can bring great benefits to the EDGAR system. To this aim we initialized MastroT providing an ad-hoc ontology written for this domain and a series of extractors to populate the ontology. We finally selected a set of queries to evaluate the effectiveness of our approach and demonstrate the advantages that can be achieved in this real-world financial application. In particular, our experiments, though preliminary, show that reasoning over the ontology is crucial increase the recall in IE, properly +9.8% from the baseline, since they allow to trigger extraction assertions that would be overlooked in query answering without reasoning.

The rest of the paper is organized as follows. In Section 2, we introduce some preliminaries on declarative rule-based information extraction and ontologies. In Section 3, we recall the OMIE framework and present the architecture and the main components of our tool. In Section 4 we discuss the query answering workflow, whereas in Section 5 we present the Financial use case. In the Section 6 we compare our tool with related work. We conclude the paper in Section 7.

[1]https://researcher.watson.ibm.com/researcher/view_group.php?id=1264
[2]https://www.obdasystems.com/mastro
[3]https://www.sec.gov/edgar/searchedgar/companysearch.html

2 PRELIMINARIES

In this section we briefly recall some basics notions regarding declarative Information Extraction and Ontologies.

2.1 Declarative Information Extraction

Declarative Information Extraction explores the relationship between the extractors, i.e., rules that produce tuples of spans (intervals of character indexes within the document), and their manipulation using relational algebra operators.

A *span* is a pair of indexes that identify sub-strings of a given document. For example, given the string `IBM Almaden`, the spans $[0,3\rangle$ and $[4,11\rangle$ identify the substrings `IBM` and `Almaden`, respectively. The extractors process the text usually via specified transducers (e.g., regular expressions with capture variables or a dictionary lookup) and produce a set of spans. They can be combined together using relational algebra operators (usually union, projection and join) to formulate new extractors.

The annotation query language (AQL), the language at the basis of System-T [12], implements this idea by allowing to define extractors using a friendly declarative SQL-like syntax. Below, we provide an example of AQL extractor with the results of its evaluation. For technical details about the syntax of AQL and the theory behind System-T we refer the reader to [2] and [6].

Example 2.1. Consider the following document **D**:

```
The revenue of the company Apple grows fast.
The new CEO of the company IBM is Arvind Krishna
```

Now let's consider the following AQL extractor

```
create dictionary Comp_dict as('company');

create view View_Comp_prefix as
extract dictionary 'Comp_dict'
    on D.text as Comp_prefix
from _Document D;

create view ViewComp as
extract pattern (<I.Comp_prefix>) (<C.token>)
    return group 0 as Comp
    return group 1 as prefix
    and group 2 as comp_name
from View_Comp_prefix I,
Generic_Export.token C;
```

This extractor is composed by two views. Each view in AQL is responsible to map spans to variables. Specifically, the `View_Comp_prefix` view assigns to the `Comp_prefix` variable all spans that match the word `'company'`, using the dictionary `Comp_dict`. The `ViewComp` view through the support of the previous one, maps to the variables `comp_name` and `Comp` the tokens that follow the word `'company'`, respectively, and the whole match (`group_0`) of the view to the variable `Comp`, using a syntax for groups similar to the one of the POSIX regex.

The result of evaluation of the `ViewComp` over the document **D** is reported in the following table:

Comp	prefix	comp_name
$[19,32\rangle$	$[19,26\rangle$	$[27,32\rangle$
$[60,71\rangle$	$[60,67\rangle$	$[68,71\rangle$

2.2 Ontologies

An ontology, defined as a specification of a conceptualization [9], encodes in formal terms an abstract, simplified view of a certain portion or aspect of the world. Ontologies are the de-facto standard for knowledge sharing on the web. They play an important role to interpret and structure Web data, and indeed a huge standardization effort was carried out by the W3C to define OWL, the standard Web Ontology Language, currently in its second version [24].

As usual in ontologies and data modeling, OWL distinguishes between *intensional* and *extensional* knowledge. Intensional knowledge is given in terms of logical axioms involving *classes* (a.k.a. concepts) and properties, which are of two types, *objectproperties* (a.k.a. binary relationships or roles) and *dataproperties* (a.k.a. attributes). *Classes* denote sets of objects, *objectproperties* denote binary relations between objects, whereas *dataproperties* denote binary relations between objects and values from predefined datatypes. At the extensional level, an OWL ontology is a set of assertions about instances of the ontology.

Example 2.2. Consider the following ontology which captures a part of the financial domain described in Section 5 and acts as a running example:

θ_1) `:Company a owl:Class`
θ_2) `:Public_Company a owl:Class`
θ_3) `:Private_Company a owl:Class`
θ_4) `:has_comp_name a owl:DataProperty`
θ_5) `:Public_Company rdf:subclassOf :Company`
θ_6) `:Private_Company rdf:subclassOf :Company`
θ_7) `:has_comp_name rdf:domain :Company`

α_1) `:IBM#ID a :Company`
α_2) `:IBM#ID :has_name 'IBM'`
α_3) `:Apple#ID a :Company`
α_4) `:Apple#ID :has_name 'Apple'`
α_5) `:Intel#ID a :Public_Company`

The intensional level ($\theta_1 - \theta_7$) declares the classes *Company* (θ_1), *Public_Company* (θ_2) and *Private_Company* (θ_3), and the attribute *has_comp_name* (θ_3), and specifies that public and private companies are special types of companies (through axioms θ_5 and θ_6, respectively) and that *has_comp_name* is a property that applies only to companies (θ_7). The extensional level ($\alpha_1 - \alpha_4$) states that the individuals `:IBM#ID` and `:Apple#ID` are instances of *Company* (through α_1 and α_3, respectively) and their names are `IBM` (α_2) and `Apple` (α_4), respectively. Finally the last assertion (α_5) states that `:Intel#ID` is a *Public_Company*. □

In the following, we only consider ontologies specified in `OWL 2 QL`, one of the standard profiles of `OWL 2` [17] (the ontology used in our experiments, a fragment thereof has been discussed in the above example, is expressed in this sub-language of `OWL 2`). Indeed, this is one of the largest fragment of `OWL 2` for which answering unions of conjunctive queries (UCQs), i.e., SQL `SELECT-PROJECT-JOIN-UNION` queries, has been shown to be tractable. Notably, answering UCQs can be carried out by a query rewriting technique that first compiles in the user query the ontology axioms that allow to obtain all inferred answers, and then evaluates the rewritten query over the extensional level of the ontology, considered as a database [1]. Below we provide an example of rewriting, in which queries are encoded in `SPARQL`, the W3C standard for querying ontologies [10].

Example 2.3. Consider the following `SPARQL` query expressed over the ontology of Example 2.2 and asking for all the companies.

```
SELECT ?X WHERE {
    ?X a :Company}
```

By evaluating the above query directly over the instance level, we would obtain as answers only `:IBM#ID` and `:Apple#ID`. However, since every public or private company is a company (θ_2 and θ_3), we have to include also them in the answer. The original query is thus rewritten, into the following UCQs (expressed in `SPARQL`):

```
SELECT ?X WHERE {
    ?X a :Company}
UNION
SELECT ?X WHERE {
    ?X a :Public_Company}
UNION
SELECT ?X WHERE {
    ?X a :Private_Company}
```

The set of answers of its evaluation over the instance level of the ontology is {`:Apple#ID`, `:IBM#ID` `:Intel#ID`}, which now includes also the previously missing one. □

3 ONTOLOGY MEDIATED INFORMATION EXTRACTION WITH MASTRO SYSTEM-T

We first recall the OMIE framework presented in [23]. An OMIE specification is a pair $\langle \mathcal{O}, \mathcal{R} \rangle$ where $\mathcal{O}$ is the intensional level of the ontology, and $\mathcal{R}$ is a set of extraction assertions of the form $\Phi \rightsquigarrow \Psi$ where Φ is an extractor written in AQL, and Ψ is a conjunctions of atoms in the alphabet of the ontology. Note that in OMIE, the extensional level of the ontology is not explicitly materialized, but the instances of ontology predicates are implicitly defined through the extraction assertions.

Example 3.1. Consider the following OMIE specification, where $\mathcal{O}$ is the intensional level of the ontology defined in Example 2.2 and $\mathcal{R}$ is composed by the following extraction assertion which imports the view of Example 2.1.

```
import view ViewComp from module m;
create view _Comp as                  :{n}#ID a :Company,
    select C.comp_name as n,       ⇝  :{n}#ID :has_name n
    from m.ViewComp C;
```

This assertion specifies how to generate instances of the class *Company* and its attribute *has_name* starting from the spans returned by the AQL extractor evaluated on a given document. In the right-hand side of the assertion, `:{n}#ID` is a *template* [4] used to specify how to construct objects from the tuples of strings identified by the spans retrieved by the extractor. Intuitively, for every span assigned to `n` by the extractor, an object identifier is constructed by substituting `{n}` with the string indexed by the span. By applying this extraction assertion to the document in Example 2.1, we obtain the extensional level assertions α_1-α_4 of the ontology showed in Example 2.2. □

Mastro System-T is a Java tool for OMIE, developed at IBM Almaden within a joint collaboration with the University of Rome "La Sapienza". Mastro System-T manages OMIE specifications of the form described above, and exposes a queryable OWL 2 QL ontology to the users, which can ask UCQs encoded in SPARQL and expressed over the ontology alphabet.

The architecture of the tool is illustrated in Figure 1. We soon notice that whereas Mastro System-T takes as input the OMIE specification (ontology and extraction assertions), as well as the user queries, documents are instead given as input to an instance of System-T, which is external to Mastro System-T and suitably interfaced with it through a dedicated software module (System-T Interface). Indeed, Mastro System-T does not directly manipulate the source documents, but for each user query it produces the specification of the AQL extractors that will be then executed by System-T to obtain the spans from which to construct the query answers, as described in more details later on.

Ontology axioms in input can be specified in any of the standard syntaxes for OWL 2 [18], and are serialized in an internal in-memory structure by the 'Ontology Manager' module. SPARQL queries are instead parsed and managed by the 'Query Manager' module. Both components are based on the Apache Jena framework[4], and are indeed inherited by the system Mastro. Extraction assertions are encoded in a tailored concrete syntax and handled by the 'Extraction Assertions Manager'.

The 'QA Engine' is the core component of Mastro System-T which implements the query answering service offered by our tool. It is in turn composed by submodules devoted to specific phases of the query answering process. Such modules, as well as the workflow for query answering are described in more detail in the next section, whereas for an in depth description of the query processing algorithm implemented in Mastro System-T and its formal properties we refer the reader to [15, 23].

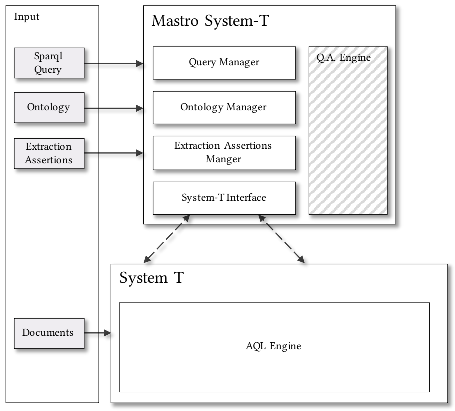

Figure 1: Architecture of Mastro System-T

4 QUERY ANSWERING

As said before Mastro System-T computes answers to the user's queries posed over the ontology by transforming them into AQL extractors and delegating their execution for information extraction from a given document to System-T. With this approach there is no need to apply every extraction assertion in the OMIE specification to the document in order to materialize all the facts instantiating the ontology. Mastro System-T triggers only the extraction assertions useful to generate the answers to the user's query at hand and returns always the most updated answer possible. This is particularly suited for dynamic scenarios, like the EDGAR one, where source documents change frequently and query answers cannot be computed on the basis of outdated materializations.

In a nutshell, the query transformation process realized by the 'QA Engine' includes an ontology-based query rewriting phase, and a further reformulation step that uses extraction assertions to transform the query over the ontology into a set of extractors to be executed over the text documents. The complete workflow is illustrated in Figure 2 and briefly explained in the following.

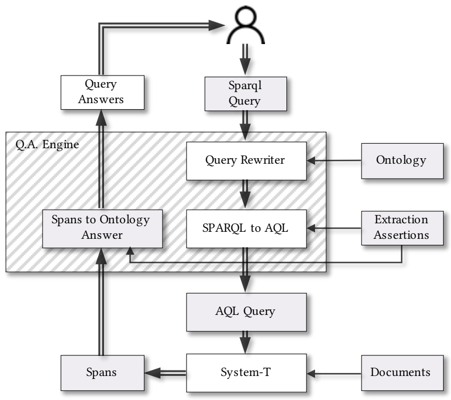

Figure 2: Query Answering Workflow

Query Rewriting. Given a SPARQL query q and the ontology O, the first step is based on a compilation of O into q, so that after the rewriting the ontology can be disregarded in the next query processing phases (see also Example 2.3). Query rewriting is a completely intensional form of reasoning and thus independent from the actual data contained in text documents. In Mastro System-T this process is carried out by the 'Query Rewriter' module and the result is a UCQs. The 'Query Rewriter' is in fact a module already existing in Mastro, where OWL 2 QL ontologies are coupled to mappings towards relational databases. By virtue of the modular nature of our query processing algorithm [23], we could re-use it as is.

Translation into AQL. The second step consists in translating the UCQs resulting from the *Query Rewriting* phase into a set of AQL extractors. Roughly, the SPARQL to AQL module substitutes, in all possible ways, each atom a in each query returned by the previous step with the extractors occurring in the left-hand side of extraction assertions referring to the ontology predicate occurring in a. This translation takes into account the possible joins inside the queries which are translated into join between spans using the special AQL predicate *Equals*.

AQL extractor Evaluation. Now the AQL extractor is given as input to System-T. The AQL extractor goes through a series of optimization steps in order to increase its performance in terms of execution speed. The results returned by System-T is a set of tuples of spans, together with the strings that they identify on the underlying document.

[4]https://jena.apache.org/

From Spans to Ontology answers. Given the set of spans and associated string values returned by previous phase, the *Span to Ontology answers* module computes the objects instantiating the ontology by using the templates declared in the extraction assertions, and give back the answers to the user.

5 CASE STUDY: OMIE OVER THE EDGAR SYSTEM

Electronic Data Gathering, Analysis, and Retrieval system (EDGAR) is a public platform where companies acting in U.S. are required by law to enter a range of information for government controls. EGARD is mainly composed by a large amount of raw text subject to significant updates over time. Since human effort is not sufficient to process this amount of data, there is the need for a mechanism that can automate the extraction phase by always providing the most update information and allowing data sharing and standardization. To carry out our experiments, we have designed a financial domain ontology tailored for the EDGAR system. The ontology contains 75 classes, 56 object properties, 214 data properties, and 907 axioms. Here we show a small excerpt of the ontology, built around the concept Company, and reported in Figure 3, where it is written in Graphol [13, 14], an Entity-Relationship like graphical language for OWL 2. The figure shows 9 classes, 3 object and 3 data properties, represented by labeled rectangles, diamonds and circles, respectively. To denote the first (domain) and the second (range) component of a property, a white and a black square were respectively used, connected to the properties they refer to with dashed arrows ending with a small diamond. A black hexagon indicates the disjunctive union of the classes connected to the hexagon dashed edge. Arrows denote sub-class relations, that is, containments at the instance level. For example, the arrow from the back hexagon to the concept Company denotes that the (disjoint) union of PublicCompany and PrivateCompany is included in Company, the arrow from the domain of has_subsidiary to Company specifies that such domain is a subclass of Company, that is, only instances of Company can occur in the domain of has_subsidiary. In words, the ontology is saying that investor, startup and insider companies are private companies (and therefore companies). Each company has a name (that is a string), and a CEO, and can have a subsidiary or a competitor company. A CEO is always associated to the company she/he directs, and is an Employee, which is a Person, and as such has a name and a surname (two strings).

As for the underlying dataset, we have selected 249 text documents containing data about 5 companies in the Fortune top 500, for a total dataset size of 250 MB. We have written 30 extraction assertions linked to an equal number of ad-hoc AQL extractors. Finally, in order to test the performance of our tool, with the main goal of highlighting the role of reasoning in the extraction phase, we selected 3 simple but representative queries described below.

The first query asks Mastro System-T to retrieve all the companies.

```
SELECT ?X WHERE {
    ?X a :Company}
```

For this query, we obtained the most interesting results, reported in Table 1, where we give the values of precision and recall that we obtained by executing the query either without or with reasoning, respectively shown under the columns 'Base' and 'Reasoning'. Values without reasoning have been obtained by disabling the query rewriting phase discussed in Section 4. In this way, we did not include in the results answers inferred by virtue of the axioms of the ontology, which has been thus considered in this case as a flat schema.

	Base	Reasoning	Gap
Precision	81.82%	82.71%	+0.89%
Recall	66.8%	76.26%	+9.46%
F-Masure	73.59%	79.35%	+5.76%

Table 1: Company query results

We note that the increase in recall in this case is due to the fact that the rewriting of the query is quite large, and includes, among other queries, also the queries

```
SELECT ?X WHERE {?X has_competitors ?Y}
SELECT ?X WHERE {?X has_subsidiary ?Y}
```

such that the extractors associated by the extraction assertions to the roles *has_subsidary* and *has_competitors* perform particularly well. This is confirmed also by the precision, which remains almost the same as the base case.

We obtained a similar behaviour for the query

```
SELECT ?X WHERE {
    ?X :has_CEO ?Y}
```

This query aks for the individuals participating in the domain of the relationship *has_ceo*. The execution of the query without reasoning makes use only of the extractor connected to *has_ceo*, which in our dataset extracts only 5 individuals. By enabling the reasoning, the number of extracted individuals increases to 2582.

The last query we show requests all the persons:

```
SELECT ?X WHERE {
    ?X a :Person}
```

In this case, the reasoning does not produces significant improvements, indeed we obtained only the 0.1% increment in recall with respect to the base case. This is due to the fact that the extractor mapped to the concept Person is already able to populate this class in an almost complete manner. That is, it looks also for strings matching patterns suited to identify special types of persons, i.e., those instantiating also (complex) classes subsumed by Person. On the other hand, defining "complete extractors" means to manually encode in them the reasoning that could be automatically done over the ontology, and this is clearly not always possible. In particular, in our ontology Person has a less articulated hierarchy and connections with other classes with respect to the class Company, and this is why extraction assertions are already able to directly provide almost all its instances.

6 CONCLUSION

In this paper we have presented the OMIE tool Mastro System-T, and discussed its first usage for extraction of public company's financial information from the EDGAR open repository.

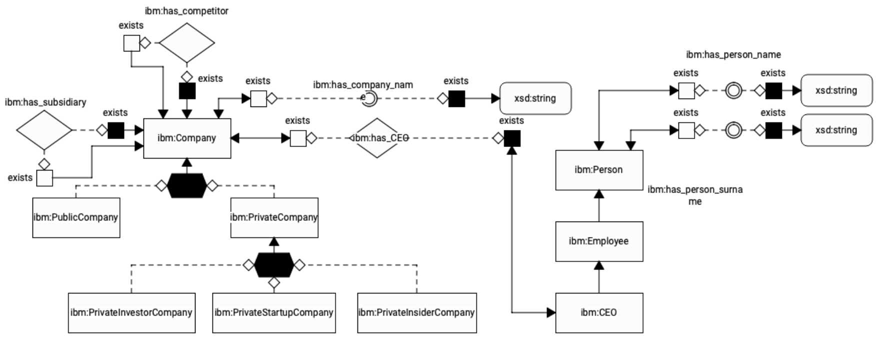

Figure 3: Ontology

From the tests we carried out, though preliminary, we could show that this approach may lead to an increase in recall for certain extractions. We have also shown how Mastro System-T can produce at query time the extractors needed to answer a user request, thus avoiding the complete and time consuming materialization of every instances of the ontology, and guaranteeing always updated answers. Our implementation is at an initial stage, and several issues still need to be addressed, as how to deal with the entity linking problem and what other roles ontology reasoning can play in information extraction. In particular, we are currently planning to extend Mastro System-T in the following directions:

- Implementation of new reasoning services to identify anomalies in the extraction assertions (e.g., assertions causing inconsistencies due to modeling errors) and to automatically deduce new extraction rules, able to improve IE performances.
- Equipping Mastro System-T with capabilities to solve entity linking tasks, always in the spirit of the OMIE.
- Creating a user-friendly interface both for designers willing to specify extraction assertions and final users who only require to ask their queries to the system.

Additional tests, also in others real world scenarios, will also be carried out to further assess the effectiveness of the tool.

REFERENCES

[1] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. 2007. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated reasoning* 39, 3 (2007), 385–429.

[2] Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick R Reiss, and Shivakumar Vaithyanathan. 2010. SystemT: an algebraic approach to declarative information extraction. In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 128–137.

[3] Jim Cowie and Wendy Lehnert. 1996. Information extraction. *Commun. ACM* 39, 1 (1996), 80–91.

[4] Souripriya Das, Seema Sundara, and Richard Cyganiak. 2012. *R2RML: RDB to RDF Mapping Language*. W3C Recommendation. W3C. Available at http://www.w3.org/TR/r2rml/.

[5] Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. 2012. MASTRO: A Reasoner for Effective Ontology-Based Data Access. In *Proc. of the 1st Int. Workshop on OWL Reasoner Evaluation (ORE)*.

[6] Ronald Fagin, Benny Kimelfeld, Frederick Reiss, and Stijn Vansummeren. 2015. Document spanners: A formal approach to information extraction. *J. ACM* 62, 2 (2015), 1–51.

[7] Dayne Freitag. 2000. Machine learning for information extraction in informal domains. *Machine learning* 39, 2-3 (2000), 169–202.

[8] Giulio Ganino, Domenico Lembo, Massimo Mecella, and Federico Scafoglieri. 2018. Ontology population for open-source intelligence: A GATE-based solution. *Software: Practice and Experience* 48, 12 (2018), 2302–2330.

[9] Tom Gruber. 2018. Ontology. In *Encyclopedia of Database Systems, Second Edition*. Springer.

[10] Steve Harris and Andy Seaborne. 2013. *SPARQL 1.1 Query Language*. W3C Recommendation. W3C. Available at http://www.w3.org/TR/sparql11-query.

[11] Alexander Hogenboom, Frederik Hogenboom, Flavius Frasincar, Kim Schouten, and Otto Van Der Meer. 2013. Semantics-based information extraction for detecting economic events. *Multimedia Tools and Applications* 64, 1 (2013), 27–52.

[12] Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick Reiss, Shivakumar Vaithyanathan, and Huaiyu Zhu. 2009. SystemT: a system for declarative information extraction. *ACM SIGMOD Record* 37, 4 (2009), 7–13.

[13] Domenico Lembo, Daniele Pantaleone, Valerio Santarelli, and Domenico Fabio Savo. 2016. Easy OWL drawing with the graphol visual ontology language. In *Proc. of the 15th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*. 573–576.

[14] Domenico Lembo, Daniele Pantaleone, Valerio Santarelli, and Domenico Fabio Savo. 2018. Drawing OWL 2 ontologies with Eddy the editor. *AI Commun.* 31, 1 (2018), 97–113.

[15] Domenico Lembo and Federico Maria Scafoglieri. 2020. Ontology-based Document Spanning Systems for Information Extraction. *Int. Journal of Semantic Computing* (2020).

[16] Deborah L McGuinness, Frank Van Harmelen, et al. 2004. OWL web ontology language overview. *W3C Recommendation* 10, 10 (2004), 2004.

[17] Boris Motik, Achille Fokoue, Ian Horrocks, Zhe Wu, Carsten Lutz, and Bernardo Cuenca Grau. 2009. *OWL Web Ontology Language Profiles*. W3C Recommendation. W3C. Available at http://www.w3.org/TR/owl-profiles/.

[18] Boris Motik, Bijan Parsia, and Peter F. Patel-Schneider. 2012. *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)*. W3C Recommendation. W3C. Available at http://www.w3.org/TR/owl2-syntax/.

[19] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. 2008. Linking Data to Ontologies. *Journal on Data Semantics* X (2008), 133–173.

[20] Borislav Popov, Atanas Kiryakov, Damyan Ognyanoff, Dimitar Manov, Angel Kirilov, and Miroslav Goranov. 2003. Towards semantic web information extraction. In *Proc. of the Human Language Technologies Workshop at ISWC 2003*, Vol. 20.

[21] Frederick Reiss, Sriram Raghavan, Rajasekar Krishnamurthy, Huaiyu Zhu, and Shivakumar Vaithyanathan. 2008. An algebraic approach to rule-based information extraction. In *2008 IEEE 24th Int. Conf. on Data Engineering*. IEEE, 933–942.

[22] Horacio Saggion, Adam Funk, Diana Maynard, and Kalina Bontcheva. 2007. Ontology-Based Information Extraction for Business Intelligence. In *Proc. of the 6th Int. Semantic Web Conf. and the, 2nd Asian Semantic Web Conf. (ISWC + ASWC)*. 843–856.

[23] Federico Maria Scafoglieri and Domenico Lembo. 2019. A formal framework for coupling document spanners with ontologies. In *2019 IEEE 2nd Int. Conf. on Artificial Intelligence and Knowledge Engineering (AIKE)*. IEEE, 155–162.

[24] Michael K. Smith, Chris Welty, and Deborah L. McGuiness. 2004. *OWL Web Ontology Language Guide*. W3C Recommendation. W3C. Available at http://www.w3.org/TR/owl-guide/.

[25] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. 2018. Ontology-based data access: A survey. In *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. 5511–5519.