Albert-Ludwigs-Universität Freiburg im Breisgau

Automatic Population of a Pharmaceutical Company Ontology

by

Ilinca Suzana Tudose

Supervisor Prof. Dr. Hannah Bast

Master Thesis
Bioinformatics and System Biology

Faculty of Engineering
Department of Computer Science

October 1, 2012

Main Supervisor:

Prof. Dr. Hannah Bast

Additional Supervisors:

Dr. Med. Philipp Daumke

Elmar Hausmann

Primary Reviewer: Prof. Dr. Hannah Bast

Secondary Reviewer: Prof. Dr. Med. Stefan Schulz

Declaration of Authorship

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work. I hereby also declare, that my thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Freiburg, October 1st, 2012

Abstract

We have implemented and evaluated a method to populate a company ontology, focusing on hierarchical relations such as acquisitions or subsidiaries. Our method searches for information about user-specified companies on the Internet using a search engine API (Google Custom Search API). From the resulted snippets we identify companies using machine learning and extract relations between them using a set of manually defined semantic patterns. We developed filtering methods both for companies and unlikely relations and from the set of company and relation instances we build this way, we construct an ontology addressing identity matching and consistency problems in a company-specific manner. We achieved a precision of 77 to 93 percent, depending on the evaluated relations.

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der Erstellung einer Firmenontologie sowie der automatischen Befüllung dieser Ontologie mit Instanzen. Wir konzentrieren uns auf hierarchische Relationen, wie zum Beispiel Firmenakquisitionen oder Tochtergesellschaften. Unsere Methode sucht mithilfe einer Internetsuchmaschine (wir haben die Google Custom Search API verwendet) nach firmenspezifischen Informationen. Ein maschinelles Lernverfahren erkennt in diesen Ergebnissen die Firmennamen, deren Relationen werden durch manuell definierte semantische Patterns extrahiert. Mit Hilfe diverser Filter sowohl auf Firmen- als auch auf Relationsebene werden anschliessend falsch positive Instanzen entfernt. Die Evaluation unseres Verfahrens ergab zwischen 77 und 93 Prozent korrekter Relationen in unserer Ontologie.

Acknowledgements

First I would like to thank Prof. Dr. Hannah Bast for accepting to supervise my thesis and for her guidance in my last months' efforts. It was an honour and her ability to see so clearly through problems has been truly inspiring.

I also owe my gratitude to Elmar Hausmann and Philipp Daumke who have closely supervised my work and to whom I have turned for valuable advice countless times. Katrin Tomanek has let me use her machine learning implementation, which is at the core of our information extraction. To her and to the rest of the team at Averbis I owe another big thank you for the many instructive conversations we had, as well as for the nice moments during the past months.

To Stefan Schulz I want to thank for accepting to be my second reviewer and even more for introducing me to the fascinating world of ontologies.

And finally, I thank Dragos for all the beautiful moments and the laughters we shared.

Contents

Chapter 1

Introduction

A major difficulty in ontology creation is gathering the knowledge, as it requires huge resources. We propose an approach to automate to a great extent the population of a company ontology, using a search engine API (e.g. Google Custom Search API). We do not aim to discover new classes or relations, but to add new instances for the existing ones. The solution we propose has a three steps structure: building a text corpus, extracting information and then organizing it in an ontology.

The first and second steps are not company-specific and highly parametrized, such that the system could easily be used for other types of entities (given training data is also provided) and/or new types relations. The last step, building the ontology from the set of retrieved triplets is highly specialized for the case of company individuals and is only partly applicable in other fields. Although restrictive, we consider the normalization of company names and relations, as well as addressing occurring violations an interesting subject.

In this thesis we will describe the system and evaluate its performance.

1.1 Motivation

Ontology generation in general and (semi-)automated ontology generation in particular are exciting subjects, still at the beginning of their time, where a lot of optimization needs to be done. First with the Semantic Web effort large ontologies covering various domains began to be created. As a rule they are *not* built automatically[23] and the systems that propose to solve the problem have several limitations (discussed in the next chapter).

A survey suggested that, already in 2001, about 90% of the information needed for competitive intelligence could be found on the Internet, being public [15]. Together with the presumption that "collective knowledge is much more powerful than individual knowledge" [12], our "Googling" approach took shape.

One reason for choosing to build a company ontology is that the subject matches perfectly the "requirements" for an approach such as ours. The huge number of companies rises serious problems for a manual approach; even restricting the domain to a couple of thousands of larger companies does not turn the task into an easy one. The number of classes is low (in a simplistic view, all companies can be seen as instances of a same class e.g. "Organization") and there are not many types of relations to be defined either. We only focus on the corporate organization (i.e. acquisitions, subsidiaries, mergers), as we consider this a key unsolved problem.

Another reason for which we chose "companies" as the subject of our ontology is the direct applicability in *competitive intelligence*. Competitive Intelligence is an ethical business practice that is broadly described by Wikipedia as "the action of defining, gathering, analyzing, and distributing intelligence about products, customers, competitors and any aspect of the environment needed to support executives and managers in making strategic decisions for an organization"[1]. Organizations nowadays use *competitive intelligence* to compare themselves with their competitors and thus being able to take [more] informed decisions and to correctly evaluate risks, markets, opportunities, their weaknesses and strengths. The key factor in these studies is, of course, knowing *who* is your competitor and this is the difficult question a company ontology should (at least partially) answer. The focus of our ontology is to provide more insight on the economic relations between companies and their corporate organization.

1.2 Contribution

I have implemented and evaluated a method for (pharmaceutical) company specific information extraction. The idea of extracting information by "Googling" has been explored before[2] and the pattern-based approach for relation extraction is widely used. However, to our best knowledge, these methods have not been tested before for this specific field and on its distinctive attributes. We also try to overcome the difficulties of working with such a heterogenous source of information by means of some simple metrics and tests we have developed specifically for this field.

[1] `http://en.wikipedia.org/wiki/Competitive_intelligence`

[2] As described in the next chapter.

More concisely, I would describe my contribution as having:

- elaborated a strategy to efficiently acquire information from the World Wide Web.
- trained a model for company recognition.
- implemented a method to identify relations between companies, based on patterns and manually defined textual formulations. I extract only four types of relations and synonyms.
- developed a method to assign synonyms and preferred terms (specific to companies).
- proposed some simple means to identify and filter mistakes.
- evaluated the efficiency of our method and the accuracy of the information in the resulted ontology.

1.3 Outline

In the next chapter we define some important concepts for this thesis and present relevant prior work. We also describe some related work and tools which are used in our work but are have not been developed by me. In Chapter 3 we already describe the system we have envisioned and implemented and in Chapter 4 we evaluate its performance and discuss the results. In Chapter 5 we draw some conclusions and present some ideas for further improvement of the system.

Chapter 2

Background And Related Work

In the first section of this chapter we aim to explain some notions that we will further use. In the rest of the chapter we try to give an overview of related work from different points of view.

Ontologies for pharma companies are a very narrow field so we will have a look at any general (semi-)automated ontology generating tools. It is also interesting to know if a satisfying (possibly manually curated) ontology already exists in the ”company” field. We will thus analyze the state of the art from three points of view:

- State of the art of domain independent (semi-)automated ontology population
- State of the art of competitive intelligence (semi-)automated ontology population
- State of the art of existing competitive intelligence ontologies

2.1 Background

Ontologies are a good way to represent domain information in a way that it is still interpretable by machines. They are becoming widely used in all kind of applications, because they allow machines to intelligently interpret data. Although the demand is growing and although there have been several attempts to automatize the process, domain ontologies are still created mainly manually. This is both time consuming and expensive. As Bedini and Nguyen(2007) state the ”bottleneck of knowledge acquisition remains one of the major problems to be solved”[8].

We formally define an **ontology** O as in Geleijnse and Korst (2005)[30]:

Definition 2.1. Ontology O is a 4-tuple $(C,\ I,\ P,\ T)$, where

$C\ =\ (c_0, c_1, ..., c_{n-1})$ is an ordered set of N classes,

$I\ =\ (I_0, I_1, ..., I_{n-1})$ with $I_j,\ 0 \leq j < n$ the set of instances of class $c_j \in C$,

$P\ =\ (p_0, p_1, ..., p_{m-1})$ a set of m binary relations on classes, with $p_k\ =\ (c_{k,0}, c_{k,1}) \in C \times C,\ 0 \leq k < m$ and

$T\ =\ (T_0, T_1, ..., T_{m-1})$ with $T_k,\ 0 \leq k < m$ the set of instances of relation $p_k \in P$, $T_k\ =\ \{(s, o) \mid p_k(s, o)\},\ s\ \in\ I_{k,0}$ (an instance of $c_{k,0}$) and $o \in I_{k,1}$ (an instance of $c_{k,1}$).

In this thesis "entity" and "individual" are synonyms for instances of a class, "concept" is a synonym of class and "property" might be used meaning relation.

A **triplet** is another representation of relation instances in which the two class instances and the relation type are stored in a 3-tuple (thus triplet) of the form

$$(subject,\ relation\_type,\ direct\_object).$$

Formally this is defined as follows (Def. 2.2):

Definition 2.2. A triplet t is a 3-tuple $(i_{k,0},\ p_k,\ i_{k,1})$, where $i_{k,0} \in I_{k,0}$, $i_{k,1} \in I_{k,1}$ and $p_k \in P$ with I, $I_{k,1}$, $I_{k,0}$ and P defined as in Def. 2.1.

Since we build our ontology from snippets and we will be using this word a lot from now on, we shall also explain it. By **snippet** we understand the "short description or an actual excerpt from the webpage"[1] provided on the result pages of any successful query on a search engine (Google in our case).

Now let's look at the usual methods of identifying instances of classes ($\in I$) (entities) and relations ($\in T$). Statistical models are widely used in entity recognition. In order to extract relations between entities, rules are needed. They can be manually defined or learned from an annotated corpus using ML approaches (e.g. Ontosophie[28]). The self defined rules can be in the form of lexico-syntactic or lexico-semantic patterns and have different levels of abstraction.

An example of a **lexico-syntactic pattern** is Equation 2.1, one of the four rules used by Hearst[17]. Here `NP` means "noun phrase", `such`, `or`, `as` or `and` are used merely for lexical matching and `{`, `}`, `(`, `)`, `*` are used with the usual Java regular expression meanings. It extracts "hyponyms"[2] from a sentence like *"... works by such authors as*

[1] https://support.google.com/websearch/bin/answer.py?hl=en&answer=134479

[2] Hyphonyms can be seen as a kind of `is-a` relation assertions from an ontological point of view.

Herrick, Goldsmith and Shakespeare" as follows:

```
(Herrick, is-a, Author)
(Goldsmith, is-a, Author)
(Shakespeare, is-a, Author)
```

```
such  NP  as  {NP,  }*  {(or|and)}  NP                    (2.1)
```

Lexico-semantic patterns "make use of concepts instead of merely lexical representation, hereby alleviating the time-consuming process of pattern definition"[23]. Rule 2.2 is an example of lexico-semantic pattern used in IJntema et al.(2012). The power of such patterns comes from the reusable information "hidden" in the ontology, e.g. `kb:buys`[3] has synonyms like 'acquire', 'take over' or 'buy', while `kb:Company` means instances of the class Company. The domain experts write such rules and a compiler expands them, so the process has an automatic part.

Rule 2.3 shows another way of representing such patterns. It is used by SPART and, given a class `Fish` exists in the ontology, a new subclass `Flying Fish` could be found using this rule. We consider these rules lexico-semantic because they make use of pre-existing information from the ontology (unlike Hearst's original patterns).

```
($sub, kb:hasSubsidiary, $obj) :-                         (2.2)
       $sub:=[kb:Company] kb:Buys
       $obj:=[kb:Company]
```

```
(Adj|N) NP<class> -> NP<subclass>                         (2.3)
```

With respect to the information sources of the ontology generation approaches, we can distinguish two directions: the ones which use *structured text* and the ones which use *unstructured text*. The systems that use structured text - basically converting XMLs, data from DTD files or tables to an ontology are of no interest to us and will not be presented here. They usually achieve promising if not satisfyingly accurate results. The main problem of these systems is obviously that they assume an existing and satisfactory formal representation of the domain of interest. At the moment this is usually not the case and it certainly isn't for company acquisitions and other corporate management changes.

With respect to the kind of information a system extracts, we can again distinguish two directions: *ontology population* and *ontology generation* in the strict sense. Ontology

[3] "kb:X" is the notation the authors use for an instance of type X from the ontology (kb = knowledge base).

generation includes class and relation generation, as well as extracting relations between entities, while the population of an ontology assumes an existing model (classes and relation types) and finds instances and relations to populate this ”root” ontology. These notions are widely used in the domain literature with the same meaning (ontology population). The focus of the scientific community seems to be ontology population and only a few tools offer solutions to the complex task of ontology generation in the strict sense. Since strict ontology generation has different challenges and expectations, we cannot correctly compare our system to such tools. Therefore our decision is to only present here related work in the field of (semi-)automated ontology population.

There are several attempts to automate the creation of ontologies. We consider as relevant prior work only the approaches for **ontology population** from **unstructured text**. While we shortly describe the most significant systems in the next section, we have also built a table to easier compare them (Table 2.1).

2.2 Triplet Extraction State Of The Art

Hearst [17] proposed in 1992 a method to find hypernyms using predefined lexico-syntactic patterns (see Rule 2.1). They have a high precision e.g. in some tests made by Maynard et al.[26] the precision was between 84% and 94% (depending on the test set) but a very low recall (below 10% according to Waechter (2010)[9]). It is from Hearst that we got the idea of a pattern-based relation extraction. We build our own expressions and we add semantic knowledge, but do not make any difference between different parts of speech.

Maynard et al. (2008)[24] proposed a technique for ontology population which takes into account contextual information associated with the terms. The system described attempts to build the correct term hierarchy, the focus being on term extraction. Two approaches are presented out of which the best performing one is based on a hierarchical classification algorithm (*Hieron*). They achieved a F1-score of 74.7% for the correctness of subclass/superclass assigning, so one type of relation (is-a). Since no other metrics are available this approach does not appear in Table 2.1.

SPART[26] uses Hearst patterns and some other lexico-syntactic patterns to identify new classes, subclasses, synonyms or instances in free text. It also uses a set of lexico-semantic patterns, named ”contextual patterns”, as in Equation 2.3. It obtains a precision of 48.5% for ”is-a” relation extraction between classes and 47.6% for instance classification; the evaluation was made on 25 Wikipedia articles.

Tool	**MUSING**	**PANKOW**	**Geleijnse**	**Polaris & Jaguar**	**IJntema**	**Ontosophie**
F1 score (triplet extr.)	89,42%	28.24%, 18.25%	?	49.67%	78.72%	12.08%
Precision (triplet extr.)	85.6%	?, 21.76%	85%-90%	52.32%	83.9%	85.62%
Recall (triplet extr.)	93.6%	24.9%, 15.73%	?	47.28%	74.14%	6.5%
F1 score (ER)	92.9%	-	?	?	?	?
Precision (ER)	93.5%	-	78%	?	?	?
Recall (ER)	92.3%	-	?	?	?	?
Class no.	> 4	59	3	3	4	?
Relation no.	12	1 (is-a)	2	3	10	?
User validation	yes	yes/no	no	no	no	yes
Rule source	?	manual	manual	manual, ML	manual & compiled	ML
Rule type	?	lexico-syntactic	lexico-semantic	syntactic & statistical	lexico-semantic	statistical
Ontology accuracy	?	Rec:24.9% (with user validation)	?	F1:48.92%, Prec:36.94%, Cov:72.39%	-	?
Domain	CI	general	Movies	CI (acquisitions)	CI (financial)	general
Source	[13], [14]	[12]	[30]	[24], [29]	[23]	[28]

Table 2.1: **Performance overview for some ontology generation tools.** Here "?" signifies values which were not specified in the cited papers, while "-" means irrelevant information in the context of the respective tool. For example IJntema extracts triplets, but does not address the problems of ontology building, thus discussing about *Ontology accuracy* in its context is irrelevant if not impossible. *Class number* and *Relation number* refer to the relations and classes in the evaluated ontology and *User validation* shows if the system is aimed to be used with a user accepting/rejecting proposed entries. However, the accuracy measures for all systems except PANKOW appear to be obtained without user validation.

Polaris[29] is a tool for entity and relation extraction from free text. It uses hand-crafted syntactic rules to find potential relations in the parse trees obtained by NLP processing. These "candidates" are then classified using different ML techniques into one of the 26 relations. It is integrated in Jaguar (see Section 2.3) so that an ontology can be built from these extracted triplets. They evaluate eight generated domain ontologies, among which an *acquisitions ontology*. The results in Table 2.1 depict the performance for the acquisitions ontology, thus a type of CI.

IJntema et al.[23] argue that when comparing lexico-syntactic to lexico-semantic patterns, the latter yield higher quality results and their construction is more time efficient. They evaluated their system for the financial and for the political domain. The results in Table 2.1 mirror the performance for the financial domain. Ten specific relations are extracted: *hasCEO*, *hasProduct*, *hasShareValue*, *hasCompetitor*, *hasProfit*, *hasLoss*, *hasPartner*, *hasPresident*, *hasRevenue* and *hasSubsidiary* which we also extract. The subject and object can be literals or one instances of the classes *Company*, *Person*, *Product*. The paper also proposes a language for defining these rules (instead of merely looking for pattern matches as we do) (See Rule 2.2).

2.2.1 Triplet Extraction for Competitive Intelligence

As part of a European project called **MUSING** (**MU**lti-industry, **S**emantic-based next generation business **IN**telli**G**ence) a tool was developed that automatically extracts company-related data which is then stored in a triplet database. They address a wider and different range of relations and entity types than we do, aiming to provide information on company's activity, contact details, board of directors or number of employees. There is no specification about company corporate hierarchy, so we presume the problem is not addressed. They crawl a small set of sites such as *Wikipedia* and *Yahoo! Finance* for information and then extract entities and relations. The system is based on GATE but with "new linguistic and named entity recognition processors that map concepts to ontology classes". The system is said to achieve a promising F-score of 84% for company related triplet extraction. [13]

Our approach and MUSING's are different as are the targeted problems. As presented in [13] and [14] MUSING provides a lot of financial insight for each company, while we provide the links between companies. We should also point that the source of information is another fundamental difference between the two systems. We start, so to say, with different baselines.

- Both the triplet extraction and the validation process are much easier than in our case, due to the fact that the information on the cited web pages is either structured or, if not, the text is grammatically correct and contains valid information.[4]
- Information about some companies[5] might miss on these pages. The potential information coverage is significantly higher in our case.

2.2.2 Using Collective Knowledge on the World Wide Web

The most similar related work we know is done by **Geleijnse and Korst**[30]. They propose a method to populate ontologies from "googled text fragments"[30] (i.e. snippets) using hand-crafted, domain specific patterns. They build the same type of queries as we do and have a slightly more restrictive approach to relation extraction : they only match occurrences of the form "[$c_{k,0}$] *expression* [$c_{k,1}$]", thus not allowing tokens between the instances and indicator words, nor do they seem to do any a priori or a posteriori triplet filtering. They do however test the instance classification by performing more Google queries of the form "The movie [Movie]" and consider the classification correct if the number of results exceeds a given threshold (5). They obtained high quality results (see table 2.1), but we should point out that the movies domain is better covered than ours[6], thus we can not expect comparable results, simply because the amount of information available about the corporate dynamics of pharmaceutical companies is much smaller. They only computed the recall for some Internet Movie Data Base (IMDB) categories e.g. best actor, best director and obtained values between 87% and 98%.

Cimiano and Staab[12] proposed in 2004 a "Googling" approach for ontology population. The aim of their system (**PANKOW**) was to classify a set of given instances or concepts with regard to a given ontology. Using Google API PANKOW would do exact searches for a set of automatically generated Hearst-like patterns and decide the sub- and superclass relations as well as instance assertion based on the number of results for each search expression. For example before adding "Niger" to the ontology the number of results for Google searches like "rivers such as Niger" and "countries such as Niger" will be compared in order to make a decision. For the first values shown in Table 2.1 "Niger" is assigned as an instance of the concept with most hits, while for the second values the results are considered correct if the right class is among the top five suggestions. This study is interesting to us because it showed the potential of "collective knowledge"

[4]For example, the accuracy measures in Table 2.1 are averaged over 12 concepts, out of which five have both precision and accuracy 100%, exactly because they use structured text. Averaging over the rest of the concepts we get about 10% lower precision (73.36%) and recall (82.83%).

[5]i.e. smaller companies which are not listed on the stock market or companies from countries with strict regulation about making public this kind of information.

[6]Simply from the fact that there is a comprehensive data base for movies with all the information Geleijnse and Korst try to extract, which is not the case for us.

on the Internet. We use similarly built search expressions to find the instances to be added (instead of requiring an input set on instances only to be classified), as well as to annotate the relations between them. We have similar approaches but with different aims.

DOG4DAG[9] (**D**resden **O**ntology **G**enerator for **D**irected **A**cyclic **G**raphs) is a plugin for OBO-Edit (a popular framework for ontology editing). It supports semi-automated generation of terms, definitions and parent-child relations from texts in PubMed and PDF repositories. It generates terms by identifying statistically significant noun phrases in text, while for definitions and parent-child relations it uses a set of pattern based web searches. According to their paper the quality of term generation is similar to the one found in manually created ontologies, while up to 78% of definitions and up to 54% of parent-child relations are correct. By definitions they mean natural language definitions, which our system does not aim to provide, but the term and relation extraction are interesting. Their 2010 paper claimed no other system achieved comparable results [9].

2.3 Ontology Building

Once the information is gathered (in our case in the form of triplets), the ontology has to be populated. This is a complex problem by itself, as it has to solve synonyms, merge entities and solve (generally relation-specific) rule violations.

The only tool we are aware of and has a **conflict solving algorithm** is **Jaguar**[24] but there is no clear statement about how this is achieved: "The *conflict resolution engine* creates a NIPF topic hierarchy link by link (relation by relation) and follows a conflict avoidance technique wherein each new link is tested for causing inconsistencies before being added to the hierarchy"[24][7]. Since no further specifications are made, we can suppose that the relations causing inconsistencies are simply left out of the ontology. In this case the process will be biased towards the relations added first. We do more than that as we try to identify correct/incorrect relations before adding them to the ontology.

Jaguar generates ontologies from free text, being able to extract concepts, relations as well as merge them together and build a hierarchy. It filters significant text parts to be further processed. As opposed to most other systems, Jaguar also has a *conflict solving* module in which cycles or redundancies are solved; it also provides an ontology merging solution, but this is beyond the scope of this thesis. The main drawback of the system is that they require carefully selected relevant documents as input, as they do not question the validity of information.

[7]NIPF = National IntelligencePriorities Framework

PANKOW tries to solve possible hierarchy conflicts using a "statistical fingerprint" (see Section 2.2.2), but the results are not comparable without user validation.

2.4 Company Ontologies

We are aware of two public company databases. They offer information about generic companies, but with a significant percentage of bio-pharmaceutical companies. Corporate hierarchy relations are added manually, while the main focus is on meta-information such as address, contact details, persons in the board of directors or stock prices.

OpenCorporates[22] is an initiative to crowdsource information about companies. The information is mainly gathered from government data while some relations, such as corporate grouping, seem to be added by hand. An impressive 44 Mio. companies are indexed, but the normalization is at the moment practically inexistent, which makes it hard to search. At the moment this thesis was written the information about subsidiaries was scarce at its best and acquisitions were not yet regarded. OpenCorporates at this moment is rather a classic database than an ontology. Therefore we will not compare our results to OpenCorporates.

CrunchBase is a "free directory of technology companies, people, and investors that anyone can edit"[8]. They also offer information about acquisitions, investments or other "milestones". This information is gathered however by hand and the coverage is low. Since they do offer the same kind of information as we do, during our evaluation a short comparison will be made.

2.5 Company Name Normalization

We know that the **MUSING** tool discussed before (2.2.1) addresses the problem of company name normalization. Coreferences between entities in the same text are solved and also same entities which come under different names are merged, using a similarity score based on some hand-crafted rules. In the example provided in Maynard et al. 2007[14] *ALCON*, *Alcon* and *Alcon Inc.* are merged together, but it is not specified what happens in more complex situations e.g. would *Alcon Laboratories* or *Alcon SA* also be merged as synonyms of *Alcon* or not.

Kintz and Fintzen[16] propose a few simple steps to normalize company names. Apart from standard name normalization techniques such as solving case problems, missing

[8]As described on the "About" page: http://www.crunchbase.com/about

hyphens/spaces or ignoring accents, they also propose a list-based "normalization" of suffixes. These suffixes are removed from company names, thus they make no distinction between different companies which (presumably) belong to the same corporation, such as *Apple Inc.* and *Apple Corp.*. In some cases this might be harmless, but it is not always the case, the authors acknowledging this limitation themselves. We try to overcome this limitation and take the company-specific normalization one step further.

2.6 Tools and Related Work

In this section we will briefly describe the main tools that we used and the mechanisms behind them. I should note that I did not create or contribute to the development or implementation of either tools/algorithms described hereafter. We describe them here instead of in the next chapter, where we specify how they are being used especially to clearly delimitate the work which does not belong to me.

2.6.1 Contextual Sentence Decomposition

The results[9] Broccoli got by adding the context decomposition showed that we could considerably lower the number of false positives. In a well performing configuration (precision-wise) we have used the tool[10] developed by Elmar Hausmann. As part of the entity recognition anaphoras are also solved by assigning to each *it*, *he*, *she*, *this* etc. the last recognized entity of matching gender.

Contexts are intuitively defined as sets of words that "belong" together according to the meaning of a phrase. A very clear example is provided in [7] using the sentence *"The usable parts of rhubarb, a plant from the Polygonaceae family, are medicinally used roots and the edible stalk, however its leaves are toxic."*. In this case contextual decomposition would split the phrase into the following contexts (solving anaphoras too):

(C1): **rhubarb**, *a plant from the Polygonaceae family*
(C2): The usable parts of **rhubarb** *are the medicinally used roots*
(C3): The usable parts of **rhubarb** *are the edible stalks*
(C4): however **rhubarb** *leaves are toxic.*

Words (eventually entities or relations) are commonly considered to "belong" together if they are used in the same sentence. An even more simplistic way of determining this is by setting a certain proximity threshold. Both this cases yield countless false results

[9]False positives dropped from 18.551 to 11.056 by using context decomposition, while the false negatives raised from 218 to 364 in a test on Wikipedia documents. [7]
[10]The tool is also part of NLP in Broccoli.

due to the false assumptions on which they are based. Looking back at our example, it is likely that any of these two methods would infer that the *rhubarb leaves are edible* since they do belong to the same sentence and *edible* and *leaves* are only three words apart.

The **contextual sentence decomposition** envisioned by Hausmann [10] takes place in two steps:

Sentence Constituent Identification is the first step in building the contexts. It groups parts of the sentence in three kinds of basic constituents: *enumerations*, *sub-clauses* and *concatenations*[7]. In a tree representation of the sentence these constituents build the inner nodes, while the leaf nodes contain actual parts of the sentence. The tool uses a set of hand-written rules to transform the tree resulted from syntactic parsing (done by the SENNA[11] parser) into the sentence constituent tree.

In the **Sentence Constituent Recombination** step the tool recursively builds the contexts from the identified sentence constituents, using a set of rules.

The rules used in both steps, together with more details and performance evaluation are provided in [7] and [10].

2.6.2 Conditional Random Fields

Conditional random fields (CRFs), introduced by Lafferty et al.[1], are probabilistic models used to segment and label data. They have a graphical structure, meaning that they can be seen as undirected graphs where the nodes are random variables over the data sequences to be labeled (X) and the corresponding label sequences (Y); the edges define dependencies between them. They are discriminative models, so they model the distribution probability $P(X|Y)$ for the jointly distributed random variables X and Y, without first computing $P(X)$.

CRFs present several advantages over other common probabilistic approaches:

- They allow **arbitrary dependencies on the observation sequence**. Thus they are **more expressive** than HMM-like models or stochastic grammars and have the ability to relax the strong independence assumptions made by these approaches.[1]

[11]"SENNA is a software distributed under a non-commercial license, which outputs a large variety of Natural Language Processing predictions: part-of-speech (POS) tags, chunking (CHK), named entity recognition (NER), semantic role labeling (SRL) and syntactic parsing (PSG)." (http://ml.nec-labs.com/senna/)

- They **avoid the "label bias problem"**. This is a problem described by L. Bottou[12] that occurs in other discriminative models, which use *directed* graphs as models. These models can be biased towards states with fewer successor states.[1]
- Features do not need to specify completely a state or observation, so the model might be estimated using **less training data**.[1]
- A general advantage of graphical modeling is that it allows much more **efficient factorization and decomposition** of a probability distribution.[2]
- They can handle **large sets of (possibly overlapping) features**. [2]
- They reduce the problem of finding a statistical model to finding an appropriate set of features. [20]

In our system we use a special type of CRFs, namely the first-order linear-chain one. I shall also refer to them simply as CRFs henceforth, hoping it will not lead to any confusions since it is the only kind of CRFs we need discuss from now on.

I should also note that the CRF tool used in this thesis is based on the Mallet implementation of CRFs and I am using it courtesy to Averbis[13].

First-Order Linear-Chain CRFs

The *first-order linear-chain CRFs* are a special case of conditional random fields in which the target variables have a linear structure, in other words they can be represented as a one dimensional vector $\overrightarrow{y} = (y_1, ..., y_n)$ (thus **linear-chain**). It is not necessary to assume that X and Y have the same graphical structure. The **first-order** characteristic comes from the fact that we only need y_{i-1} and y_i at each step when computing the probability.

In the context of linear-chain CRFs "the three basic problems" below are solved by the algorithms developed for HMMs, but using modified forward and backward scores. [2]

- "Estimation of model parameters $\theta = (\lambda_1, ..., \lambda_k)$"[2]
- "Finding the best label sequence $\overrightarrow{y}*$"[2]
- "Estimating the probability of a label sequence"[2]

[12]Leon Bottou, *Une aproche theorique de l'apprentissage connexionniste: Applications a la reconnaissance de la parole.* Doctoral Disertation, Paris XI University, 1991

[13]Details about these can be found in Klinger and Tomanek (2007)[3] and Tomanek 2010[2].

Since it is not my work I will refrain from explaining the training and estimation methods into more detail. An expanded discussion of the methods with formulas and arguments can be found in Klinger and Tomanek (2007)[3] and Tomanek (2010)[2], Chapter 2.2.

Features for CRF We will now shortly explain the types of features that we used in our configuration.

Word class "replaces capital letters with 'A', lower letters with 'a', digits with '0' and all other characters with '_'" [20].

Brief word class collapses consecutive identical characters from the "word class" form into one[20]. E.g. "Novartis AG" would become "Aa_A".

Orthographic features are based on regular expressions and target 17 properties such as: hasDash, isAlphaNumeric, isCapitalized, hasGreekLetter, etc. [20].

Lexicon files can be provided and features are constructed from the lexicon words by adding binary features with the meaning "belongs to lexicon X", "doesn't belong to lexicon Y".

2.7 Other Tools

Google Custom Search API

The **Custom Search API**[14] provides tools for programatically performing Google-powered searches as well as to customize these searches. It provides (Java) methods to get the results to custom built, URL-embedded queries either as JSON objects or XMLs. The JSON objects contain a lot of information such as links, page titles, snippets, number of hits, page number, etc. It is an innovative tool which, as we hope to prove in this thesis, can be efficiently used to acquire and organize diverse knowledge. Of course, it comes with some restrictions too, among which we mention the query and result limits.

The API is free to use up to 100 request per day and up to 10,000 requests per day can be bought. Even 10,000 requests is not much, so the queries must be built efficiently enough to overcome this.

The number of results for each query is restricted to 100 (so 10 result pages). This is another fact that needs to be taken into account in the query building strategy. It seems however that the most important information is found on the first pages - Geleijnse and Korst (2005)[30] report that using first 50 (instead of 10) result pages increases

[14]https://developers.google.com/custom-search/docs/js/cselement-devguide

the number of retrieved instances by 7%. The reliability of this information is however discussable since the domain of the cited study was ”movies” and differences are hard to estimate.

SVM Classifier

Support Vector Machines (SVMs) are a well established supervised learning models which are used to classify input data. The basic linear model has only two possible output classes. We use such a classifier courtesy of Dr. Katrin Tomanek [2] for annotating patent applicants/assignees for our Use-Case (see Sec.**??**). We only used basic features: containsComma, isOneLetter, ContainsHyphen (for each token) and also numberOfTokens.

Chapter 3

Our System

Ontology population in the context of Semantic Web must "deal with the massive and heterogeneous data of the World Wide Web and thus improve existing approaches for knowledge acquisition which target mostly small and homogeneous data collections"[36]. The method we propose is based entirely on searches performed using a search engine[1]. We use the Google searches both as information source and in the decision making process. Most methods we are aware of use web searches in the decision making process, to help prune candidate entries for the ontology.

Our system tries to offer a solution for:

- Named entity recognition for organizations
- Name normalization, partly specific to bio-pharmaceutical organizations
- Triplet extraction (pattern based)
- Knowledge filtering
- Consistent ontology population

3.1 Method Overview

The method we propose iteratively gathers information on user-specified companies (Sec.3.3). From this information we automatically extract entities (Sec.3.4) and relations (Sec.3.6) which we then evaluate trying to identify correct/incorrect information. The valid entities and relations are then grouped as an ontology (Sec.3.8). An overview

[1]The search engine API was Google Custom Search APIin our case.

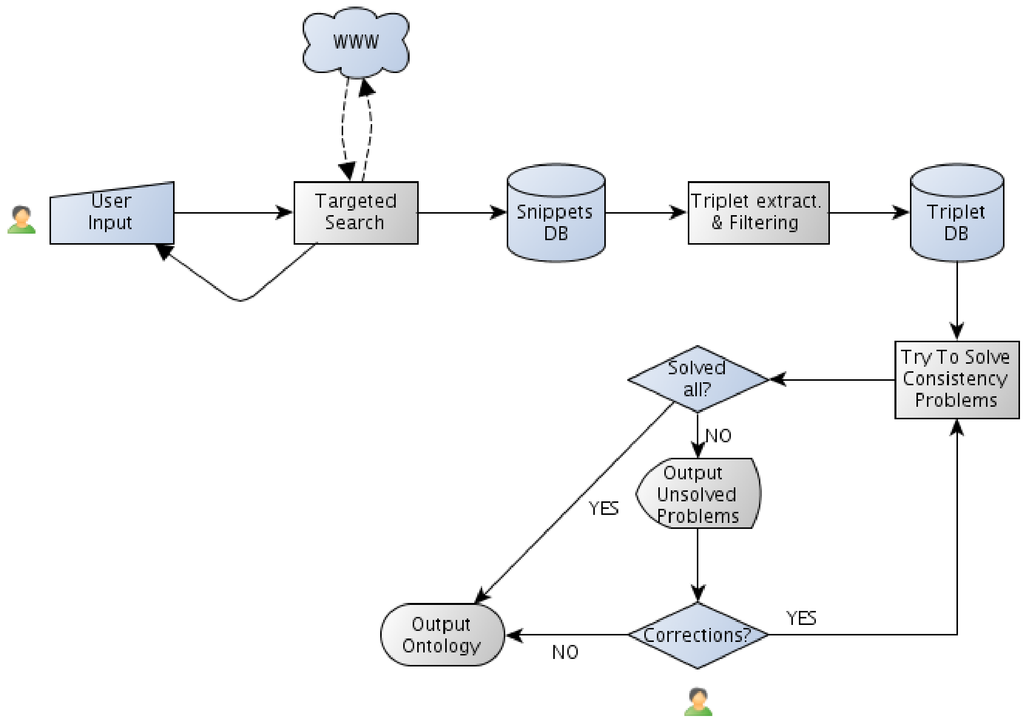


FIGURE 3.1: **Overview of our ontology generation system.** At the moment the user starts the process by specifying a company of interest and a relation type. Queries are generated using handcrafted patterns and the results are stored in a snippet DB. This process can be repeated as many times as wished and only afterwards are the triplets extracted. In order to build the ontology from all these triplets in our DB we call a conflict solving module which automatically solves up to 70% of the conflicts. It is up to the user if the unsolved conflicts get checked or not, as an ontology is output anyway and unsolved conflicting triplets are simply ignored.

of the process is given in Fig.3.1, where the person icon identifies the parts where the user interacts with the system.

3.2 Defining Relations

The relations of interest are pre-defined both as `owl:ObjectProperties` and as a set of indicator words for query building and/or relation extraction. At the moment each relation has a definition file and a blacklist file. We could easily merge these files, but we think they are more human-readable this way. As future work these properties could also be read directly from the base ontology.

At the moment a relation is defined by :

- One **explicit formulation** of the relation per row e.g. *"signed definitive agreement to acquire"* for the `acquired` relation.

- A **searchable field** consisting of a boolean value for each formulation meaning if the corresponding formulation should be used in query building or just in the lexico-semantic patterns. This is done because we observed that some formulations are not very efficient in search patterns but they still occur in snippets and we don't want to miss that information. We use at most 2 expressions for querying and keep the rest (up to 16) for pattern matching in triplet retrieval.
- the **name** of the relation, which should match the name in the base ontology
- a value specifying if the relation is the **inverse** of another one (e.g. `acquiredBy` is the inverse of `acquired`)
- a **blacklist** of terms that should not appear between the subject and relation or between the relation and the direct object. These are either terms that change the meaning of the phrase (e.g. *stake* in "Novartis acquired additional stake in Alcon" identifies a `shareholderBy` relation while `acquired` would be false) or terms that add "uncertainty" to the statement (e.g. *intends to* in "Roche Holding intends to acquire Genentech").

3.3 Collecting Information

Our system differs from most other ontology population tools because it does not require a pre-existing text input from the user, but only a “seed company”. It is a quite contradictory property: a “text mining”[2] tool which requires no text input but builds it itself. Depending on the domain or relations of interest this can be a precious asset. Let's take for example only the field of companies. General financial (mostly structured) information[3] can be found on Yahoo! Finance (among others). More particular pharmacological information[4] can be found in trivial to parse XMLs provided by DrugBank[31]. On the other hand, information about corporate structure is difficult to find. There is no central repository with such data but only (in the best case) the company has a list of acquisitions and/or subsidiaries somewhere on its Web page. Thus, providing the robust and comprehensive textual input that information extraction systems usually require is impossible in this case.

Our ground truth is built by performing targeted Google searches via the Google Custom Search API. The API returns JSON objects from which we extract some values, most notably the snippets. By the nature of this process the snippet DB is exposed to

[2]Entity and relation recognition part can be seen as text mining / information retrieval.
[3]E.g. stock value;
[4]E.g. brands and types os medicines a company produces;

"noise". Our information gathering process is not selective with regard to the source of information. Thus, we supposed that part of the information in our repository is inaccurate and therefore the "knowledge filtering", mostly carried out in the ontology building part, is crucial.

Formulating Queries

For any given instance `company` $\in labels(I_{Company})$, where $Company \in C$ (a class in our ontology) we formulate two kinds of queries:

$$[\texttt{company}] *?\ expression \tag{3.1}$$

$$expression\ [\texttt{company}] *? \tag{3.2}$$

The star (*) is a wildcard and might miss in relation searches but it is always used when searching for company suffixes (see Sec. 3.5) and *expression* is any searchable expression. In our case we have observed that queries of the form "[`company1`] *expression* [`company2`]" are too restrictive, thus not efficient.

A query example would be "operates as a subsidiary of Watson Pharmaceuticals", for the specified company "Watson Pharmaceuticals". Our system will use these kind of queries to store all available results (between 0 and 100 for one query).

3.4 Entity Recognition

Entity Recognition needs some disambiguation as it can mean different things in different projects. In our case we use it in the sense of named entity recognition and our only named entities are companies.

We define as *company names* groups of words which indicate names of organizations (any kind of company, research institute, university, profit or non-profit organization). After discussing with a domain expert[11] we decided to leave "business", "plant", "factory", "production line" etc. out of the scope of this thesis. They might be legal entities themselves (thus matching our above formulated definition of company names), but transactions of this kind are so common that their importance is minor.

In Table 3.2 we provide some examples of what we consider correct and incorrect company names, as expected from the ER tool. One can notice from the few examples provided below that we have tried to provide sufficient train data to have a flexible

True positive	**False positive**
Nestle USA Inc.	I. Nestle S.A.
Nestle Health Science S.A.	Tiger Food Brands
Nestl	Type Aktiengesellschaft
NESTL Philippines	BOOST Kid Essentials Nutritionally Complete Drink
Roche	Ortega foods business
Nestl Singapore (Pte) Ltd.	Nestle S.A.based
Chiron	Valid emails

Table 3.1: Examples of correct or incorrect annotated names, obtained with a recent model for the named entity tagger.

named entity tagger, that can interpret words with different cases or punctuation variations. We should note that false annotated names such as the ones in Table 3.2 are not systematic mistakes. For example the words *valid* and *emails* are not usually tagged as part of a company name and these mistakes occur only in some sentences (sometimes with unusual/incorrect syntax), not due to erroneous annotation in the train set. While in *Snippet1 valid emails* is tagged as a company name, in the highly similar *Snippet2* it is not.

Snippet1: "Jun 18 , 2012 .. . subsidiary of Nestl (226 executives) . Subsidiaries (1) - Executives (17) . Food (555 org charts) www.nestlehealthscience.com . Valid emails (8) .. . "

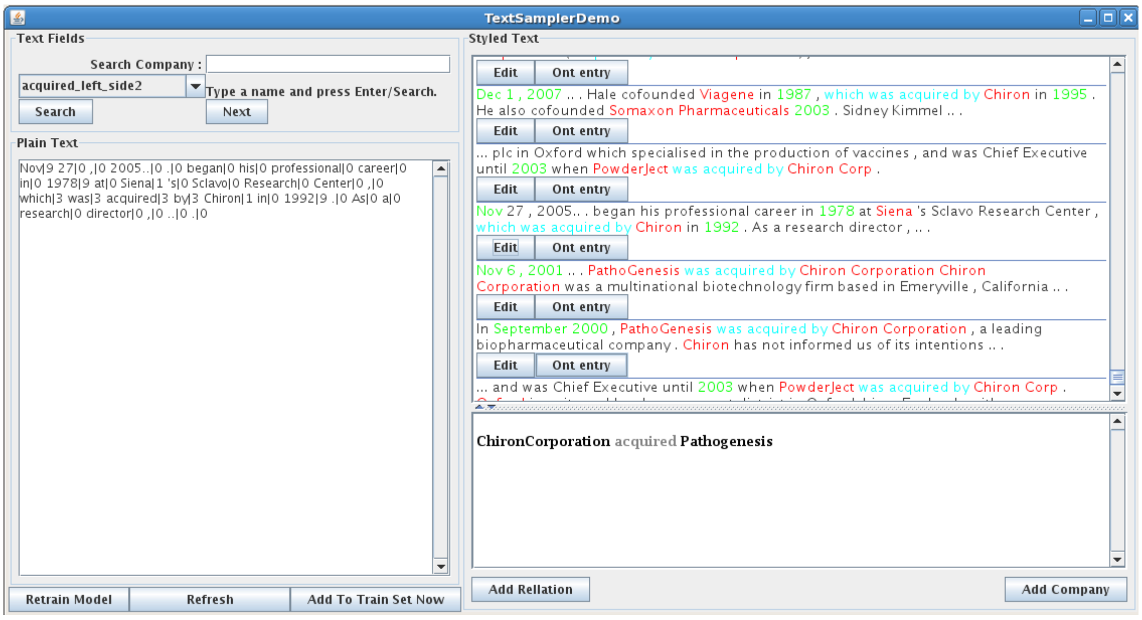
Snippet2: "subsidiary of Nestl (234 executives) . Subsidiaries (2) - Executives (19) . Beverages (204 org charts) www.nestle-waters.com . Valid emails(2) . Print List & Org .. . "

For the scope of ER we have used a **conditional random fields** (2.6.2) tool, courtesy of Katrin Tomanek [3]. I only provided the train data and tested several feature configurations.

The **train data** consists of roughly 2500 manually annotated snippets. The annotation of company names was carried out in two phases. First, 700 random snippets were annotated by hand. With this train set we already had a cross-validation F1-score of over 80%, which we considered promising enough. Using this model we annotated the continuously increasing set of snippets and we only added to the train set snippets in which companies were not correctly labeled. We have thus spared the redundant work of labelling entities with already learned patterns.

A simple tool was specifically developed to provide a graphical interface (Fig.3.2) for part of the data extraction, error tracking or model improving. The colored representation of our tool made error spotting a lot easier and train set extension faster. The few buttons in the interface spared the user from manually modifying several text files or calling shell or Java scripts. Given the high frequency of these actions, the interface helped us save valuable time.

Obtaining the current statistical model was thus a dynamic process. We considered this as a must, given that the nature of out target set could change significantly after each new company search. We are thus trying to overcome the risk of having a biased or unrepresentative model.



Nov|9 27|0 ,|0 2005..|0 .|0 began|0 his|0 professional|0 career|0 in|0 1978|9 at|0 Siena|1 's|0 Sclavo|0 Research|0 Center|0 ,|0 which|3 was|3 acquired|3 by|3 Chiron|1 in|0 1992|9 .|0 As|0 a|0 research|0 director|0 ,|0 ..|0 .|0

Dec 1 , 2007 .. . Hale cofounded Viagene in 1987 , which was acquired by Chiron in 1995 . He also cofounded Somaxon Pharmaceuticals 2003 . Sidney Kimmel .. .

... plc in Oxford which specialised in the production of vaccines , and was Chief Executive until 2003 when PowderJect was acquired by Chiron Corp .

Nov 27 , 2005.. . began his professional career in 1978 at Siena 's Sclavo Research Center , which was acquired by Chiron in 1992 . As a research director , .. .

Nov 6 , 2001 .. . PathoGenesis was acquired by Chiron Corporation Chiron Corporation was a multinational biotechnology firm based in Emeryville , California .. .

In September 2000 , PathoGenesis was acquired by Chiron Corporation , a leading biopharmaceutical company . Chiron has not informed us of its intentions .. .

... and was Chief Executive until 2003 when PowderJect was acquired by Chiron Corp .

ChironCorporation acquired **Pathogenesis**

FIGURE 3.2: Simple GUI for data extraction, error tracking and model improving.

The set of **features** was optimized on the initial set of 700 annotated snippets. In the configuration evaluated in this thesis we are using the following features[5]:

- The size of prefixes and suffixes
- A lexicon with company specific suffixes and one with company names
- The case of the words
- Word class
- Basic word class
- Orthographic features

We have tried to use part-of-speech tags but both precision and recall were 1% lower on a set of 300 random manually annotated snippets. Taking into account these results

[5]The features which are not self-explanatory are explained in Sec.2.6.2

together with the time costs, we have decided to leave them out. We suspect two reasons for this poor performance:
1) The snippets are often grammatically incorrect;
2) The heterogeneity of POS tags which are part of company names.

We only use two **labels**, as at this point our sole intention is to tag company names in unstructured text (snippets). The labels are **1** for company names and **0** for all other words. In the output file individual words are tagged instead of groups of words as we would usually wish, e.g. `Memory|1 Pharmaceuticals|1 Corp|1` instead of identifying the whole as a company name - `MemoryPharmaceuticalsCorp|1`. We build the names by simply aggregating all consecutive tokens marked with a company label. Given a snippet annotated by our NE tagger :

"Jun|9 15|9 ,|9 2011|9 ..|0 .|0 Watson|1 Pharmaceuticals|1 acquires|2 Greece|0 ś|0 Specifar|1 .|0 Watson|1 Pharmaceuticals|1 ,|1 Inc.|1 (|0 NYSE|0 :|0 WPI|0)|0 and|0 Greece|0 ś|0 Specifar|1 Pharmaceuticals|1 S.A.|1 .|0 ",

our method returns the snippet in the form below, saving the natural language name variations too ("Watson Pharmaceuticals , Inc." is added as a synonym of "WatsonPharmaceuticalsInc").

"Jun 15 , 2011 .. . WatsonPharmaceuticals#1 acquires Greece ś Specifar#1 . WatsonPharmaceuticalsInc#1 (NYSE : WPI) and Greece ś SpecifarPharmaceuticalsSa#1 ."

An obvious limitation is that consecutive names which are not separated by any special sign (e.g. comma, point, etc.) will be merged together. This is quite unlikely in normal text, but occurs in snippets. Therefore, one of the things we test in the retrieved company names is if they contain company suffixes in the middle, e.g. ThermoFisherScientific**Inc**ThermedicsInc wouldn't be added to the ontology due to the "Inc" in the middle (see Sec. 3.5).

3.5 Entity Filtering

A common problem with ML algorithms is that one can not really control the way they generalize from the learned features, which makes it difficult to assure they won't return false positives too. In our case the most common mistakes were: mistaking countries, cities or regions, eventually with a company suffix for organizations; the same with names of medicines or brands; names containing only suffixes; entities grouped together by our aggregating technique identified by a suffix in the middle; genitives, which are wrong because they clearly indicate it is not about the tagged company but about its subsidiary, unit, business, etc.

Error class	**Number of occurrences**	**Example**
Location	34	Cambridge, SunnyvaleInc
Drug	89	Prozac, Cafcit
Suffix only	101	Ag, Kgaa, LtdCo
Suffix in the middle	145	EliLillyCoLilly, AlconIncAvelingen
Genitive	177	Novartis', Crucell's

Table 3.2: **Occurrence numbers for the kinds of false positive NE we handle.** Occurrence numbers reflect how many times a triplet was rejected for having as subject or direct object a NE from the given classes. All triplets containing names from one of these classes are considered by our system invalid and are deleted from the database without further confirmation from the user.

Based on the common mistakes described above we have integrated some filters. Location and suffix-only tests have a precision of 100%. For the first four tests the overall precision was 92%, the drugs test introducing 97% of the errors. This is usually explained by the fact that smaller companies often commercialize brands with the same name as the company itself or because the synonyms in DrugBank are not perfect (we load over 27,000 synonyms). Usually these do not have a negative impact on the overall recall because if "Ici" is rejected for matching a DrugBank entry, supposing a "IciLtd" also occurs, it will be kept. For obtaining the accuracy values above we evaluated the rejected names from the first four categories; the names were considered correct if there is a company with the given name and incorrect otherwise.

The genitive test performs surprisingly well too. Only one group was incorrectly tagged as genitive (" 'Usability Wars' ") but it was a false positive anyway. The 177 genitives were not used when computing the overall precision because we need context information in order to know if a triplet candidate rejected for containing a genitive was correct or not. This is not the case for the rest of the tests i.e. we can be sure that triplets like *(Drug, acquired, Company)* or *(Company, hasSubsidiary, Location)* are not correct.

The first four tests use predefined lists while the genitive test uses only regular expressions. For the drugs list we used the names provided by DrugBank[31], for the list of locations we mixed lists of countries and largest cities from Wikipedia[6] and GeoNames[7] and the company suffix list we created ourselves (see Sec. 3.7).

[6] http://en.wikipedia.org/wiki/World%27s_largest_cities(23 August 2012), http://en.wikipedia.org/wiki/Largest_cities_of_the_European_Union_by_population_within_city_limits(8 September 2012), http://en.wikipedia.org/wiki/List_of_United_States_cities_by_population(2 September 2012)

[7] Geonames is a "geographical database covers all countries and contains over eight million location names that are available for download free of charge. " (http://www.geonames.org/)

Another condition we set is that all (company) individuals in the ontology should have a company suffix. If no suffix was found in the analysed snippets, we call a method to look for possible endings online. This test increases the precision but also lowers the recall, so the user can choose if individuals without suffixes should be exported or not. See section 3.7 for more details about the implementation and the Evaluation chapter for some statistics.

3.6 Relation Extraction

As we have stated before, our relation extraction is pattern-based and we study two approaches: matching original snippets or contexts.

We build lexico-semantic patterns :

$$\texttt{company}\ \ (OT)\{0,n_1\}\ \ \texttt{relIW}\ \ (OT)\{0,n_2\}\ \ \texttt{company}\ \ , \qquad \text{where}$$

OT = other unmarked tokens, `relIW` = relation indicator words (i.e. the expressions defined for each relation) and $X\{n,m\}$ is a quantifier meaning that X can appear at least n but not more than m times.

We want to remind the reader that at this point both companies and relation indicator words (expressions) are identified and a snippet looks like *"In the same month , US - based WatsonPharmaceuticals#1 announced_it_has_acquired#2 another company, AscentPharmahealth#1 , the Australia and South East Asia generic pharmaceutical ..."*, where *#1* is our label for companies and *#2* for a relation type. At the moment we allow at most four (n_1, n_2) tokens between the companies and the relation indicator words. These tokens are checked for blacklisted words and given at least one word is found, the triplet is rejected.

```
1: mark companies in the $inputString$
2: mark relation indicator words in the $inputString$
3: for all relIW $\in$ set of expressions for all relations do
4:    $pattern \leftarrow$ company $(OT)\{0,n_1\}$ relIW $(OT)\{0,n_2\}$ company
5:    if $inputString$ matches $pattern$ AND $\nexists t \in tokenGroup_2 \bigcup tokenGroup_2, t \in blackList$
      then
6:       return triplet.
7:    end if
8: end for
```

Algorithm 1: Method to extract triplets from text. The method receives as input either the whole snippet or on its contexts, one at the time.

Analysing our first results for regular expression based triplet retrieval we noticed that about 50% of the false positive triplets were retrieved due to the inability to correctly interpret more ”complex” formulations. Even sentences, which are easy to understand from a human reader’s point of view, were not correctly processed. Given the sentence below and the recognized companies (in bold), our pattern-based relation extractor would find the triplet `(Sanofi, acquired, NewportLaboratories)`, while after the context decomposition[8] we are able to correctly retrieve `(Merial, acquired, NewportLaboratories)`.

*“**Merial** , the Animal Health division of **Sanofi** , has acquired **NewportLaboratories** , a privately held company based in Worthington , Minnesota. ”*

The list of contexts for the above mentioned sentence is :
Context 0: Merial,, has acquired NewportLaboratories,.
Context 1: Merial,, the Animal Health division of Sanofi
Context 2: Worthington, Minnesota
Context 3: NewportLaboratories, a privately held company based in Worthington,

As one can remark the names in the context are already in camel case. We annotate companies and replace the natural language names before the context decomposition. We have come to this approach due to two reasons:
1) The parser in the context decomposition tool often got confused by the heterogeneity of POS-tags in names and the parsing quality dropped dramatically. Most often points or commas in the company names were misinterpreted and often parts of the same company name were considered to belong to different contexts.
2) Since our named entity tagger was trained on snippets, we expected it to work better on the structure of snippets than on the structure of contexts.

3.7 Identity Matching

By *identity matching*[9] we understand identification and merging of different name variations which refer to the same individual. It is a very important step because these individuals with different names are usually extracted from different sources (snippets in our case) and are often linked to complementary information. Maynard et al.(2007)[14] name it ”one of the fundamental problems” in the context of ”proliferation of information”.

We would like to prove by means of an example that correct identity matching is important in order to provide complete information. Our system retrieved the following not

[8]Context decomposition is explained in 2.6.1.
[9]In this thesis ”identity matching” and ”synonym solving” are used with the same meaning. Since the IDs of our classes/instances are actual names, the task of identity matching resumes to identifying the correct synonyms.

normalized triplets: two times (`AbbottLaboratories`, `acquired`, `KosPharmaceuticalCo`) and one time (`AbbottLabs`, `acquired`, `TapPharmaceuticals`). In the end we would of course like to see both `TapPharmaceuticals` and `KosPharmaceuticalCo` owned by `AbbotLaboratoriesInc`. Poor or no identity matching yields a new set of problems too, usually injectivity violations, e.g. (AbbottLabs, acquired, SolvayPharmaceuticals) and (AbbottLaboratories, acquired, SolvayPharmaceuticals) which, without identity/name matching, would imply that SolvayPharmaceuticals was acquired by two companies which contradicts our injectivity assumptions (see Sec. 3.8).

We distinguish two kinds of suffixes: pharmaceutical (e.g. *healthcare*, *systems*, *solutions*, *diagnostics*, *pharma*, *pharmaceuticals*, *biosciences*, etc.) or company specific (e.g. *inc*, *co*, *ltd*, *ag*, *gmbh*, etc.). The lists were built empirically and contain about 40 entries for the pharma-specific suffixes and 80 entries for the company endings.

Usually name matching is solved by considering all names with the same root synonyms (see Sec. 3.7). In our case we remove both company and pharma suffixes in order to obtain the root i.e. the root of `Abbott Laboratories Inc.` is in our system `abbott`, while in the systems presented in the related work section only company suffixes seem to be removed, i.e. the root would be `Abbott Laboratories` on the same example. We consider this kind of normalization[10] too strong. An example to support our affirmation is that ”Thermo Fisher Scientific C.V.”, ”Thermo Fisher Scientific GmbH”, ”Thermo Fisher Scientific Spa” are listed as distinct entities on the official subsidiary list of ”Thermo Fisher Scientific Inc.” [32].

In the following a pseudocode description is presented for our identity matching method (Alg. 2), which calls the following methods (Algorithms 3 and 4).

```
1: $CL \leftarrow getAllCompaniesInTriplets()$
2: $CL \leftarrow$normalize names
3: $groups \leftarrow groupComaniesByRoot(CL)$
4: for all $group \in groups$ do
5:     assign preferred terms and synonyms for $group$
6: end for
```

Algorithm 2: Method used for identity matching of company names

Whenever we test equality we use a custom function which can solve common synonyms and obviously performs case insensitive comparisons, i.e. *NovartisPharmaAg* and *novartisPharmaceuticalsAg* are found ”equal”. Throughout our processing we use company names in camel case with punctuation marks and other special characters removed because they are too often inconsistently used.

[10]such as Alcon Inc. = Alcon Ltd.

```
groups ← empty group
for all unique name ∈ CL do
  root ← replaceCompanySuffixes( name )
  root ← replacePharmaSuffixes( root )
  root ← to lower case
  if root ∉ groups then
    add root to groups
  end if
  add name to groups[root]
end for
return group
```

Algorithm 3: Method which groups names according to their root.

```
while group contains name with suffix do
  if group contains exactly one entry with suffix then
    changes ← true
    preferred ← company with suffix
    remove preferred from group
    add preferred to solved
    for all name ∈ group do
      if name is substring of preferred or one of its synonyms then
        assign name as synonym of preferred
        remove name from group
      end if
    end for
  else
    if group contains ≥ 2 entries with suffix then
      get number of hits foreach name with suffix
      preferred ← name with most hits
      remove preferred from group
      for all name ∈ group do
        if name is substring of preferred or one of its synonyms then
          assign name as synonym of preferred
          remove name from group
        end if
      end for
    else
      if group contains no entry with suffix then
        search possible suffixes online
        if found new name with suffix then
          add new name to the group (and start over)
        else
          Finish.
        end if
      end if
    end if
  end if
end while
```

Algorithm 4: Method which assigns synonyms and preferred terms to the entries in a group.

We store all query results, together with the date, so that the DB can be updated after a given time threshold.

We want to show now by means of an example how our method works. Let's take the group[11] with root `[novartis]` (Table 3.3). We see that there are several names with company suffixes so the first problem we try to solve is with which of them should `Novartis` be merged. So we proceed by searching each company name with a suffix on the Internet. We do exact[12] queries and store the number of result pages. This gives the numbers in the third column in Table 3.3, based on which we decide that "Novartis" is a synonym of "NovartisAg". We assign the rest of names in the same way and we remain with `NovartisGroup`, `NovartisMedicalNutrition`, `NovartisNutrition` which do not occur in any form with suffix. For these we need to look for suffixes online. In our case we found only one option each time (e.g. NovartisGroupLtd for NovartisGroup) so we assign the preferred term directly. In case more options exist we use the number of Google hits again. In the end we merged together the following suggested individuals, here showing the preferred term followed by the synonym(s):

NovartisMedicalNutritionSas: NovartisMedicalNutrition, NovartisMedical
NovartisAg: Novartis
NovartisInternationalAg: NovartisInternational
NovartisGroupLtd: NovartisGroup
NovartisCorporation: NovartisCo
NovartisConsumerHealthInc: NovartisConsumerHealth
NovartisPharmaceuticalsCorporation: NovartisPharma, NovartisPharmaceuticals

The remaining names in Table 3.3 were not matched to any other name.

3.8 Ontology Building

By building the ontology we understand putting together the triplets we have retrieved so far in such a way that no inconsistencies are introduced, (ideally) no false information is added and (ideally) without duplicate entries for the same individual (solved by the identity matching algorithm). We can only guarantee the first condition (that no inconsistencies are introduced), but concerning the intrinsic value of the information we can only use heuristics to lower the level of false positive entries.

[11]We want to point out that at this point a partial normalization is already done, since we use "camel case" names. For example NovartisAg alone has 10 natural language synonyms in our ontology.
[12]In Google the exact queries are identified by citation marks

Candidate name	Contains Suffix	Hits
Novartis	false	-
NovartisCo	*true*	19,399
NovartisGroup	false	-
NovartisInternational	false	-
NovartisMedicalNutrition	false	-
NovartisNutrition	false	-
NovartisPharma	false	-
NovartisPharmaceuticals	false	-
NovartisAg	*true*	2,339,999
NovartisCorporation	*true*	84,599
NovartisInternationalAg	*true*	51,599
NovartisLtd	*true*	7,179
NovartisNutritionCorporation	true	189,999
NovartisPharmaAg	*true*	166,999
NovartisPharmaGmbh	*true*	88,499
NovartisPharmaKk	*true*	115,999
NovartisPharmaLtd	*true*	153,999
NovartisPharmaSas	*true*	21,899
NovartisPharmaceuticalsCorporation	*true*	319,999
NovartisPlc	*true*	258

Table 3.3: Group with root `[novartis]`.

3.8.1 Structure

Before presenting the heuristics we developed we should explain what our ontology looks like. We focus on four types of relations between individuals of a generic class `Company`: `acquired`, `hasSubsidiary`, `holdsSharesBy` and `hasMergerPart`. All our instances have camel case IDs and synonyms in natural language.

`shareHolderBy` is a generic relation for share acquisition. We do not interpret the information ourselves - for example one might interpret the acquisition of majority stake as a subsidiary relation. In order to correctly process this kind of information one has to take into account a wide variety of facts e.g. laws differ from one country to another, larger companies might have two kinds of shares with or without decision rights, etc.. This is an interesting subject, but very complex and is out of the scope of this thesis.

`acquired` refers to acquisitions of companies only. For this thesis we restricted both the domain and codomain of the relation to instances of the class company.

`hasSubsidiary` is self explanatory. The difference between an acquired company and a subsidiary is that the latter remains "separate, distinct legal entities for the purposes of taxation, regulation, and liability"[13], while an acquired company might be merged into

[13]http://en.wikipedia.org/wiki/Subsidiary , last modified 3 September 2012.

an existing division of the parent/holding company and cease to exist as the original legal entity. It is also possible that an acquired company becomes a subsidiary.

`hasMergerPart` is a special relation because we need to take into account the outcome. A merger in the strict sense happens when "two firms agree to go forward as a single new company rather than remain separately owned and operated. This kind of action is more precisely referred to as a 'merger of equals'. The firms are often of about the same size. Both companies' stocks are surrendered and new company stock is issued in its place. For example, in the 1999 merger of Glaxo Wellcome and SmithKline Beecham, both firms ceased to exist when they merged, and a new company, GlaxoSmithKline, was created"[14]. According to Wikipedia this happens extremely rarely but often "friendly" acquisitions or acquisitions in which the two CEOs agree to join together are referred to as mergers.

The last three relations are **transitive** and **injective**. Intuitively when a company acquires another, it becomes the owner of the daughter company's subsidiaries too (transitivity) and a company cannot be a subsidiary of more than one other company at a given point (injectivity).

Let f be a function with the domain D and codomain C. We call f injective if

$$\nexists\ a,\ b\ \in\ D, a\ \neq\ b\ \ such\ that\ \ f(a) = f(b). \tag{3.3}$$

In our case both C and D are the set of instances of the class `Company` and the injective relations (f) are `subsidiaryOf`, `acquiredBy` and `isMergerPart`.

We don't have any reflexive relations so we also check for $f(a) = a$.

3.8.2 Triplet filtering

We empirically developed some measures and thresholds to remove obvious mistakes. For example a very small company is unlikely to acquire a very large company or, if a very large company acquires another very large company it is likely that there will be more information available.

There are three values we take into account: the number of hits (h), the size of the "subtree" (st) and the triplet occurrence (to). The number of hits is the number of result pages returned by Google Custom Search API; Triplet occurrence is the number of times the triplet was found (either in different snippets or the same); The size of the

[14]http://en.wikipedia.org/wiki/Mergers_and_acquisitions , Last modified 6 September 2012.

"subtree" is the number of companies which were acquired by, or are subsidiaries of the company of interest.

$$st_{parent}/st_{child} < 0.1 \tag{3.4}$$

$$h_{parent}/h_{child} < 0.02 \tag{3.5}$$

Our system retrieved the triplet (`JohnsonSonInc`, `acquired`, `BayerAg`) and $h_{JohnsonSonInc} = 35,600$, $h_{BayerAg} = 2,790,000$, $st_{JohnsonSonInc} = 1$ and $st_{BayerAg} = 40$. The triplet is incorrect and we can easily check that equation 3.4 holds while equation 3.5 doesn't , so the triplet is considered unlikely/suspicious.

Since the test described by equation 3.5 costs Google queries, the current approach is to check 3.4 first. This approach can allow more mistakes than ordering 3.5, 3.4, since our data collection is biased towards some companies. However, so far only one triplet was incorrectly rejected.

Another filter removes triplets containing individuals from one of the five error classes described in Sec. 3.4.

3.8.3 Conflict solving

The main problems we face when building the ontology are **cycles** and **injectivity violations**. Both of them make sense only in the context of `acquired`, `hasSubsidiary`, `hasMergerPart` triplets. The "shareHolderBy" relation cannot be considered in either cases because it is possible that a subsidiary has some shares in its mother company (yet not a cycle) and it is the definition of share holding that several companies or maybe persons too own different percentages of stake (yet no injectivity violation).

We call **overlapping** the triplets with the same object and different subjects. Formally this is described in Def. 3.1. These overlapping triplets are the "candidates" to injectivity violations. We make the difference between the two because overlapping triplets can be solved in many cases and do not introduce any inconsistency/violation.

Definition 3.1. We call two triplets $t_1 = (i_{1,0},\ p_1,\ i_{1,1})$ and $t_2 = (i_{2,0},\ p_2,\ i_{2,1})$ overlapping if $i_{1,0} = i_{2,0}$ and $i_{1,1} \neq i_{2,1}$, where the notations are used as in the definition of triplets (Def. 2.2).

Here is an example of overlapping triplets which are not a true injectivity violation:
(`NovartisAnimalHealth`, `acquired`, `VericoreHoldingsLimited`)
(`Novartis`, `hasSubsidiary`, `VericoreHoldingsLimited`)

(`NovartisAg`, `hasSubsidiary`, `VericoreHoldingsLimited`).
In this case `Novartis` and `NovartisAg` are synonyms and `NovartisAnimalHealth` is a subsidiary thereof. In such cases we choose the most specific triplet (see Figure 3.3).

We have observed that most of the cycles were introduced by the overlapping triplets. On a version of our system with ˜4000 triplets 286 groups of overlapping triplets were introducing about 300 cycles while without these overlapping triplets only 3 cycles were detected. Cycles are difficult to solve because we have to identify the point at which a relation is wrong, which can have a means that the search space is proportional to the length of the cycle. Triplets on the other hand are easier to identify and solve. In Alg. 5 one can notice that we focus on solving overlaps instead of cycles and we try to solve one group of overlapping triplets at once.

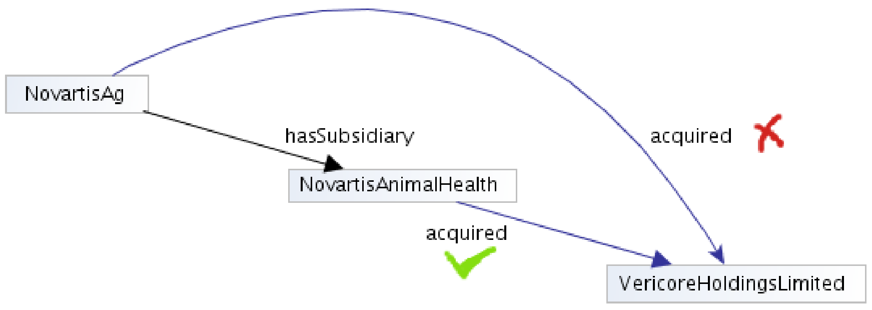


Figure 3.3: **One of our strategies to solve overlapping triplets** is to choose the most specific triplet or the "deepest" leaf if we regard acquisitions and subsidiaries as inducing some kind of hierarchy. We pictured here the case when in our ontology Novartis Animal Health is a known subsidiary of Novartis AG and we have two overlapping triplets (`NovartisAg`, `hasSubsidiary`, `VericoreHoldingsLimited`) and (`NovartisAnimalHealth`, `acquired`, `VericoreHoldingsLimited`) and we insert in the ontology only the second triplet. This is motivated by the human tendency to generalize.

In algorithm 5 we use the expression *"suspicious triplet"* which is not self explanatory. We mean the tests defined earlier in the **Triplet Filtering** subsection of this.

The ontology can be exported either as text triplet files (one file for each relation type) or in OWL. For building the OWL ontology we use the OWL API.

```
1: $ont \leftarrow$ add shareHolderBy triplets.
2: split triplet list in $simpleTripletsList$ and $overlappingGroups$
3: for all $triplet \in simpleTripletsList$ do
4:     $ont \leftarrow$ add $triplet$
5:     if $ont$ contains cycles then
6:         $ont \leftarrow$ remove $triplet$
7:         print $triplet$
8:     end if
9: end for
10: for all $group \in overlappingGroups$ do
11:     while $size(group) > 0$ AND new changes were made do
12:         if $\exists$ exactly 1 $triplet \in group$ with subject with company suffix then
13:             $ont \leftarrow$ add triplet with valid subject
14:             if $ont$ contains cycles then
15:                 $ont \leftarrow$ remove triplet
16:             else
17:                 $group \leftarrow \emptyset$
18:             end if
19:         end if
20:         $maxOcc \leftarrow$ maximum number of triplet occurrences in $group$
21:         if $\nexists\ triplet \in groups.t.maxOcc : occurrenceNo(triplet) > occTreshHold$ then
22:             $ont \leftarrow$ add triplet with maximum number occurrences
23:             if $ont$ contains cycles then
24:                 $ont \leftarrow$ remove triplet
25:             else
26:                 $group \leftarrow \emptyset$
27:             end if
28:         end if
29:         if all subjects are in a hierarchical relation then
30:             $ont \leftarrow$ add triplet with valid subject
31:             if $ont$ contains cycles then
32:                 $ont \leftarrow$ remove triplet
33:             else
34:                 $group \leftarrow \emptyset$
35:             end if
36:         end if
37:         $group \leftarrow$ delete suspicious triplets
38:     end while
39: end for
    $ont \leftarrow$ add synonyms
```

Algorithm 5: Our ontology building method. This method requires the set of normalized triplets as input.

Chapter 4

Evaluation and Discussion

In this chapter we will describe the way in which we have built our evaluation, the results we have obtained and then analyze these values.

4.1 Measures

4.1.1 Standard Accuracy Measures

Both for the scope of triplet evaluation and ontology evaluation we will use three standard accuracy measures : recall, precision and F1 score. For the triplet evaluation we will also use a self-made measure, the ”triplet-wise recall”, which we explain in section 4.1.2.

Definition 4.1. The **recall** measures the “completeness” of the positive results set and is computed using the following formula: $recall = \frac{TP}{TP + FN}$, where TP is the number of true positives and FN the number of false negatives.

Definition 4.2. The **precision** measures the quality of the returned results and is computed using the following formula: $prec = \frac{TP}{TP + FP}$, where TP is the number of true positives and FP the number of false positives.

Definition 4.3. The **F1 score** is the harmonic mean of precision and recall:
$F1 = 2 \cdot \frac{prec \cdot recall}{prsec + recall}$, where the precision and recall are defined as above.

All these measures give us information about the **correct results** in different contexts (precision in the context of all returned results or the recall in the context of the expected correct results), so we would like to explain what we understand by *correct* and *results.*

By **result** in the context of triplet evaluation we understand triplets returned by our system. They are candidates for ontology instances and their relations. In the context of ontology evaluation we consider the triplets which were inserted in the ontology as results.

We should point out that the relations we are interested in are not commutative, i.e. *"Novartis A.G. acquired Alcon Inc."* is not at all the same as *"Alcon Inc. acquired Novartis A.G."*. In this example the triplet (`NovartisAG`, `acquired`, `AlconInc`) would be correct, while a result such as (`AlconInc`, `acquired`, `NovartisAG`) would be rejected and considered a false positive.

For the scope of our triplet evaluation, we consider a triplet **correct** (i.e. positive) if we have an entry with the given triplet for the current snippet in our gold standard (see sec. 4.2) set. In other words we consider a triplet correct if the information in it can be found in the respective snippet, regardless if it is universally true or not. See section 4.2 for more information about how we built our gold standards. For the ontology evaluation we consider a result correct if the information is also available in an official subsidiary list. It is only for the `has`Subsidiary relation that we got close to a "gold standard", that is both correct and complete.

Another decision we made was to count the triplets (either right or wrong) only once per snippet. Ideally we would want to retrieve all correct triplets, even if repeating in the same snippet. We thought of several ways of building the gold standard such that repeats are scored according to their occurrences, but each had its special cases when either the false positives, true positives or false negatives could be biased. The major problem was that, when using context decomposition, results might be duplicated. This is simply due of the way in which contexts are built and parts of the original sentence might "artificially" repeat in the contexts. Thinking of our goal, we only need to retrieve each triplet at least once. Therefore we consider a reasonable approach to score each triplet once per snippet.

Below is an example where the triplets repeat identically (snippet 1) and one where a small difference makes one triplet correct and the other one false, due to the genitive (snippet 2).

BostonScientificCorp acquires RevascularTherapeuticsInc (pending) . BostonScientificCorp acquires RevascularTherapeuticsInc (pending) , Feb 15 , 2011 . . . (snippet 1)

. . . prompted the investigation came out of Warner-Lambert . When Pfizer acquired Warner-Lambert several years later , Pfizer also acquired Warner-Lambert's . (snippet 2)

4.1.2 Triplet-wise Recall

Thinking in perspective, our aim is not to have a 100% recall on triplet retrieval, but to build a correct and complete[1] ontology. It makes therefore no big difference if we retrieve a triplet once or twice, as every triplet, regardless of its number of occurrences, is a candidate for a new ontology entry. It is therefore most important that a triplet is retrieved, but not that it is retrieved from every single snippet in which it occurs. Covering every single possible formulation and exception for every single relation is naïve, so our aim is actually to define some expressions that cover most of the cases. This makes sense especially in our context where the search queries usually return several results with different formulations of the same fact.

Here is an example to show how missed triplets can be harmless. Given *Snippet1* and *Snippet2* below, our system could return the triplet (`Sanofi`, `acquired`, `Medley`), from the first snippet and (`Sanofi`, `acquired`, `HelvepharmAg`) from the second, missing the other 2 correct triplets.

Apr 1 , 2009 ... Sanofi acquires Brazilian generics firm Medley for 500mm on Strategic Transactions , Apr-01-2009 . (snippet 1)

Dec 21 , 2009 ... Sanofi has acquired generic - drug makers HelvepharmAg of Switzerland , MedleySa of Brazil and LaboratoriosKendrickSa of Mexico,... (snippet 2)

If our gold standard had only these 2 snippets our triplet-wise recall would be **2/3**, while the normal recall should be **2/4**.

Our triplet wise recall is defined just like the usual one ($\frac{true\ positives}{true\ positives\ +\ false\ negatives}$) and only the way in which we count the true positives and false negatives differs. In this case both sets contain *no duplicates* and at the set of true positives we add any triplet[2] that was retrieved from another snippet. We make here an approximate search in such a way that we make no difference between `acquired` and `hasSubsidiary` and companies might have only slight variations in their names. For example, if in our gold standard we expected to find the triplet (`BayerHealthCareAG`, `acquired`, `iSense`) and in our final triplet database our only similar entry is (`BayerHealthCareAG`, `hasSubsidiary`, `iSenseCorporation`), we will consider the desired triplet was retrieved. However, most triplets originally missing triplets are found in the exact form.

[1] While we wish to have universally correct entries, we can only speak of completeness with respect to the facts available in our snippets.

[2] Positive (expected) triplet from our gold standard.

4.2 Gold Standard

We build two standards because we want to evaluate two different things. The first thing we check is how well we manage to extract information (in the form of triplets) from the snippets. The second and most important thing to check is the quality of the ontology we are building.

4.2.1 Triplet Gold Standard

We have used 500 random snippets in our gold standard. The triplets (if any) are annotated by hand and the results are considered positive if a triplet was found and negative if no triplet information was available. We should note that many snippets do not contain enough information to extract a triplet. This can be either because queries were not efficiently build, because the searched information was found in the title of the document but in the snippet not any more or because no second company was available (which is most often the case). Evaluating 240 random snippets from our database we have found that only 46.25% contained the kind of information we were looking for. For the snippets obtained with "acquired" searches the efficiency was even lower - only 25.8% contained the wished information. The reason for this disparity is that there are many types of acquisitions (e.g. plants, legal rights or land), while the "subsidiary" concept does not have such various meanings.

We have manually extracted triplets from a set of random snippets by only looking at the context of the snippet. The information in our gold standard might be actually wrong because we do not verify the validity of the statements made in the respective snippets. For example, if information about an acquisition was clear, we added the corresponding triplet to the gold standard without checking the truth value of it. The only extra information we used was to check for names without company suffixes not to be drug or brand names.

We want to point out that, by the nature of its construction, the triplet gold standard is also the standard for entity recognition since we manually annotate the company names. Therefore we have to evaluate the reasons for false positives and false negatives.

In order to clearly describe our gold standard we show in Table 4.1 some of the special cases and our approach to solving them.

Snippet	**Triplet**	**Comment**
... investing $63 .5 million in North Vancouver CPW is investing 27 million in a new Malaysian factory Boehringer Ingelheim acquires Amgen's Fremont facility ...	-	facility acquisition, not company
Aug 22 , 2011 ... Glaxo Group (GSK) , a wholly subsidiary of GlaxoSmithKline plc , has acquired 25 .4% stake in UK-based biotechnology firm Autifony ...	(GlaxoSmithKlinePlc, hasSubsidiary, GlaxoGroup); (GlaxoGroup, holdsSharesBy, Autifony)	The second triplet might only be retrieved using contextual decomposition, but since the information is clear enough we add it to the expected results.
US - Boston Scientific Corp plans to acquire ReVascular Therapeutics Inc , a Sunnyvale-based manufacturer of medical devices . - Feb 15 , 2011 : Mergers ...	-	Since it "plans to acquire", we can not be sure that the acquisition actually happened.
SE; the ordinary shareholders received one ordinary share of Fresenius SE & Co. ... which held approximately 58% of the ordinary shares in Fresenius SE prior ...	-	"..." identify different contexts in the Google snippets, so we can not assume the subject of the first context is the subject of the second one too.

Table 4.1: Some annotation examples from our gold standard.

4.2.2 Ontology Gold Standard

If the triplet retrieval measures were relative (to the set of snippets in the gold standard), the measures obtained by ontology comparison should be absolute. We use the conditional "should" because we can not be sure that our sources are 100% accurate either. They are however the best information we could get.

We have four official lists of subsidiaries, published on the company's web pages: *Watson Pharmaceuticals Inc.* and *Thermo Fisher Scientific Inc.*, *Eurofins Scientific Group*[34] and *BRISTOL MYERS SQUIBB Co.*[35].

The subsidiary list for Watson Pharmaceuticals (as of February 14, 2012) [33] contains 140 worldwide subsidiaries while the Thermo Fisher Scientific list (as of March 23, 2011) seems to be more comprehensive, as it contains over 600 listed subsidiaries[32], thus being the most challenging case we have found.

For the ontology evaluation we have analysed a total of roughly 300 “child” companies for the 4 owning companies mentioned above. It was a laborious process as we had to consult several sources and information was not always easy to find.

We were also allowed to compare our results to a dictionary of pharmaceutical companies curated by Novartis AG [11]. It is important to mention that we can not consider this a gold standard, as we do not have any information about its absolute correctness or completeness. Thus, we will not speak about precision and recall in its the context. It is however the best collection of such information (at least from what we are aware of), so it is very interesting to compare our results to it.

4.2.3 Co-occurrence baseline

It is also important to evaluate the complexity of our triplet extraction task. For example extracting such data from good structured tables should be quite trivial; extracting information from grammatically correct and well formatted text should also be easier than from a poor-grammar text. In our case the difficulty lies of course in the nature of our snippets. Important factors could be formulations ambiguity or if they are structured in full and/or correct sentences.

The co-occurrence is a common baseline in relation extraction. It basically means taking all pairs of entities as positives[3] and compute the precision for these results, given a certain gold standard. Normally the recall would be 100%, but in our case, since we do not use already tagged entities but the (not-perfect) named entity tagger, some tuples can simply not be retrieved because the entities were not recognized. The recall obtained in this way is also very interesting because it is our algorithm’s upper limit (given the same statistical model will be used).

The **simple co-occurrence** takes all pairs of entities in a document (i.e. snippet) as positives. We can formally describe the co-occurrence tuple set for a given snippet i, as (C_i):

$$\forall\ e_k\ \neq\ e_l \in E_i,\quad C_i = C_i \cup \{(e_k,\ e_l)\}\ , \tag{4.1}$$

where E_i is the set of entities (companies) in $Snippet_i$ (without repetitions), $E_i \in I$ and the initial C_i is $\emptyset$.

The simple co-occurrence is interesting because it gives us a feeling about the nature of the information in our snippets. A high precision would indicate that the main challenge lies simply in identifying the correct relation between entities, whereas a lower precision indicates that identifying the correct entity pairs also require some attention. A high

[3]As if there were a relation between them.

precision does not indicate an easy task, because the relation and subject/object are not always trivial and the difficulty of these tasks is not taken into account in this approach.

Of course, for the simple co-occurrence evaluation we consider $(a,\ b) = (b,\ a)$. The co-occurrence is usually used merely for the existence of a relation, i.e. entity A and entity B are in a unspecified relation, which explains why we don't use triplets, but tuples of 2 instances without any relation.

We would also like a more specific baseline, which takes into account the difficulty of relation type disambiguation and subject/object identification. Thus we take all possible combinations of (ordered) triplets. In our special case we formally describe the **typed co-occurrence** triplet set (C_i) as

$$\forall\ e_k \neq e_l \in E_i,\ \forall p_x \in P,\quad C_i = C_i \cup \{(e_k,\ p_x,\ e_l),\ (e_l,\ p_x,\ e_k)\}\ , \tag{4.2}$$

where E_i is the set of entities (companies) in $Snippet_i$ (without repetitions), $E_i \in I$ and the initial C_i is $\emptyset$.

The typed co-occurrence is our actual baseline and we can only compare our results to it (not to the simple co-occurrence). We need this measure to be able to evaluate our results: these results will give us a lower limit for precision and a upper limit for recall. For example, if the baseline would yield a 75% precision, then the 85% precision achieved by our algorithm wouldn't be very impressive, as opposed to a 30% co-occurrence precision.

Since we have no symmetric relations, we set the constraint that $e_k \neq e_l$ and since we only count each triplet only once, we also remove duplicates from each snippet's set of companies. These two adjustments are also deviations from the usual co-occurrence approach and both improve the precision.

4.3 Results

Triplet Evaluation

We will start with the evaluation of triplet extraction, first when using contextual sentence decomposition (Table 4.2) and then without (Table 4.3). For the first column we evaluated three relations: `hasSubsidiary`, `acquired` and `shareHolderBy`. We do not evaluate `shareHolderBy` by itself because we do not have enough data. We used the gold standard described in sec. 4.2.1 and the triplet-wise (TW) recall was computed by re-evaluating the false negatives by hand, as described in sec. 4.1.2. The triplet-wise recall represents the percentage of positive results in our gold standard that were actually found in our triplet DB, may they come from the snippets in the GS or not. We remind the reader that all these values only give a feeling about how well we make use of the information available in the extracted snippets, without any consideration of the its intrinsic truth value.

For the triplet DB used for the TW recall, the number of distinct normalized triplets was 4012 for the simple retrieval and 3512 for contextual decomposition. This means about 12% less unique triplets obtained when using contextual decomposition.

	All	**acquired**	**hasSubsidiary**
Precision	84.97%	82.93%	87.21%
Recall	58.10%	61.26%	59.06%
F1 score	69.01%	70.47%	70.42%
TW - Recall	86.21%		

Table 4.2: **Evaluation of our triplet extraction with contextual sentence decomposition of the snippets.**

	All	**acquired**	**hasSubsidiary**
Precision	82.52%	85.39%	82.86%
Recall	67.46%	69.09%	70.16%
F1 score	74.23%	76.38%	75.98%
TW - Recall	92.96%		

Table 4.3: **Evaluation of our triplet extraction an whole snippets.**

The lower overall results in the first column are explained by the low accuracy for `holdsSharesBy` - 38.46% F1 score for whole snippets.

We now need the co-occurrence results in order to better evaluate these values.

Measure	**Simple Co-occ.**	**Typed Co-occ.**
Recall	80.55%	80.55%
Precision	31.96%	5.56%
F1	45.77%	10.40%

Table 4.4: **Accuracy measures for result sets built using the two co-occurrence approaches.** We used the same gold standard as the one for triplet evaluation and the same three relations.

In Table 4.4 we present the results for a naïve approach (see section 4.2.3) which, by definition, maximizes the recall. Since for our gold standard the recall upper limit lies at 80.55% or recall values seem a little better now (58.10% using context decomposition and 67.46% without). Strictly from the relation extraction point of view (so ignoring the entity recognition limitations) we can compare our recalls with the co-occurrence maximum. We thus see that the CD-approach retrieved **72,13%** of the relations and the simple approach has a relative recall of **83.74%**. We need this kind of comparison because, as we have stated before, our gold standard serves as evaluation standard both for entity recognition as well as for relation extraction. By making this comparison to the typed co-occurrence recall we can evaluate the recall for relation extraction *alone*.

The 31.96% precision for the simple co-occurrence (table 4.4) proves that alone the fact of choosing the correct companies (with no respect to which is the subject and which the object, nor to the relation type) is not trivial.

The difference between the results we got using contextual sentence decomposition or not confirm two things: CD improves precision but the grammar in the snippet makes text parsing tools inefficient. The explanation for the lower recall is simply the fact that the parser could not correctly interpret the tokens in the sentence, thus the contexts could not be built as we would have intuitively expected. The fact that, when adding part of speech tags as features the NE tagger model yields a 1% lower cross-validation[4] precision indicates the same problem.

Efficiency

Before we get to the evaluation of the ontology it would also be interesting to see how efficient we use the queries. We have improved our query setting strategy such that we now get 7,3 snippets/query. We remind the reader that we do exact searches so we often have no result at all and also the maximum number of snippets we might get per query is 10. So 7.3 is quite OK. In the snippets we get with our last tool version we find a new normalized triplet in every 8,69 snippets (we used the last 7148 snippets and we got 822 unique triplets).

As we have described in the previous chapter we also perform some test queries. These queries are either "normalization queries", used in synonym solving (for assigning the preferred term or for finding company suffixes) or "test queries", used when testing triplets before adding to the ontology. We need about 0,3[5] new normalization queries per triplet and for our tested set of 822 unique triplets we only needed 20 test queries.

Probably the most important fact for the reader and/or a potential user is the actual number of queries needed for an ontology entry. For our last configuration we evaluate it seems that 1.9[6] queries are on average enough for adding a triplet in the ontology.

[4]5 folds

[5]We estimate this number from the normalization queries we needed for the 822 triplets but some companies existed in previous ontology versions so the query results were saved. We have only counted the new queries.

[6]This is however not representative for the overall costs of building this ontology because at the beginning we used queries quite inefficiently.

However, this computation did not take into account the fact that some queries were already saved, thus not performed again. We can add, for the worst-case scenario 6 more queries: 2 test queries and 4 normalization queries (two for each company). This means less than 8 queries for a new normalized triplet[7].

Ontology Evaluation

For comparing our results to the subsidiaries lists we “flattened” the subtrees and evaluated each of the entries in our ontology as *correct* or *false* by checking if the given company existed on the official subsidiary list. Only case and punctuation differences were allowed (which was obvious to do since we used our camel case entity names, not the natural language synonyms). Another level was introduced and companies which were retrieved by our system with a different suffix (e.g. Biolab Ltd. instead of Biolab Pty.) were considered *“almost correct”*. In the evaluation results presented below the “almost correct” companies were considered false for the strict evaluation and correct in the approximate one.

We also point out that we cannot talk about precision and recall here, because we do not know that everything else is false.

Company	True subsidiaries strict (%)	True subsidiaries approximate (%)
Thermo Fisher Scientific Inc.	32%	45%
Watson Pharmaceuticals Inc.	74%	84%

Table 4.5: **Percent companies found following ”acquired” and ”hasSubsidiary” relations which are also listed on the latest official subsidiary list.** The percent is computed relative to the whole number of companies found in our ontology in this way.

When describing our system we mentioned that the user can decide if recognized entities for which no suffix could be found should be added or not. For the case presented in Table 4.5 particular case, allowing these entities to be added will lower the precision by a few percents (see table 4.6).

In Table 4.7 we analyse the instances which, according to our ontology, belong to one of the two companies. We have followed subsidiary, merger and acquisition relations for this evaluation and we did not allow companies found suspicious by our algorithm (e.g. because no company suffix was found). Below we describe headings of the columns:

[7]By normalized we understand that the company names are normalized; there is nothing to be normalized at the relation.

Company	True subsidiaries strict (%)	True subsidiaries approximative (%)
Thermo Fisher Scientific Inc.	27%	38%
Watson Pharmaceuticals Inc.	68%	74%

Table 4.6: **Percent companies found following "acquired" and "hasSubsidiary" relations which are also listed on the latest official subsidiary list. This time we allow entities without company suffix in our ontology.** We start from the company in the first column. The percentage is computed relative to the whole number of companies found in this way, but we cannot talk about precision here.

EM: Exact matching subsidiaries, relative to the latest official subsidiary listing. Only case or punctuation differences are allowed.
SM: Approximative matching subsidiaries where only the suffix differs (e.g. *Eurofins (Germany) AG*, *Eurofins (Germany) GmbH*). For example *HamiltonCompany* and *Fisher Hamilton L.L.C.* or *EberlineInstrumentCorporation* and *Thermo Eberline LLC* would not be considered as (SM) matching.
ASD: Acquisitions, subsidiaries or divisions which could be found on official websites and no information about a following sale or spin-out as found (although some might be dissolved). If for example a company was founded as a division of its returned parent, but later spun-off, we consider that no link was found to justify its presence in this list (NL). By official web page we understand either the subject or object company's web page, Wikipedia.org or SEC.gov.
asd: ASD but found on other "trustworthy looking" web pages (e.g. businessweek.com, secinfo.com), not blogs or tweets. **IN**: Incomplete names. They could be considered "harmless" because no real company exists with this name and are easy to identify by a human reader e.g. *Africa Inc*, *Systems Inc.* This is most often not a problem caused by the NE tagger, but by the nature of the snippets, which often contain truncated names at the beginning or end.
NE: Other named entity tagger problems, also easy to spot for the human eye. This refers to situations where extra words are erroneously tagged as part of the name, e.g. ***At****PhadiaUsInc*, *DoeIngallsInc****More***, or (not so often) distinct names which are erroneously aggregated together, e.g. *LombScientificThermoFisherScientificInc.*
NL: No relevant link could be found by the above mentioned methods (ASD or asd). To our best knowledge, these are false positive relations.

Looking at tables 4.7 and 4.8 we observe 17 out of 185, respective 2 out of 82 false positive relations. The incomplete names and the ones with other NE "problems" could be considered harmless, as in some situations a non-existing company listed as subsidiary

Company	EM	SM	ASD	asd	IN	NE	NL
Thermo Fisher Scientific Inc.	41	11	28	10	14	6	15
Watson Pharmaceuticals Inc.	45	2	6	1	0	4	2

Table 4.7: **A more detailed analisys of the same "owned" companies returned by our algorithm.**

Company	EM	SM	ASD	asd	IN	NE	NL
Bristol-Meyers Squibb	33	6	8	3	2	1	2
Eurofins Scientific	10	1	11	5	0	0	0

Table 4.8: **A more detailed analisys of the subsidiary companies in our ontology.** The sets were obtained following only `hasSubsidiary` relations.

will propagate no more false positives. This is of course up to the user and depends on the use case. In the context of our approach and the level of accuracy we can expect from it, we consider correct results the first four categories in the tables above (EM, SM, ASD, asd). The returned belonging companies for which no trustworthy link could be found are considered false positives. This, is of course, not an absolute truth value, but the best evaluation we could make. According to this (personal) interpretation of the above presented results we have obtained table 4.9.

Relations evaluated	Correct relations	Not-existing companies	False relations
`acquired`, `hasMergerPart` and `hasSubsidiary`	77.83%	12.97%	9.19%
`hasSubsidiary`	93.9%	3.66%	2.44%

Table 4.9: **Our personal interpretation of the results presented in tables 5.7 and 5.8 .**

When describing related work prior in this thesis we mentioned we would compare our results with the information on CrunchBase.com. We have taken the first 10 pharmaceutical companies listed by Wikipedia as having the highest revenues 2009-2010[8]. The subsidiaries of these companies sum to a modest total of 23, but we still found only 19 of them. One was a fresh acquisition (July 2012) and our queries were older than that and the other three were possibly not recognized as companies. On the same set of companies we have over 400 direct subsidiaries and acquired companies.

The fact that we could not retrieve all acquired companies listed on CruncBase indicates a certain limitation of our approach. One obvious limitation is the already mentioned

[8]http://en.wikipedia.org/wiki/List_of_pharmaceutical_companies, last modified on 23 September 2012 at 13:10.

Google restriction to 100 results. Besides, not all information has the same "visibility" on the Internet, nor do all companies benefit from the same level of exposure. It is likely that our approach will perform different for different companies, according to the volume of information available, as well as on its quality (i.e. how often is a company subject of rumour).

When comparing our results to the Novartis dictionary we observe the same relation between the two sets. We usually find significantly more companies, but not all the ones from the dictionary. We have randomly chosen 11 'small' companies having between 17 and 75 entries[9] each (35 on average). Leaving out the synonyms from these entries, we were left with 62 "child companies". Searching only for the roots of these companies (so 4 queries each time) we have found 86 "child companies", out of which 22 were listed in the dictionary too.

[9]We should note that the entries in this dictionary are a flat list of subsidiaries, acquired companies and name variations thereof (synonyms).

Chapter 5

Conclusion and Further Work

In this thesis, we presented an approach to find, extract and organize information from natural language texts on the World Wide Web. Such structured information can be interpreted by machines and hence be used in intelligent applications. The task of identifying instances of classes and their relations in a text corpus (information extraction) has been translated into an ontology population problem. This task is complicated by the structure of the snippets and the heterogeneity of the information in them.

The triplet wise recall of 92.9% shows that the information in snippets is accessible to extraction tools despite their structure and poor grammar. However, these same characteristics can introduce false positives and, given the (overall) unreliable nature of the information on the Internet, filtering techniques are needed. The fact that we have under 10 percent false relations in our ontology shows that even simple tests and conflict solving approaches as ours can be efficient.

As further work we think a probabilistic model could be used in the triplet extraction part. An option would be to use such a model to evaluate how "trustworthy" a snippet is e.g. from the way in which it is written, its length or (probably most important) its source (URL). The other possibility is to use a model to assign probabilities to already extracted triplets. In this case the "trustworthyness" of the snippet could be used as a feature, together with the values we also test, such as the number of snippets from which the triplet was retrieved (what we called triplet occurrence number) or the number of results returned by Google for the exact search of the name (hits number). Another obvious improvement would be to completely automatize the process.

More relations could also be used. From our experience with the kind of information already available in these snippets, locations and dates could be easily extracted.

List of Figures

List of Tables

Abbreviations

CD	**C**ontext **D**ecomposition
CI	**C**ompetitive **I**ntelligence
DB	**D**ata **B**ase
ER	**E**ntity **R**ecognition
GUI	**G**raphical **U**ser **I**nterface
MAP	**M**ean **A**verage **P**recision
MAREC	**MA**trixware **RE**search **C**ollection
ML	**M**achine **L**earning
NE	**N**amed **E**ntity
NL	**N**atural **L**anguage
NLP	**N**atural **L**anguage **P**processing
POS	**P**art **O**f **S**peech

Bibliography

[1] J. Lafferty, A. McCallum and F. Pereira. Conditional Random Fields: probabilistic Models for Segmenting and Labeling Sequence Data. *Proc. 18th International Conf. on Machine Learning*, 282–289, 2001.

[2] K. Tomanek. Resource-Aware Annotation through Active Learning. *Doctoral Dissertation, Dortmund Technical University*, 2010.

[3] R. Klinger, K.Tomanek. Classical Probabilistic Models and Conditional Random Fields.2007.

[4] M. Lupu, J. Huang, and J. Zhu. Evaluations of Chemical Information Retrieval Tools. *Current Challenges in Patent Information Retrieval, Springer*, 109–124, 2011.

[5] D. Alberts, C. B. Yang, D. Fobare-DePonio, K. Koubek, S. Robins, M. Rodgers, E. Simmons and D. DeMarco. Practical Experience and Requirements for Searching in the Patent Space. *Current Challenges in Patent Information Retrieval, Springer*, 3–43, 2011.

[6] K. H. Atkinson. Toward a More Rational Patent Search Paradigm. 2008.

[7] H. Bast, F. Baeurle, B. Buchhold and E. Hausmann. Broccoli: Semantic Full-Text Search at your Fingertips. *CoRR*,2012. URL ad.informatik.uni-freiburg.de/papers.

[8] I. Bedini, B. Nguyen. Automatic Ontology Generation: State of the Art. *University of Versailles Technical report.* 2007.

[9] T. Waechter, M. Schroeder. Semi-automated Ontology Generation Whithin OBO-Edit. *Vol. 26 ISMB 2010*88–96, 2010.

[10] E. Haussmann Contextual Sentence Decomposition with Applications to Semantic Full-Text Search. *Master Thesis, Albert-Ludwig University, Freiburg im Breisgau* URL http://ad.informatik.uni-freiburg.de/files/Contextual%20Sentence%20Decomposition.

[11] Novartis Pharma AG, NIBR Text Mining Services, Basel Switzerland.

[12] P. Cimiano, S. Staab Learning by Googling. *SIGKDD Explorations*, Volume 6, Issue 2 24–33, 2004

[13] H. Saggion, A. Funk, D. Maynard and K. Bontcheva Ontology-based information Extraction for Business Intelligence. *ISWC'07/ASWC'07 Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference* 843–856, 2007

[14] D. Maynard, H. Saggion, M. Yankova, K. Bontcheva and W. Peters Natural Language Technology for Information Integration in Business Intelligence. *Abramowicz, W. (ed.) 10th International Conference on Business Information Systems*, Poland 2007 URL http://gate.ac.uk/sale/bis07/musing-bis07-final.pdf.

[15] T.S.H. Teo and W.Y. Choo Assesing the impact of using the Internet for competitive intelligence. *Elsevier, Information and Management 39*, 67–83, 2001 URL http://www.cuaed.unam.mx/puel_cursos/cursos/d_gcfe_m_tres/modulo/modulo_3/m3-4.pdf.

[16] M. Kintz and J. Finzen A Simple Method for Mining and Visualizing Company relations Based on Web Sources. URL http://www.e-business.iao.fraunhofer.de/Images/110127_A%20Simple%20Method%20for%20Mining%20And%20Visualizing%20Company%20Relations%20Based%20on%20Web%20Sources_v1%204_tcm462-104925.pdf.

[17] M.A. Hearst Automatic Acquisition of Hyponyms from Large Text Corpora. *P*roc. of the Fourteenth International Conference on Computational Linguistics, Nantes, France 1992

[18] D. Alberts, C. B. Yang, D. Fobare-DePonio, K. Koubek, S. Robins, M. Rodgers, E. Simmons and D. DeMarco. Practical Experience and Requirements for Searching in the Patent Space. *Current Challenges in Patent Information Retrieval, Springer*, 3–43, 2011.

[19] MAREC Overview. URL http://www.ir-facility.org/prototypes/marec;jsessionid=152B56716C267268E63FEF177D96D0E4

[20] B. Settles Biomedical Named Entity Recognition Using Conditional Random Fields and Rich Feature Sets. *P*roceedings of the COLING 2004 International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA). Geneva, Switzerland. 2004.

[21] WIPO Report World Intellectual Property Indicators - Highlights. 2011. URL http://www.wipo.int/export/sites/www/ipstats/en/wipi/pdf/941_2011_highlights.pdf

[22] A. Zouak, R, Nkambou Evaluating the Generation of Domain Ontologies in the Knowledge Puzzle Project. *IEEE Transactions on Knowledge Engineering*2009

[23] W. I. Jntema, J. Sangers, F. Hogenboom, F. Frasincar A Lexico-Semantic Pattern Language for Learning Ontology Instances from Text. *Web Semantics: Science, Services and Agents on the World Wide Web, Elsevier*2012

[24] M. Balakrishna, M. Srikanth Automatic Ontology Creation from Text for National Intelligence Priorities Framework (NIPF). *Proceedings of 3rd International Ontology for the Intelligence Community (OIC) Confer- ence, pages 812, Fairfax, VA, USA* 2008

[25] D. Maynard, Y. Li, W. Peters NLP Techniques for Term Extraction and Ontology Population. *Buitelaar, P., Cimiano, P., eds.: Bridging the Gap between Text and Knowledge - Selected Contributions to Ontology Learning and Population from Text. IOS Press*2008

[26] D. Maynard, A. Funk, W. Peters SPRAT: A Tool for Automatic Semantic Pattern-based Ontology Population. *Proceedings of ICSD*2009

[27] B. Marshall, D.McDonald, H.Chen, W.Chung EBizPrort: Collecting and Analyzing Business Intelligence Information. *Journal of American Society for Information Science and technology*2004

[28] D. Celjuska, M. Vargas-Vera Ontosophie: A Semi-Automatic System for Ontology Population from Text. *Technical Report, KMI 2004*2004

[29] M. Balakrishna, D. Moldovan, M. Tatu, M. Olteanu Semi-Automatic Domain Ontology Creation from Text Resources. *Proceedings of the Seventh International Language Resources and Evaluation (LREC10)*2010 URL http://hnk.ffzg.hr/bibl/lrec2010/pdf/627_Paper.pdf

[30] G. Geleijnse, J. Korst Automatic Ontology Population by Googling. *BNAIC 2005*, p.120-126

[31] DrugBank 3.0: a comprehensive resource for 'omics' research on drugs. Knox C, Law V, Jewison T, Liu P, Ly S, Frolkis A, Pon A, Banco K, Mak C, Neveu V, Djoumbou Y, Eisner R, Guo AC, Wishart DS. Nucleic Acids Res. 2011 Jan;39(Database issue):D1035-41. PMID: 21059682

[32] Thermo Fisher Scientific Inc. Worldwide Subsidiary Listing. URL http://www.thermofisher.com/global/en/about/company/Legal_Entity_Quarterly_Update_Jan_2011.pdf 2011

[33] WATSON PHARMACEUTICALS, INC., Form 10-K, For the fiscal year ended December 31, 2011

[34] Eurofins Scientific S.E., Annual report 2011

[35] BRISTOL MYERS SQUIBB CO, 10-K Annual report. Filed on 02/17/2012.

[36] Georgios Paliouras, Constantine D. Spyropoulos and George Tsatsaronis Knowledge-Driven Multimedia Information Extraction and Ontology Evolution. *Bridging the Semantic Gap.* Springer 2011.