Thesis for the Degree of

Doctor of Philosophy

2017

Propositional Knowledge: Acquisition and Application to Syntactic and Semantic Parsing

Bernardo Cabaleiro Barciela
University Master's Degree in Languages and Computer Systems (National Distance Education University)

Doctoral Programme in Intelligent Systems

Academic advisor:
Anselmo Peñas Padilla
Associate Professor in the Languages and Computer Systems Department (National Distance Education University)

Escuela Internacional de Doctorado
EIDUNED

Thesis for the Degree of

Doctor of Philosophy

2017

Propositional Knowledge: Acquisition and Application to Syntactic and Semantic Parsing

Bernardo Cabaleiro Barciela
University Master's Degree in Languages and Computer Systems (National Distance Education University)

Doctoral Programme in Intelligent Systems

Academic advisor:
Anselmo Peñas Padilla
Associate Professor in the Languages and Computer Systems Department (National Distance Education University)

A mi familia

Acknowledgements

Esta tesis pone punto final a mis estudios de doctorado, y, a la vez, supone el fin de una bonita etapa de mi vida. A lo largo de este tiempo me han acompañado muchas personas que han hecho que esta experiencia sea inolvidable. Estas líneas son para intentar agradecerles todo lo que han hecho por mí.

En primer lugar, me gustaría mostrar mi agradecimiento a Anselmo Peñas por su dirección tanto del trabajo de fin de máster como de esta tesis doctoral. Durante todo este tiempo Anselmo ha puesto mucho empeño para formarme, no sólo en el campo del PLN sino también como investigador. Quiero agradecerle la libertad que me ha dado para trabajar en lo que me interesaba, y el especial cuidado que ha puesto en corregir y mejorar esos trabajos.

Me gustaría dar las gracias también a mis compañeros en el departamento de lenguajes y sistemas, tanto a estuvieron como a los que están. Con ellos he aprendido mucho de procesamiento de lenguaje, de aprendizaje automático y de ciencia en general, pero lo más importante es que su amistad me ha ayudado a disfrutar (y soportar) el periodo del doctorado. Doy las gracias a Guille, Rubén, Damiano, Álvaro, Emilio, David, Arkaitz, Gara, Victor, y Miguel Ángel y al resto de compañeros que han pasado por LSI estos años. Quiero agradecer también a los que se han dedicado a perder sus días en el Lizarrán arreglando el mundo conmigo, a Ángel, Agustín, Andrés y Javi. También quiero darles las gracias a Lourdes y a Conchi, que me han alegrado y dado ánimos todas las tardes sin falta.

I am also thankful to my hosts during my two stays at the University of Edinburgh, Mirella Lapata, Oier López de Lacalle and Kristian Woodsend. They were really supportive and their ideas were really inspiring for my research work. In addition, I would like to thank to the members of the ILCC group for being really friendly with me, and help me through those days.

I am equally thankful to my supervisor at the University of York, Suresh Manandhar, for his implication and kindness during my stay there and even later. I highly doubt that there are many advisors that are willing to spend that much time and effort with a visiting student. I also want to thank Burcu Can, Alexandros Komminos and Nils Monning for their help there and for sharing some really good moments.

Además de los compañeros de trabajo, me gustaría darles las gracias a mis amigos, tanto los de Madrid, celtillitas y compañía, como los de Vigo, que a pesar de la distancia siempre han estado cerca. Y a Antón, que ha sido un gran compañero de piso, sobre todo cuando tuvo que ejercer de ambulancia. A todos vosotros, muchas gracias.

Por último, quiero dar las gracias a mi familia. A mis abuelos, que me acompañarán siempre. A mis abuelas, que son un ejemplo de fortaleza y ternura. Por supuesto a mis padres, que me han apoyado incondicionalmente y sin los que terminar el doctorado hubiese sido literalmente imposible. Y a mis hermanos, tios, primos y sobrinos. No puedo estar más orgulloso de todos vosotros.

Abstract

Propositional Knowledge: Acquisition and Application to Syntactic and Semantic Parsing

Interpretation of natural language is one of the central challenges for the development of an artificial intelligence. In general, interpretation requires to build a context of entailed implicit information (from hearer and speaker background knowledge) that permits to recover the original communicative intention. Natural language processing tasks are concrete realizations of our human ability to comprehend and use language, thus in the long term we will need to provide full interpretation capabilities to machines, starting with the development of methods to acquire and use background knowledge.

We explore the use of propositions as background knowledge and its utility for language interpretation. Propositions encode knowledge in the form of assertions using natural language, and provide a straightforward way of expressing knowledge without domain restrictions. Propositional knowledge can be derived directly from meaning representations that, in turn, can be obtained directly from text, and therefore, knowledge and representations can be easily connected to perform the textual inferences required for language interpretation.

In this thesis, we propose the automatic acquisition of propositional knowledge from large corpora whose documents are represented as graphs. The frequencies of occurrence permit to express a sense of plausibility. The resulting proposition store supposes a middle ground between meaning representations and structured knowledge bases. This opens new research lines that we address in this work. One the one hand, the connection of the meaning representation with the proposition stores so that they can play the role of the background knowledge that enables an inference. On the other hand, the mapping between proposition stores and structured knowledge bases. We explore these research lines with two specific tasks related to natural language understanding: syntactic and semantic parsing.

Specifically for syntactic parsing, we address the problem of appositive correction. Appositives are grammatical dependencies that are often used to express that an instance belongs to a semantic class. We use propositional knowledge to measure the semantic compatibility between entities and entity types with semantic classes. Then we use this information to disambiguate cases where there are several grammatical valid candidates to govern an apposition.

Regarding semantic parsing, we build a lexicon that permits to map natural language utterances in the form of propositions with linked data relations, and show how to use this resource in a question answering system. In addition, we propose a method to evaluate grounding and the effect that the lexicon has in the task, independently from the processes of training or querying.

Using propositional knowledge for textual inference represents a new paradigm for language interpretation. The goal is to validate this paradigm and to explore from it the main areas involved: meaning representation, knowledge acquisition and textual inferences. Results show that proposition stores are a general purpose resource that permit to address different tasks related to language interpretation, opening new and promising research avenues.

Keywords: Propositional Knowledge, Language Interpretation, Syntactic Parsing, Semantic Parsing

Resumen (Spanish Abstract)

Conocimiento Proposicional: Adquisición y Aplicación en Análisis Sintáctico y Semántico.

La interpretación de lenguaje natural es uno de los retos centrales en el desarrollo de la inteligencia artificial. En general, la interpretación requiere construir un contexto de información implícita e implicada (a partir del conocimiento antecedente del emisor y el oyente) que permite recuperar la intención comunicativa del emisor. Las tareas de procesamiento de lenguaje natural son realizaciones concretas de nuestra habilidad humana para comprender y usar lenguaje, y por lo tanto a largo plazo tendremos que proveer capacidades completas de interpretación a las máquinas, empezando con el desarrollo de métodos para adquirir y usar conocimiento antecedente.

En concreto, exploramos el uso de proposiciones como conocimiento antecedente y su utilidad para la interpretación del lenguaje. Las proposiciones codifican conocimiento en forma de aserciones utilizando lenguaje natural, y proporcionan una manera directa de expresar conocimiento sin restricciones de dominio. El conocimiento proposicional puede ser derivado directamente desde representaciones semánticas del texto, que a su vez pueden ser obtenidas automáticamente a partir de texto. De esta manera, se abre la posibilidad de conectar de manera directa la representación de un texto con el conocimiento necesario para su interpretación.

En esta tesis, proponemos la adquisición automática de conocimiento proposicional desde grandes corpus cuyos documentos son representados como grafos. Las frecuencias de ocurrencia permiten expresar un sentido de plausibilidad. El almacén de proposiciones resultante supone un término medio entre representaciones semánticas y bases de conocimiento estructuradas. Esto abre nuevas líneas de investigación que abordamos en este trabajo. Por una parte, la conexión entre la representación semántica y los almacenes de proposiciones de manera que estos jueguen el papel del conocimiento antecedente que habilita una inferencia. Por otra parte, la correspondencia (mapping) entre los almacenes de proposiciones y las bases de conocimiento estructurado. Exploramos estas líneas de investigación con sendas tareas específicas relacionadas con la comprensión del lenguaje: el análisis sintáctico y el análisis semántico.

Específicamente para análisis sintáctico, abordamos el problema de la corrección de aposiciones. Las aposiciones son estructuras gramaticales que se usan frecuentemente para expresar que una instancia pertenece a una clase semántica. Usamos conocimiento proposicional para medir la compatibilidad semántica entre entidades y tipos de entidad con clases semánticas. Posteriormente empleamos esta información para desambiguar casos donde hay varios candidatos gramaticalmente válidos para gobernar una aposición.

Respecto al análisis semántico, construimos un diccionario que permite enlazar expresiones en lenguaje natural en forma de proposiciones con relaciones de una base de datos estructurada, y mostramos cómo utilizar este recurso en un sistema de búsqueda de respuestas (Question Answering). Adicionalmente, proponemos un método para evaluar el proceso de enlazado (grounding) y el efecto en la tarea que tiene el diccionario obtenido, independientemente del proceso de entrenamiento y cableado de consultas.

El uso de conocimiento proposicional en inferencias textuales representa un nuevo paradigma para la interpretación del lenguaje. El objetivo es validar este paradigma y explorar desde él las principales áreas involucradas: representación semántica de los textos, adquisición de conocimiento antecedente y habilitación de inferencias textuales. Los resultados obtenidos muestran que los almacenes de proposiciones son un recurso general que permite abordar tareas muy dispares relacionadas con la interpretación del lenguaje, abriendo así nuevas y prometedoras líneas de investigación.

Palabras Clave: Conocimiento Proposicional, Interpretación del Lenguaje, Análisis sintáctico, Análisis Semántico

Resumo (Galician Abstract)

Coñecemento Proposicional: Adquisición e Aplicación en Análise Sintáctica e Semántica.

A interpretación da linguaxe natural é un dos retos centrais no desenvolvemento da intelixencia artificial. En xeral, a interpretación require construír un contexto de información implícita e implicada (a partir do coñecemento antecedente do emisor e o oínte) que permite recuperar a intención comunicativa do emisor. As tarefas de procesamento da linguaxe natural son realizacións concretas da nosa habilidade humana para comprender e usar linguaxe, e polo tanto a longo prazo teremos que prover capacidades completas de interpretación ás máquinas, comezando co desenvolvemento de métodos para adquirir e usar coñecemento antecedente.

En concreto, exploramos o uso de proposicións como coñecemento antecedente e a súa utilidade para a interpretación da linguaxe. As proposicións codifican coñecemento en forma de asercións empregando linguaxe natural, e proporcionan unha maneira directa de expresar coñecemento sen restricións de dominio. O coñecemento proposicional pode ser derivado directamente dende representacións semánticas do texto, que á súa vez poden ser obtidas automaticamente a partir de texto. Desta maneira, ábrese a posibilidade de conectar de maneira directa a representación dun texto co coñecemento necesario para a súa interpretación.

Nesta tese, propoñemos a adquisición automática de coñecemento proposicional dende grandes corpus cuxos documentos son representados como grafos. As frecuencias de ocorrencia permiten expresar un sentido de plausibilidade. O almacén de proposicións resultante supón un termo medio entre representacións semánticas de texto e bases de coñecemento estruturadas. Isto abre novas liñas de investigación que abordamos neste traballo. Por unha parte, a conexión entre a representación semántica dos textos e os almacéns de proposicións de maneira que estes xoguen o papel de coñecemento antecedente que habilita unha inferencia. Por outra parte, a correspondencia (mapping) entre os almacéns de proposicións e as bases de coñecemento estruturado. Exploramos estas liñas de investigación con sendas tarefas específicas relacionadas coa comprensión da linguaxe: a análise sintáctica e a análise semántica.

Especificamente para análise sintáctica, abordamos o problema da corrección de aposicións. As aposicións son estruturas gramaticais que se empregan frecuentemente para expresar que unha instancia pertence a unha clase semántica. Empregamos coñecemento proposicional para medir a compatibilidade semántica entre entidades e tipos de entidade con clases semánticas. Posteriormente empregamos esta información para desambiguar casos onde hai varios candidatos gramaticalmente válidos para gobernar unha aposición.

Respecto á análise semántica, construímos un dicionario que permite enlazar expresións en linguaxe natural en forma de proposicións con relacións dunha base de datos estruturada, e mostramos como empregar este recurso nun sistema de procura de respostas (Question Answering). Adicionalmente, propomos un método para avaliar o proceso de enlazado (grounding) e o efecto na tarefa que ten o dicionario obtido, illadamente do propio proceso de adestramento e cableado de consultas.

O uso de coñecemento proposicional en inferencias textuais representa un novo paradigma para a interpretación da languaxe. O obxectivo é validar este paradigma e explorar a partir del as principais áreas involucradas: representación semántica de textos, adquisición de coñecemento antecedente e habilitación de inferencias textuais. Os resultados obtidos mostran que os almacéns de proposicións son un recurso xeral que permite abordar tarefas moi dispares relacionadas coa interpretación da linguaxe, abrindo así novas e prometedoras liñas de investigación.

Palabras Clave: Coñecemento Proposicional, Interpretación da linguaxe, Análise Sintáctico, Análise Semántico

Contents

List of Figures

List of Tables

1

Introduction

The problem of intelligence seemed hopelessly profound. I can't remember considering anything else worth doing.

Marvin Minsky

Contents

1.1 Motivation

Language is basic for human cognitive abilities, and, in fact, sometimes intelligence has been reduced to the ability to comprehend and create language. Such vision was popularized by the Turing Test (Turing, 1950), which states that a machine is intelligent when its answers to a human interrogator cannot be distinguished from human responses. One crucial capability of such a machine is to perform **language interpretation**.

Early work on Natural Language Processing (NLP) (Schank and Abelson, 1977; Hobbs et al., 1988) was really ambitious and aspired to replicate human ability to interpret language, a task often known as Natural Language Understanding (NLU). At that point, the goal was unreachable, partially due to the limitations on computation and data. Then, the focus shifted to practical, simplified problems where *ad hoc* solutions, typically statistic based approaches, are able to obtain satisfactory results, even sometimes to the point to consider these problems as solved. Still, these techniques are far from full language interpretation. We follow the idea that NLP techniques are now mature to retake the original goal of NLU (Etzioni et al., 2006; Manning, 2016), since it is the most promising option to finally cope with complex language problems.

In order to explain what we understand by language interpretation in practice, let's consider a simple example of a pair of sentences extracted from the Winogard Schemas (Levesque et al., 2011):

(1.1) The city council refused the demonstrators a permit because *they* feared violence.

(1.2) The city council refused the demonstrators a permit because *they* advocated violence.

Sentences 1.1 and 1.2 are ambiguous since there is a case of pronominal anaphora where the pronoun *they* could refer either to *the city council* or to *the demonstrators*. Although both sentences could be syntactically equivalent, in the sense that they can be described with the same syntactic structure, typically a human would assign the pronoun *they* to a different entity on each case. Due to its past experiences, a human reckons as more plausible that *city councils fear violence* and that *demonstrators advocate violence* rather than their counterparts. Thus, the interpretation of the sentence requires to resolve the ambiguity using our previous knowledge of the world.

Again, consider the following pair of sentences:

(1.3) David supports the team of his *wife*, Julia.

(1.4) David supports the *team* of his wife, The Vikings.

Both sentences 1.3 and 1.4 have identical surface structure. However, determining the syntactic structure requires to resolve the coreferences between the two parts of the sentences. As in the previous example, people use their previous knowledge to realize that most probably *Julia* is best identified as a *wife*, whereas *The Vikings* are described as *team*.

Consider a third example:

(1.5) Stolen Van Goghs back on display after years in criminal underworld.

Sentence 1.5 has the same lecture for most of the readers. However, achieving the proper interpretation requires to resolve the metonymy where the name *Van Goghs* is actually meaning *Van Gogh's paintings*, although other alternatives such as *Van Gogh's sculptures* are possible. Unlike the previous example, the disambiguation does not require to choose between two options, but to retrieve a missing relevant piece of information that is not stated in the text.

In every case, the interpretation is based on knowledge about the target of the interpretation. For instance, Table 1.1 shows the kind of knowledge that would be useful for each example. An utterance is judged with a probability, which is a measure of its plausibility, that is, how well that option fits with our previous beliefs. In this work, we refer to this kind of knowledge as **background knowledge**. For our purpose, background knowledge is open-domain, general-purpose knowledge expressed in a machine readable format. In practice, systems depend on the availability and precision of the background knowledge to interpret language.

Utterance	Probability
city council fear violence	↑
city council advocate violence	↓
demonstrators fear violence	↓
demonstrators advocate violence	↑
Julia is a wife	↑
Julia is a team	↓
The Vikings are a wife	↓
The Vikings are a team	↑
Van Gogh has paintings	↑
Van Gogh has sculptures	↓

Table 1.1: Background Knowledge Base. An utterance is associated with a probability as a measure of its plausibility.

In our view, language interpretation requires to contextualize a meaning representation with respect to its relevant background knowledge. As in the examples, a background knowledge can be used for instance to choose a plausible interpretation in an ambiguous sentence or to recover a missing piece of information.

The first challenge that we tackle is how to represent knowledge so it can be connected with language. In our previous examples, we show how assertions with an assessment of confidence can be used to interpret sentences. We use the term **propositional knowledge** to refer to this knowledge in the form of assertions (propositions).

Early approaches tried to encode knowledge into a static ontology or any other kind of structured knowledge. However a large portion of knowledge cannot be represented with a predefined logical model or ontology, and therefore its use for general purpose language interpretation is limited. Alternatively, propositions permit to encode any knowledge since they use natural language as vocabulary.

In addition, propositions can be acquired directly from text for three main reasons. Firstly, there are plenty of available textual data, specially on the Web. Secondly, tools and methods for NLP, and specifically syntactic parsers, are more refined, mature and efficient, and finally, the computational power is bigger and cheaper. Our aim then is to automatically derive propositional knowledge from large textual resources so that the frequencies of occurrence in text can give a sense of plausibility.

Our second challenge is to structure language so it can be connected to knowledge. We need to define a structure that can be handled by computers and that can represent information conveyed by text. This structure is usually denominated semantic representation or meaning representation (Schubert, 2015; Oepen et al., 2014). We use the latter term in this thesis. In numerically specifiable and finite state problems, such as chess, the necessary knowledge can be easily encoded in a machine in a set of formal rules. In contrast, natural language is not finite, and therefore it is impossible to describe all the domain knowledge with rules. In this dissertation we propose and explore a **graph-based representation** derived directly from linguistic processing tools.

Our third challenge is to define a way to connect language and knowledge to enable the **textual inferences** required for language interpretation. A mayor difficulty that limits language interpretation come from the semantic gap between meaning representations and background knowledge. However, these research areas are often tackled in isolation, and the synergies are lost. We believe that this distance must be minimized in order to be able to mutually improve each other.

The graph-based representation opens the possibility of directly extract propositional knowledge, reducing the gap between meaning representations and background knowledge. The result is that meaning representations and background knowledge share the same vocabulary, and therefore the connection to perform textual inferences is enabled. We address two specific tasks that require textual inferences: syntactic and semantic parsing. We choose these tasks because, although they are typically tackled using statistical approaches, they require some sort of understanding to be properly solved.

Summing up, this thesis studies **a new paradigm that uses propositional knowledge as a middle ground between meaning representations and structured knowledge**. Figure 1.1 compares the traditional approach to acquire and link knowledge with text and our approach based on propositional knowledge stores. In previous approaches, the interaction between documents and knowledge bases were limited to be used in one direction to populate the knowledge base and in the opposite direction to tag specific concepts in the text with an entity linking process. Our proposal is to use propositional knowledge to reduce the gap between meaning representations and knowledge bases, facilitating the interaction between each other.

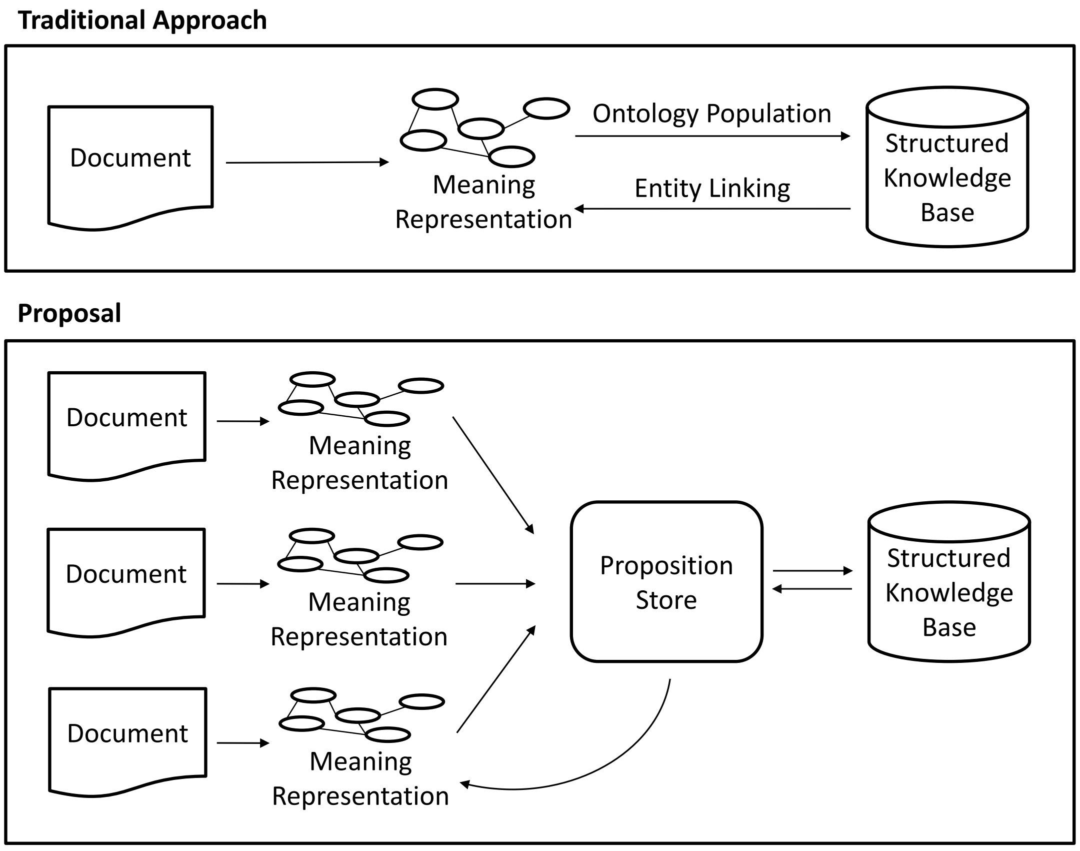


Figure 1.1: Comparison between traditional approaches and our proposal. Our proposal depicts the intended relation between proposition stores, meaning representations and structured knowledge bases.

In this thesis we validate this paradigm by showing how to acquire propositional knowledge and exploring its connection with meaning representations and structured knowledge bases. We explore this connection in two tasks related to syntactic and semantic parsing as two samples of the new research opportunities that this paradigm opens. In the first case, propositional knowledge is used to drive the disambiguation of the syntactic structures that involve appositions. In the second case, we explore the mapping from propositions to linked data properties in the context of Question Answering.

1.2 General Hypotheses

In the previous section we have motivated the area of propositional knowledge acquisition, and the general utility towards NLU. In this section we define the scope of the thesis by considering three main work hypotheses:

Research Hypothesis 1: *Propositional knowledge can be used to represent and store background knowledge.*
Propositions encode predicate-argument structures that can capture open-domain knowledge. Propositions are manipulable by computers, and enable general purpose inferences. Gather a large number of propositions through the representation of large collections of text yields a background knowledge base that mimics common sense.

Obviously, to improve the background knowledge base would facilitate the inference process. The problem is that it is not trivial to measure how good a knowledge base is. For a specific purpose knowledge base, the default evaluation consist in to measure the performance in the specific task for what it was designed. As this is an extrinsic evaluation, it is not completely independent, as it relies on the ability of the system to harness the knowledge base for the task. For general purpose knowledge bases the evaluation is even harder, because the type and the number of tasks for evaluation is not defined. Even if an independent evaluation is unfeasible, it is still desirable to improve the background knowledge base through an adequate acquisition phase. This is the base for the next hypothesis.

Research Hypothesis 2: *A graph based representation is useful to encode documents for proposition extraction.*
A prerequisite to acquire propositional knowledge is to encode text in a meaning representation. Even if an intrinsically evaluation of the knowledge base is infeasible, we define some properties for the meaning representation that reasonably can improve the acquisition of propositions.

These properties are: (1) Capture the relations present on the original text, including long distance relations; (2) relations can be morphosintactic, semantic or temporal; and (3) favour that sentences with identical meaning have the same representation, disregarding syntactic realizations. Graph are a flexible meaning representation close to language that can encode open domain text. Compared with other representations, graphs can produce sophisticated features, such as clustering different mentions of named entities and connecting long-distance relations. Moreover, graphs can be enriched with semantic information in order to acquire novel propositions. Summing up, graphs enable to acquire more propositional knowledge while at the same time reducing the sparsity.

Research Hypothesis 3: *Propositional knowledge can serve as basis to perform textual inference.*
A proposition store can be used as a background knowledge base that enable textual inferences related to language interpretation tasks. Previously we have discussed that the utility of a knowledge base is dependent on the task where it is applied. In this case, we aim to adapt a general purpose knowledge base to two different tasks.

Specifically, syntactic and semantic parsing are knowledge-dependent tasks that require to interpret language to be solved and can benefit of propositional knowledge. Regarding syntactic parsing, we develop a method to improve the apposition dependence detection in cases where there are several grammatically correct candidates to act as governor. We consider the semantic compatibility of both parts of the apposition using propositional knowledge as evidence to show its potential to interpret language.

Then, we study how to ground propositional knowledge into relations and types of a linked data database with a twofold goal. First, we provide a method to build a lexicon that maps natural language into logic forms to be used to feed a question answering system. And second, we enable a standalone evaluation of such lexicon.

1.3 Objectives

In previous sections we have discussed the context of this research project and stated the main hypotheses that drive our research. In this section we state our specific objectives to check whether the hypotheses hold.

The main goal of this thesis is to study the utility of background knowledge in the context of language interpretation. Specifically, **we aim to explore the feasibility and limitations of automatic propositional knowledge acquisition and its use for textual inference**. We develop an end-to-end approach that covers knowledge representation, knowledge acquisition and inference, and studies the relations among these steps.

We pursue this research goal through the following set of specific objectives:

Objective 1: Build a conceptual model of the meaning of a sentence though deep processing of text. Represent documents in a meaning representation that expresses the information contained on plain text. In particular, explore a graph-based representation that is capable of representing documents at document level and includes syntactic, semantic and temporal relations.

- ***Objective 1.1:*** Design and implement a methodology to represent documents as semantically enriched morphosyntactic graphs.
- ***Objective 1.2:*** Build a collection of document represented as graphs as a resource for feature extraction and proposition extraction.
- ***Objective 1.3:*** Study the generation of new features.
- ***Objective 1.4:*** Extrinsic evaluation in relation extraction.

Objective 2: Automatically acquire knowledge from the graph-based representation. Extract knowledge into propositions, which are predicate-argument structures that express open relations.

- ***Objective 2.1:*** Design and implement a methodology to automatically extract propositions from the graph-based representation.
- ***Objective 2.2:*** Build a proposition store from the collection of graphs.
- ***Objective 2.3:*** Demonstrate the application to named entity disambiguation.

There are many open research questions about the use of propositional knowledge and the inferences that can be performed with it. The following objectives correspond to the tasks that we tackle.

Objective 3: Improve apposition dependency detection using knowledge about semantic classes. Measure the semantic compatibility between the two sides of an apposition in order to decide the most suitable governor.

- ***Objective 3.1:*** Define the problem of apposition correction and classify the errors that parsers commit.
- ***Objective 3.2:*** Build a gold standard with conflicting cases of apposition detection and manually annotate the correct relation.
- ***Objective 3.3:*** Design and implement a method for apposition correction based on background knowledge.
- ***Objective 3.4:*** Evaluate the apposition correction method in the gold standard and compare to parsers in the state of the art.

Objective 4: Analyse the impact of the mapping from utterances into semantic relations for semantic parsing over linked data. Align propositions to linked data properties to feed a semantic parser.

- ***Objective 4.1:*** Design and implement a methodology to map from propositions into linked data properties.

- ***Objective 4.2:*** Study the contribution of knowledge acquisition on closing the gap between natural language and linked data relations. Isolate the evaluation of the knowledge acquisition from other typical steps in semantic parsing such as training and querying. Evaluate which syntactic structures should be aligned and what is the impact of each one.
- ***Objective 4.3:*** Study whether external linguistic resources are useful to reduce the lexical gap in the context of mapping propositions to linked data relations.

1.4 Research Questions

We enunciate a set of research questions for each objective. The resolution of the objectives provides evidence to response the questions.

- **Graph-based Representation.**

  Our first objective is to create a compact representation at document level that aggregates morphosyntactic and semantic information. Regarding this representation:

  - **Research Question 1.1:** *What are the steps to build a graph-based representation from text at the whole document level using off-the-shelf tools?*
  - **Research Question 1.2:** *What kind of lexical, syntactic and semantic relations can be expressed by these graphs?*
  - **Research Question 1.3:** *What other features do the representation provide?*
  - **Research Question 1.4:** *What features of a information extraction classifier are affected by the graph representation?*
  - **Research Question 1.5:** *Which is the performance of a classifier trained with those features in a task of automatic relation extraction?*
  - **Research Question 1.6:** *In the same task, once the graphs are generated, what is the effect of the new semantic information?*

- **Propositional Knowledge Acquisition.**

  Our second objective is to automatically acquire a propositional knowledge base derived automatically from the graphical representation of documents.

  - **Research Question 2.1:** *Is it feasible to build knowledge bases from documents represented as graphs? What does this kind of representations provide?*

- **Research Question 2.2:** *What problems arise from an automatically generation of propositional knowledge bases?*

- **Improving Parsing with Automatically Acquired Semantic Classes.** Our next objective is to use the extracted knowledge to perform textual inference, specifically to improve parsing in appositive dependencies. We tackle the problem of grammatical ambiguity when there are several candidates to govern an apposition with a named entity as a dependent. We assume that the common noun with higher semantic compatibility with the named entity is the best choice to govern the apposition. In this context, we formulate the next research questions:

  - **Research Question 3.1:** *What kind of errors do parsers commit on appositions because of the missing semantic information?*
  - **Research Question 3.2:** *Is it possible to overcome these errors considering information captured previously from text collections? What evidence can it provide to characterize the named entity?*
  - **Research Question 3.3:** *What is the most effective way to measure the semantic compatibility between the candidate classes and the named entity?*
  - **Research Question 3.4:** *What configuration of evidence and measures achieves the best results?*

- **Grounding Proposition Stores for Question Answering over Linked Data.** Question Answering, and specifically semantic parsing, typically requires a lexicon that maps natural language into semantic representations such as the entities and properties of linked data. The goal is to study whether a Grounded Proposition Store (GPS) built by aligning propositions with linked data properties can contribute to close the lexical gap between natural language and linked data labels.

  In the context of a Semantic Parser trained using raw text for distant supervision:

  - **Research Question 4.1:** *What are the methodological steps to build a GPS?*
  - **Research Question 4.2:** *What is the impact of the GPS when used to feed a semantic parser for question answering?*
  - **Research Question 4.3:** *What linguistic phenomena (syntactic-semantic relations) should be considered in the knowledge acquisition step?*
  - **Research Question 4.4:** *Are external linguistic resources useful for enriching the GPS?*

1.5 Research Methodology

This section states the research methodology we followed in order to achieve our research goals. It is composed of eleven principal steps:

1. Review of the state of the art: We review the literature in order to highlight the research opportunities in the context of knowledge acquisition and inference. We study the partial problems involved, and how they are related.

2. Develop a graph based representation using deep language processing techniques such as POS tagging, coreference resolution, dependency parsing and temporal analysis.

    - Cluster different mentions of a named entity across a document.
    - Normalize predicates through the simplification of syntactic relations.
    - Include implicit semantic information of the text.
    - Analyse the contribution of the graph based representation in the KBP2011 Temporal Slot Filling Task by comparing a system trained using features from a standard dependency parsing and features extracted from the graph-based representation.

3. Use the previously defined representation to obtain a large collection of documents represented as graphs. Represent corpora both from general domain and specific domain.

4. Define and implement a automatic knowledge acquisition methodology to extract knowledge into propositions.

    - Use the graph representation to extract propositions at document level.
    - Extract propositions using both syntactic and semantic patterns.
    - Include information about named entities and semantic classes.

5. Build and evaluate a Proposition Store. Use the acquisition methodology to compile a large knowledge base of propositions from the collection of graphs.

    - Asses plausibility of propositions based on probabilistic estimation.
    - Report results in several different large corpora.
    - Validate the utility of the Proposition Store in a named entity disambiguation task by harnessing propositions as evidence for testing the One Sense Per Collocation hypothesis.

6. Improving Parsing with Automatically Acquired Semantic Classes.

- Build a gold standard of appositive dependencies. Identify and analyze a set of sentences with ambiguous apposition dependencies and classify the types of errors that parsers commit.
- Develop a method for appositive correction based on background knowledge, specifically in class-instance relations.
- Study different metrics to measure the semantic compatibility and evaluate the results.

7. Semantic parsing task: Grounding Proposition Stores for Question Answering over Linked Data.
   - Study the contribution of knowledge acquisition on closing the gap between natural language and database predicates.
   - Evaluate quantitatively which syntactic structures are useful for alignment purposes.
   - Enable a standalone evaluation of the knowledge acquisition step in the semantic parsing task, abstracting away the impact of training and querying.
   - Improve the alignment through WordNet as a source for external linguistic knowledge.
8. Perform an extensive analysis of the results, and draw global conclusions.
9. Propose a new set of research lines consistent with the conclusions of the thesis.
10. Report the advances in the research in national and international conferences and journals.
11. Write this dissertation summarizing the research conducted, the final contributions to the community and the conclusions obtained.

1.6 Outline

This thesis is composed by seven chapters. In the following lines we provide a summary of the contents of each chapter:

Chapter 1: Introduction.

We motivate our study of the acquisition and application of propositional knowledge and define the scope of the thesis through a set of hypotheses, objectives and research questions.

Chapter 2: Background and Related Work.

Language interpretation poses many challenges, among them, there are three key areas: Meaning representation, knowledge acquisition and textual inference. In this chapter we discuss some of the most relevant work in these areas and identify some unsolved problems.

Chapter 3: Graph Based Representation of Text.

In this chapter we explain the characteristics of the representation that we use as the basis for the knowledge acquisition. This is a representation at document level, obtained from plain text to a graph representation through deep text processing techniques. We also measure the contribution of using this representation for feature generation in an information extraction task.

Chapter 4: Automatic Capture of Propositional Knowledge.

This chapter is devoted to general-purpose knowledge acquisition. Our approach extracts propositions at document level from large corpora of documents represented with the previously presented graph based representation. These propositions express background knowledge that are the base for further textual inferences.

Chapter 5: Improving Parsing with Automatically Acquired Semantic Classes.

In this chapter we define our method to exploit semantic class knowledge to improve parsing on appositive structures. To do so, we select the grammatical option that is semantically more compatible. We define a set of measures of compatibility and evaluate its performance.

Chapter 6: Grounding Proposition Stores for Question Answering over Linked Data.

We show how grounding propositions within a distant supervision framework can improve the performance of question answering: Our approach converts utterances into graphs, which in turn are used to extract propositions. The propositions are aligned with Freebase labels using distant supervision through entity linking. The output is used to feed a semantic parser for question answering over linked data.

Chapter 7: Conclusions and Future Work.

In this chapter we state the main conclusions obtained and we discuss some interesting lines of research.

Additionally, we include the following appendices with complementary information:

Appendix A: Publications of the Author.

This appendix shows the complete list of papers related to the development of this thesis.

Appendix B: Graph-based Representation Example.

This appendix provide an example of a document represented using the graph based representation considering the two possible outputs, Dot and pseudo-JSON.

Appendix C: Proposition Store Examples.

This appendix shows the most frequent propositions of each Proposition Store.

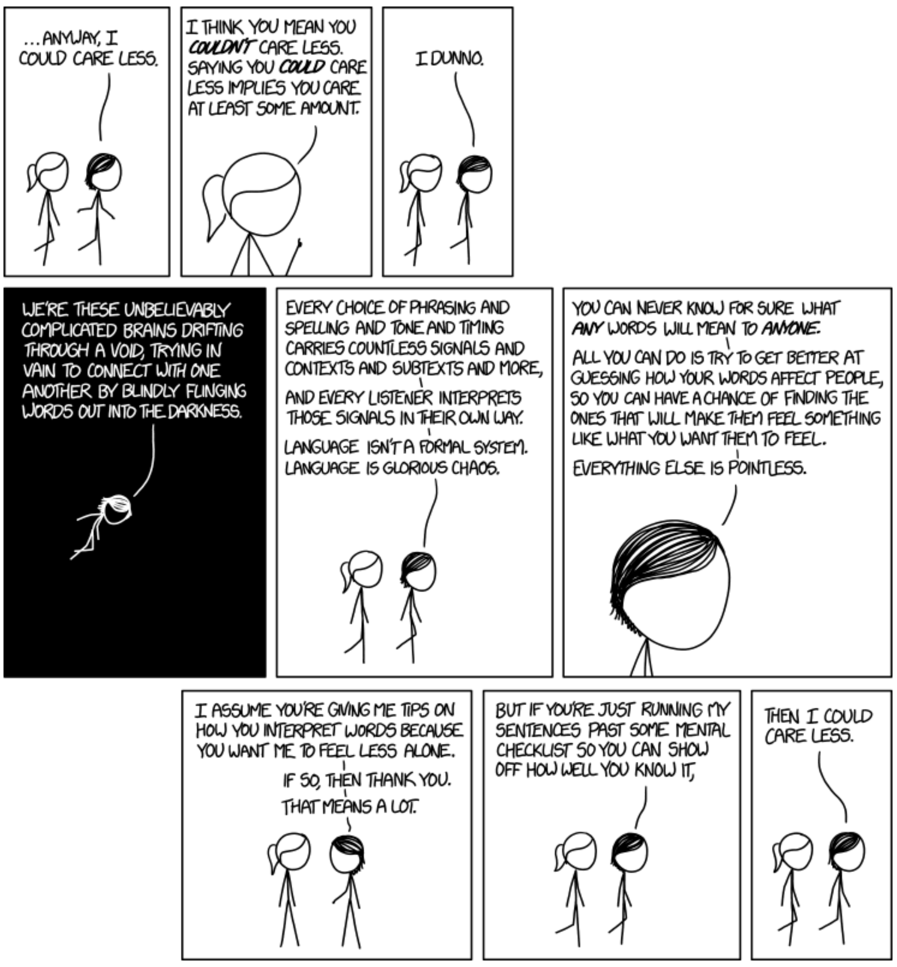
Appendix D: List of Acronyms.

This appendix details the acronyms used in this thesis.

2

Background and Related Work

Language interpretation poses many challenges, among them, there are three key areas: Meaning representation, knowledge acquisition and textual inference. In this chapter we discuss some of the most relevant work in these areas and identify some unsolved problems.


...ANYWAY, I COULD CARE LESS.
I THINK YOU MEAN YOU COULDN'T CARE LESS. SAYING YOU COULD CARE LESS IMPLIES YOU CARE AT LEAST SOME AMOUNT.
I DUNNO.
WE'RE THESE UNBELIEVABLY COMPLICATED BRAINS DRIFTING THROUGH A VOID, TRYING IN VAIN TO CONNECT WITH ONE ANOTHER BY BLINDLY FLINGING WORDS OUT INTO THE DARKNESS.
EVERY CHOICE OF PHRASING AND SPELLING AND TONE AND TIMING CARRIES COUNTLESS SIGNALS AND CONTEXTS AND SUBTEXTS AND MORE, AND EVERY LISTENER INTERPRETS THOSE SIGNALS IN THEIR OWN WAY. LANGUAGE ISN'T A FORMAL SYSTEM. LANGUAGE IS GLORIOUS CHAOS.
YOU CAN NEVER KNOW FOR SURE WHAT ANY WORDS WILL MEAN TO ANYONE. ALL YOU CAN DO IS TRY TO GET BETTER AT GUESSING HOW YOUR WORDS AFFECT PEOPLE, SO YOU CAN HAVE A CHANCE OF FINDING THE ONES THAT WILL MAKE THEM FEEL SOMETHING LIKE WHAT YOU WANT THEM TO FEEL. EVERYTHING ELSE IS POINTLESS.
I ASSUME YOU'RE GIVING ME TIPS ON HOW YOU INTERPRET WORDS BECAUSE YOU WANT ME TO FEEL LESS ALONE. IF SO, THEN THANK YOU. THAT MEANS A LOT.
BUT IF YOU'RE JUST RUNNING MY SENTENCES PAST SOME MENTAL CHECKLIST SO YOU CAN SHOW OFF HOW WELL YOU KNOW IT,
THEN I COULD CARE LESS.

I Could Care Less. https://xkcd.com/1576/

Contents

2.1 Introduction

Interpretation of natural language is one of the central challenges for the development of an artificial intelligence. In general, interpretation requires to build a context of entailed implicit information (from hearer and speaker background knowledge) that permit to recover the original communicative intention. For example, if we know that books are usually written by adults, we can (probably) discard that a *children's book* is a *a book written by children* and then infer that it is *a book written for children*. Interpretation has been extensively studied, but so far it remains as an unsolved problem. As a result, interpretation is often simulated with simplified tasks such as mapping text to a queryable database or an ontology, or to solve problems such as lexical ambiguities or selectional preferences.

A key point for interpretation is the availability of the background knowledge. Typically, background knowledge is simulated using manually built resources, and hence their scalability is limited and coverage are far from what is needed for NLU. Several works point to automatically acquired knowledge, and specifically knowledge acquired from textual sources, as a promising method to build resources of general purpose knowledge. We explore the core problem of acquiring background knowledge directly from previous readings of large amounts of text, in a way that it can be used for the textual inferences that language interpretation requires. Coming back to our initial definition, the goal is *"to build the context that permit to recover the original communicative intention"*. In our view, there are three main issues that have to be tackled in order to perform interpretation: meaning representation, knowledge acquisition and textual inference.

Firstly, natural language has to be represented in a computer-operable form, which is the problem of **meaning representation**. That is, computers require a manipulable structured format that can encode the meaning conveyed by a text, plus a method to transform natural language into such a format.

Secondly, interpretation capabilities depend on the background knowledge available. Thus, a pre-requisite to natural language understanding is **knowledge acquisition**. In order to be used, knowledge has to be stored. As meaning representations, knowledge representations are computer oriented formats. However, a meaning representation aims to encode the information conveyed by a text, while the aim of a knowledge representation is to efficiently store large amounts of information abstracting away linguistic realizations, while providing reasoning capabilities over the stored information. Specifically, we want to study forms of knowledge distilled from raw text that are represented and stored in structures close to natural language. Our hypothesis is that this is the way to reduce the gap between meaning representation and knowledge bases.

The last issue is to perform **textual inference** to recover missing information given the meaning representation and the available knowledge.

Each of these three challenges, meaning representation, knowledge acquisition and textual inference, influence the outcome of the interpretation. As a result, often specific solutions are tailored for concrete tasks obviating the general problem of language interpretation.

Although the three steps are heavily related, these three research areas have been addressed by separate research communities leading to partial models of interpretation: Most meaning representation approaches are based on linguistic models, but usually there is a lack about how to connect them to background knowledge bases. In most cases, efforts barely surpass entity linking. However, the meaning representation imposes a limit on what semantics can be mapped from natural language to the knowledge representation.

Knowledge acquisition processes define the type and scale of the knowledge extracted. Popular approaches are based on predefined data models that permit to store facts and beliefs. The relationship to natural language is artificial and some kind of mapping learning is required, that usually yields a partial acquisition.

Finally the inference step is affected both for the meaning representation and for the knowledge stored. The larger the gap between both representations the more limited inferences we can expect.

In this chapter, we discuss previous works on how to automatically build and use a knowledge base for language interpretation. It is structured in the following sections: First, Section 2.2 summarizes the end goal of language interpretation and reviews the approaches that simultaneously address meaning representation and its connection to background knowledge. Then, we study the problem of representation from different perspectives in Section 2.3. In Section 2.4 we explore recent approaches to the knowledge acquisition problem. Section 2.5 is devoted to the use of textual inference. Finally, in Section 2.6 we show our conclusions and justify the need of a joint proposal to address the three problems simultaneously.

2.2 Language Interpretation

According to the Cambridge Dictionary, **interpretation** can be defined as:

“to decide what the intended meaning of something is.”

Specifically considering **language interpretation**, Hobbs (Hobbs et al., 1993) gives another definition:

"The process of interpreting sentences in discourse can be viewed as the process of providing the best explanation of why the sentences would be true"

Interpretation, an ability that humans perform effortlessly, has been a challenge for computers from the beginning of the artificial intelligence. Since a full interpretation system is still deemed as out of reach, interpretation has been tackled by computers differently from humans, mainly by addressing on particular sub-problems.

For instance, we mention the Turing Test at the beginning of this thesis as the prototypical task for showing human intelligence in machines. In the Turing test, a computer pretends to be a human through a natural language dialog with human judges. However, the test does not systematically address some of the fundamental challenges of NLU such as the ability to recognize textual entailment, to summarize text or to understand complex meaning relations, and it is often deceived with techniques that do not aim to comprehend language, and instead they rely on statistical techniques to search in an answer space.

To avoid these shortcuts, in the past years some more modern tests have been proposed to demonstrate NLU through synthetic datasets, primarily in question answering since evaluation is simpler than in other heavy-knowledge dependent tasks like summarization or textual entailment, specially when answers are drawn from a close space (i.e. true-false or multiple choice scenarios). Each test focus on different subproblems, and therefore require a special set of skills to be solved.

The Winograd Schemas (Levesque et al., 2012) present a simple statement and a binary choice question, as in the example, *"'The city councilmen refused the demonstrators a permit because they feared violence. Question: Who feared violence? Answer 1: The city councilmen. Answer 2: The demonstrators".* This task aims to evaluate the ability of the systems to retrieve relevant common sense knowledge.

Both QA4MRE (Peñas et al., 2011) and MCTest (Richardson et al., 2013) provide a short paragraph where a system have to find an answer for a question. This task requires both knowledge on how to read the text and also common sense knowledge.

In the bAbI tasks (Weston et al., 2015), the goal is to require episodic memory that captures the long-term structure within a sequence. Each question is associated with a set of statements that describe a series of events. Then, a system must connect the pieces of information among the statements in order to retrieve the answer. This is related to general dialogue systems, which, despite being a long-term goal of AI, remain out of the state of the art because current systems are not able to reason over a given story, that is, to iteratively receive statements in natural language and find coherent answers. Consider the following example extracted form (Weston et al., 2014):

Prior to any question, a system is feed with a story composed by several actions:

- *Joe went to the kitchen.*
- *Fred went to the kitchen.*
- *Joe picked up the milk.*
- *Joe travelled to the office.*
- *Joe left the milk.*
- *Joe went to the bathroom.*

The story must be comprehended in order to response to some queries:

- *Where is the milk now?*
- *A: office*
- *Where is Joe?*
- *A: bathroom*
- *Where was Joe before the office?*
- *A: kitchen*

In the following sections we review full proposals for language interpretation. First, we go back to the eighties and nineties to find classic interpretation theories, and then we review rising efforts in the stream of Open Information Extraction that are able to generate Background Knowledge Bases directly from the linguistic processing of texts.

2.2.1 Conceptual Dependency Theory

The Conceptual Dependency Theory (CDT) (Schank, 1972) is an early theory about NLU that stated that sentences with the same meaning should be represented equally, and this representation should contain all the information contained in a sentence, both explicit and implicit. The model encodes meaning using a predefined set of primitives, which represent possible actions in the world; a set of states, which act as preconditions and results of the actions; and dependencies, that are used to connect primitives with each other and with actors, objects or other kinds of arguments.

For instance, a `PTRANS` is a primitive that describes the transfer of location of an object. `PTRANS` has four slots that denote what kind of objects can participate on the action: `ACTOR`, the concept that realizes the transfer; `OBJECT`, the object transferred; `FROM`, the location where the transfer begins; and `TO`, the location where the transfer ends. The inference rules use the primitives to make explicit the implicit information. For example, the `PTRANS` primitive can be used to infer that the object that participates in the transfer was initially in the `FROM` location and after the transfer is in the `TO` location.

Although the first claim was that these elements were enough to produce a canonical form of natural language sentences, this theory was deemed as insufficient soon after since the set of primitives was really small (from 10 to 12 primitives) to represent the amount of knowledge that can be expressed with natural language. However, it still a challenging task to map natural language even to this small set of actions.

This theory is further refined with the concepts of scripts, goals and plans (Schank and Abelson, 1977) that model not only the contents of the sentences, but the stereotypical sequence of events that are often described by sentences. These underlying models are used to infer missing information from a sentence based on the expected intention of the writer. A complete review of the development of CDT and associated theories until early 90's is provided in (Lytinen, 1992).

2.2.2 Meaning Text Theory

A different approximation to interpretation is Meaning-Text Theory (MTT) (Mel'cuk and Polguère, 1987). MTT is a linguistic framework that states that language is structured in four layers, phonetics, morphology, syntax and semantics, and the process of understanding involves the mapping from the lower layer, phonetics, to the top layer, semantics. Conversely, language generation would imply a top-bottom mapping. The relation between layers is similar to a translation. With this idea, it provides a methodology to interpret language step by step in increasing complexity. So far, the translation from phonetics to syntax is largely solved, and many research efforts point to translate syntax into semantics. However, unlike scene construction, MTT focus on single sentences and does not address the meaning of a complete text.

MTT uses different meaning representations to encode the knowledge of each layer. The semantic representation (SemR) uses a graph representation that relates predications with arguments with directed edges. Moreover, a predication can act as an argument of another predication, and an argument can act as argument of many predications. The syntax is divided in two main levels, deep syntactic representation and surface syntactic representation, the first is used to represent the universal dependencies between lexemes and the second to encode the particularities of each language.

2.2.3 Discourse Representation Theory

Discourse Representation Theory (DRT) (Kamp, 1988; Kamp and Reyle, 1993) is a theory that views interpretation as the process of constructing a picture in which new elements are incorporated as they appear on the text. DRT is closely related to File Change Semantics (Heim, 1983). DRT defines discourse representation structures (DRS), which are cognitive models that are built incrementally with both the information contained in the sentences plus the relation between sentences itself. This theory is capable to tackle problems such as the resolution of anaphora, the model of verbal tense, and quantification of variables. Unlike MTT and other interpretation conceptions, DRT does not concern about information truthness, and focus on the added information that each new sentence conveys.

DRS are typically represented using first order logic. A DRS contains two types of information: information about discourse referents and information about the properties of these referents and how they are related. A discourse referent is an entity that takes part on the discourse and that can be refereed to with different anaphoric expressions. DRSs may be connected using DRS languages, which are operators similar to first order logic such as negation ($\neg$), disjunction ($\vee$) and implication ($\implies$).

The main implementation of the DRT is the Boxer parser (Curran et al., 2007), which departs from combinatory categorical grammars parsers to generate a discourse representation structure. Other notable DRT based parsers are (Baldridge and Lascarides, 2005; Lin et al., 2014).

2.2.4 Interpretation as Abduction

In Hobbs words, *"Abduction is inference to the best explanation"*. Abduction refers to the process of adding new information based on the most plausible explanation for some known facts. Human use abductive inferences to connect sentences to their previous background knowledge in order to build a coherent meaning.

Abduction is appealing for discourse interpretation over other inference mechanisms such as deduction because humans naturally omit information that listeners are supposed to know. Therefore, in order to build language interpretation systems, these should be able to retrieve this missing information. Typically, sentences are represented using first order logic with two main goals. On the one hand, redundancies can be merged by matching variables, and on the other hand, new predicates can be created where an assumption is necessary.

Utterance	Logical Form	Denotation
one plus two	1 + 2	3
Who is the wife of Obama	Obama - married_to	Michelle Obama
Move the red brick to the right	move(red brick - right)	`move(red brick - right)`

Table 2.1: Examples of semantic parsing.

Discourse interpretation through abduction was a popular topic during the 80s and 90s (Charniak and Goldman, 1989; Hobbs et al., 1993). Specifically, the Tacitus Project (Hobbs et al., 1988, 1993) aim to build a large knowledge base of commonsense and show how abduction can be used to resolve cases of syntactic ambiguity such as interpretation of nominal compounds and metonymy resolution.

Recent work (Ovchinnikova et al., 2014b) builds a framework to process discourse for similar tasks, including resolution of syntactic and semantic ambiguity, interpretation of nominal compounds and reference resolution for the final goal of recognizing textual entailment. Abduction is currently a popular research topic with many examples of notable reasoners for many tasks such as interpretation of metaphors (Ovchinnikova et al., 2014a), solving lexical ambiguities (Blythe et al., 2011) or plan recognition (Kate and Mooney, 2009; Singla and Mooney, 2011)

2.2.5 Semantic Parsing

Semantic parsing can be described as the task of mapping input utterances u into logical forms l. In most semantic parsers, especially those devoted to question answering, logical forms are executed to obtain a denotation d. For example, the utterance $u =$ *one plus two* can be mapped into the logical form $l = 1 + 2$ and further executed to obtain the denotation $d = 3$. There are several kinds of semantic parsers depending on the type of action that the system has to produce. For example, given the utterance *move the red brick to the right* the system has to produce the executable action *move(red brick, right)*. Table 2.1 show some examples of utterances, logical forms and denotations for different semantic parsers.

Semantic parsing can be seen as a translation from natural language into a logical form (Liang, 2015). This idea has two practical implications: First, expressions, like entities, are symbols that only have a meaning in the concrete context of the parser. The importance of this property lies in that semantic parsing is independent on the execution. The second practical implication is that the final meaning of an utterance is composed by the union of its subexpressions, that is, it is compositional. This property allows us to express a vast amount of meaning by composing simple and small units.

There are two main subproblems involved: Data representation and grounding. Data representation refers to choosing a meaning representation, which is often solved through intermediate representations, i.e. combinatory categorical grammar (Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2013), synchronous context-free grammars (Wong and Mooney, 2007), dependency based compositional semantics (Liang et al., 2013; Berant et al., 2013; Berant and Liang, 2014) or latent representations (Berant et al., 2013; Kwiatkowski et al., 2013). Although it is an interesting research topic, to the best of our knowledge there is not an in-depth comparison. In any case, choosing a good representation depends on the type of inference (i.e. grounding) that the system is performing.

Grounding refers to the process of mapping the meaning representation to the target vocabulary. The more structured the target vocabulary is, the easier the grounding. However, less structured target vocabulary can represent larger amounts of knowledge, at the cost of being less exploitable by automatic systems.

Traditionally, semantic parsing models are trained by aligning hand-labeled logical forms with the target ontology (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005). However, building a large corpus of manually annotated utterances is expensive, and up to this point this procedure is not applicable for open domain systems that aim to accept utterances with high variability. There are two main focus on semantic parsing development. First, semantic parsers typically depend on large amounts of annotated data. Recent research propose different methods to reduce supervision; sentences aligned with system behaviour (Chen and Mooney, 2011), using paraphrases (Berant and Liang, 2014; Fader et al., 2013b), question-answer pairs (Berant et al., 2013) and align databases with text (Reddy et al., 2014). On the other hand, there is interest on scaling semantic parsers up to web size.

There are two main uses for semantic parsing. First, it can be used as a method to generate a canonical meaning representation from natural language. Even though dependency parsing represents a great advance for many NLP tasks, the algorithms of parsing heavily rely on the syntactic tree structure to be efficient. This structure is not compatible with a more semantic view of the representation. Semantic parsers tackle this issue, transforming text into graphs where a node can be an argument for many predicates. This task is also very related with semantic role labeling, as it usually involves some sort of argument identification and labeling for verbal and nominal predicates. However, this task also deals with other semantic phenomena, such as negation, possessives and comparatives.

The second main use of semantic parsing is Question Answering (QA). Although it can be solved using other methods such as information extraction, one popular approach of QA is to interpret a question, that is, an utterance by obtaining a logical form that expresses the semantics embedded. The logical form can be transformed into a database query, that is further executed into a structured knowledge base in order to retrieve an answer. The main motivation is that users without training on database management are able to find information by expressing their needs with their own language.

Traditional QA uses ever-growing knowledge bases (Kwiatkowski et al., 2013; Berant et al., 2013; Fader et al., 2013b, 2014; Berant and Liang, 2014) in which a logical form extracted from the question may be executed to retrieve an answer.

Attention models and memory networks have shown to be promising options in task that require to simulate episodic memory (Weston et al., 2014; Sukhbaatar et al., 2015; Kumar et al., 2015). Although these models are powerful, they still scale worse than other simple machine learning algorithms.

A new interesting line of research aims to solve questions both in images and in text using the same learning method (Andreas et al., 2016).

Currently semantic parsing is attracting the attention of the research community as it can be seen in many tasks such as the SemEval 2014 Task 8: Broad-Coverage Semantic Dependency Parsing [i], the Question Answering over Linked Data series [ii] or the ACL 2014 Workshop on semantic parsing[iii].

2.3 Meaning Representation

Machines require to manipulate the semantics of a sentence through a formal representation. The process of translating natural language into a representation is known as **semantic analysis**. The notation of the representation is known as **meaning representation**, and the framework that specifies the syntax and semantics of a meaning representation is the **meaning representation language**.

A meaning representation is required to process semantically the input utterances, specially when it involves the use of background knowledge that is not present in the input. Therefore, choosing a representation is important because it has an impact on which semantics can be captured and what kind of inferences can be done.

A representation is useful for a task as long as it provides the right computational properties needed. However, it is desirable to build representations that are reusable across different tasks. Although it may be subjective which are these properties, from a human point of view it should be easy to distinguish which representations actually

[i]http://alt.qcri.org/semeval2014/task8/
[ii]http://qald.sebastianwalter.org/
[iii]http://sp14.ws/

encode the meaning of the text. We discuss some of the properties that we aim to capture in our representation. An extensive explanation of the desiderata of a more generic broad-coverage semantic representation is included in (Schubert, 2015).

Hobbs (1985) argues that a representation language has to be defined according two criteria:

1. *The notation should be as close to natural language as possible.* This criterion refers to the ability of a language to encode every fact. Therefore, according exclusively with this criterion, natural language would be the ideal choice.
2. *The notation should be syntactically simple.* The representation should simplify the manipulation of the facts extracted. Unfortunately, natural language is in disadvantage against other structured representations like ontologies.

In the following sections we describe some of the most popular meaning representations: Logic-based representations, syntactic representations, semantic roles and frames, abstract meaning representations and distributional representations.

2.3.1 Logic Based Representation

As a meaning representation, logic formalisms are based in the reasoning with logical forms. A logical form is an expression defined in a fully specified, unambiguous artificial language. Logical forms are one of the most popular meaning representations, and often these terms are used as synonyms.

Logic formalisms aim to provide a general purpose meaning representation that can be expressed without any particular tie to the domain. This property makes logic formalisms more expressive that formalisms tied to predefined models.

First Order Logic (FOL) is perhaps the most representative case of logic formalism. In FOL, logic forms are composed by terms, that denote objects of the domain. Forms can be divided in constants, functions and variables. A constant is a specific object in the world, such as a proper noun (e.g. $Madrid$). A function represent a concept in the world, for example those denoted by genitives (e.g. $LocatedIn(Madrid)$). Functions are equivalent to unary predicates. Variables are references to anonymous objects. Variables are the key for inference purposes.

A predicate is a relation that involves several terms (e.g. $CapitalOf(Madrid, Spain)$). Predicates can be connected using logic connectives, i.e and ($\wedge$), or ($\vee$) and not ($\neg$). Finally, quantifiers can be used to denote that a variable refer to a particular object or to every object in a collection, that is, the existential quantifier $\exists$ and the universal quantifier $\forall$. For example $\exists x CapitalOf(x, Spain)$ denotes that there is an entity that is the capital of Spain. FOL provides a sound computational basis for inference.

Different versions of logic formalisms have been extensively used in NLU, such as the mentioned FOL (Bird et al., 2009), lambda calculus (Carpenter, 1997), natural logics (MacCartney and Manning, 2009; Moss, 2010), montague grammar (Montague, 1970), diagrammatic languages (Kamp and Reyle, 1993), robot controller languages (Matuszek et al., 2013) and database query languages (Zelle and Mooney, 1996).

Perhaps one of the most salient variation of FOL was ontological promiscuity (Hobbs, 1985). This meaning representation aimed to be simple as well as general. To do so, Hobbs dispensed with three constraints that in his opinion made previous representations such as semantic networks unable to capture some aspects of language while at the same time being able to perform inference. These constraints regard ontological structure, syntactic constraints and efficient inference. He proposed a flat logical notation with an ontologically promiscuous semantics. In his own example, *"A boy wanted to build a boat quickly"* can be represented through a conjunction of atomic quantified predicates:

$$\begin{split}(\exists e_1, e_2, e_3, x, y) Past(e_1) \wedge want'(e_1, x, e_2) \\ \wedge\, quick'(e_1, e_3) \wedge build'(e_3, x, y) \wedge boy(x) \wedge boat(y)\end{split} \tag{2.1}$$

Which means, e_1 occurred in the past, e_1 is the event of x wanting e_2, e_2 is the event of e_3 being quick, and e_3 is the event of building y, and x is a boy and y is a boat.

Note the reification process. The same utterance could be interpreted as:

$$(\exists x, y) Past(quick(build(boy(x), boat(y)))) \tag{2.2}$$

The logical form 2.2 is a nested version of logical form 2.1. The method to translate a nested logical form into a flat form is to reify the nested expressions, that is, to create a event that denotes each of the predicates of the logical form, in this example, e_1, e_2 and e_3.

Some authors depart from dependency trees to generate reified logic forms (Agerri and Peñas, 2010). There are many meaning representations inspired in ontological promiscuity (Mollá, 2001; Ovchinnikova et al., 2014b).

2.3.2 Syntactic Representation

A syntactic representation models sentences considering the relations of its linguistic units. Strictly, syntactic representations are not meaning representations. However, syntax and semantics are close (See Meaning-text theory in Section 2.2), and we describe syntactic parsing here as a common method to structure raw text. In fact, current syntactic dependency representations, such as Stanford's CoreNLP) are very close to a meaning representation.

Syntax is defined as "The arrangement of words and phrases to create well-formed sentences in a language". Syntax plays an important role on defining the meaning of a sentence, and a syntactic parsing, which is the task of label text with the syntactic elements, is often used as a preprocessing tool for NLP tasks. We discuss here two main notions of syntactic parsing, constituency and dependency parsing. Both constituency and dependency parsing can be defined by grammars. A grammar is defined by a lexicon, which are the symbols of the domain, and a set of rules, which define the way that symbols can be combined. Symbols are divided in terminal symbols, i.e. words in the language, and nonterminals, which express generalizations of terminals and are particular to the grammar, for instance, verbal phrases and noun phrases.

Constituency parsing groups words into constituents, which are linguistic elements that behave as a single phrase. Relations between constituents are unlabelled. The traditional approach to perform constituency parsing is grammar-driven, that is, first a grammar is defined and then the problem is how to map a text with that grammar.

However, predefined grammars lack of the generalization capabilities that natural language requires. Most of the parsers build a probabilistic grammar from a Treebank, a corpora of annotated sentences. For English, the most used Treebank is the Penn Treebank (Marcus et al., 1993). Context Free Grammars (Booth and Thompson, 1973; Baker, 1979) are early examples of probabilistic grammars. The two main approaches for statistical parsing are the head-driven phrase structure grammar (Pollard, 1994) with salient examples as the Collings parser (Collins, 2003), the Charniak parser (Charniak and Johnson, 2005), the Stanford parser (Klein and Manning, 2003; De Marneffe et al., 2006); and the Lexical Functional Grammar (Kaplan and Bresnan, 1982). More recently, these techniques were combined with neural networks to further improve the results as in (Socher et al., 2013).

In dependency parsing, every word is a lexical item, and it is linked to other words with a directed labelled relation called dependencies. The result is a sentence is represented as a directed tree in which nodes are words, edges are dependency tags and there is an additional special node as the unique root.

Dependency parsing has been tackled in several CoNLL shared tasks (Buchholz and Marsi, 2006; Nilsson et al., 2007). As with constituency parsing, the focus has switched from pre-specified grammars to probabilistic data-driven grammars. This process is called inductive dependency parsing (Nivre, 2005). Some of the salient models used include transition-based dependency parsing (Nivre, 2004), graph-based dependency parsing (McDonald and Nivre, 2007) and neural networks (Chen and Manning, 2014).

Traditional parsers provide their output in one of the three main sets of dependency types: the ConLL dependency parsing shared tasks (Buchholz and Marsi, 2006;

Nilsson et al., 2007), Stanford dependency labels (De Marneffe and Manning, 2008) or unlabelled dependencies. Recently, the Stanford Universal Dependencies (Nivre et al., 2016; Schuster and Manning, 2016) was released with the goal of providing a normalized set of dependency relations across languages.

Some examples of popular available dependency parsers are: Minipar (Lin, 2003) (extended) Stanford Parser (De Marneffe et al., 2006), MaltParser (Nivre et al., 2006), Ensemble Parser (Surdeanu and Manning, 2010) or FANSE parser (Tratz and Hovy, 2011).

Syntactic parsing has been extensively used for all types of NLP tasks such as summarization, question answering and textual entailment.

2.3.3 Semantic Roles and Frames

A important drawback of syntactic parsers is that they do not represent the full meaning of a sentence. Different syntactic realizations of a sentence, for instance active and passive voice, may convey the same meaning, but the syntactic dependencies retrieved are not equivalent. Consider the following sentences:

(2.3) They cancelled the meeting.

(2.4) The meeting was cancelled.

A syntactic analysis would indicate that *the meeting* is the direct object in sentence 2.3 and the passive subject of the sentence 2.3. The **semantic role** of a constituent is the role that it plays on a sentence regardless of the syntactic realization. These constituents are the phrases that point the agent (who), the patient (whom), place (where), cause (why), time (when) and others. In both examples, *the meeting* plays the same semantic role, which is being the patient of the event *cancel*.

The number and type of semantic roles depend on the specific predicate. Although there are some common types of roles that are shared across many predicates, such as subject and object, it is difficult to define a set of semantic roles that covers all the possibilities.

PropBank (Palmer et al., 2005) is a corpus manually annotated with numbered semantic roles. The argument *Arg0* denotes the prototypical agent, and *Arg1* denotes a prototypical patient or a theme. Subsequent arguments denote semantic roles such as location, extend and cause. The specific meaning of each argument depends on the predicate, which is verb sense draw from WordNet. That is, even is the same labels are used for all verbs, the interpretation of the meaning of the label is verb-specific. For some tasks, choosing verbs as predicates poses a problem because synonyms are not grouped. Consider the following sentence:

(2.5) The meeting was called off.

In this case, the meaning and the semantic roles are equivalent to sentences 2.3 and 2.4. However, the predicate is different, thus the meaning is represented differently.

A frame is conceptual structure that represents a prototypical situation or event. A frame is characterized with several frame elements, which are frame-specific roles. For instance, the frame *Transportation* contains the frame elements *Movers*, *Means* and *Path*. Besides, frames can be arranged hierarchically, e.g. *Driving* can be a subframe of *Transportation*.

The drawback is that frames have to be defined by hand some verbs may not be represented by any frame, and, depending on the granularity, related verbs can be grouped under the same frame, losing some meaning nuances.

Both FrameNet and PropBank have been extensively used for NLP tasks such as question answering (Shen and Lapata, 2007; Kaisser and Webber, 2007) or word sense disambiguation (Chen and Palmer, 2009; O'Hara and Wiebe, 2009). Together with VerbNet and WordNet, they have been interlinked through the project SemLink (Palmer, 2009) as they provide complementary information.

The tasks of automatically tag a sentence with its frames and semantic roles is called Semantic Role Labelling (SRL). Traditional SRL uses syntactic trees as a base, often aligned with annotated text (Gildea and Jurafsky, 2002; Hacioglu et al., 2004). These methods generally are divided in two steps, first identify the arguments to be labelled and then tag these arguments. The tendency now is to move to unsupervised SRL or Semantic Role Induction to spread the annotation across different languages and domains (Swier and Stevenson, 2004; Lang and Lapata, 2014). It is a key process for many NLP applications, and therefore there is a long research tradition. A complete overview of the field is presented in (Palmer et al., 2010).

SRL is closely related to predicate-argument identification(McCord et al., 2012), where roles are substituted by arguments that lack of a specific semantic as agent or patient, and instead they perform a different undefined action depending on the predicate.

2.3.4 Abstract Meaning Representation

Abstract Meaning Representation (AMR) (Banarescu et al., 2012) is an specification of a semantic representation that captures "who is doing what to whom" in a single simple data structure. Ideally, sentences with the same meaning lead to identical AMR. It abstracts a limited amount of morphological and syntactic variability in order to facilitate the annotation, thus it is feasible to create a large AMR bank. Although an older version of AMR was available since late 90's (Langkilde and Knight, 1998), the redefinition of the standard by Banarescu et al. plus the availability of new tools and annotated resources explain the popularity of AMR. The main advantage compared to previous formal semantic is that AMR agglutinates annotation for named entities, coreference, semantic relations, discourse connectives and temporal entities.

In AMR each sentence is represented as a tree in which edges represent relations and leaves represent concepts. Relations use the PropBank lexicon when available, and stemmed forms if not. Named entities are linked to Wikipedia when possible, and measurable entities such as date and quantities are normalized. Coreference entities are merged in a single leave. However, AMR do not consider coreference across different sentences. Unlike dependency parsers, it does not annotate the individual words in a sentence.

The main motivation of AMR is to unify several strands of research on semantic representation. It includes semantic role labelling, named entity recognition, coreference resolution, temporal annotation and discourse connectives.

Currently there is a strong interest in developing tools related to AMR, that took shape into the Abstract Meaning Representation Parsing and Generation SemEval Task[iv] (May, 2016), which result in many ARM parsers (Artzi et al., 2015; Pust et al., 2015; Peng et al., 2015; Vanderwende et al., 2015; Wang et al., 2015; Flanigan et al., 2014) and AMR-annotated corpora (Banarescu et al., 2013; Vanderwende et al., 2015).

2.3.5 Distributional Representation

Distributional representation is based on the distributional semantics hypothesis enunciated by Harris (1954): *semantically similar words occur in similar contexts*. A distributional representation assign a meaning to each linguistic structure, typically a word, but in some cases can be a multi-word or a phrase. The meaning is encoded in a high dimensional vector that is based on a statistical analysis of the text, and the semantic similarity is measured as a distance between vectors.

The expressiveness and the precision of the model grows with the amount of data for training. This property, which was a major drawback in the early years of NLP, makes ideal the actual situation with the availability of large textual datasets.

The basic distributed model is bag-of-words. In this model, a text is represented as a vector with the frequencies of each of its words. This approach is computationally very cheap and allows expressing a brief notion of context. However, this model is very limited, as it loses many semantics encoded on the original text structure.

[iv] http://alt.qcri.org/semeval2017/task9/

Recently, distributed representations are the starting point to build more compressed and powerful vectorial representations due to neural network based approaches like Word2Vec (Mikolov et al., 2013b) and GloVe (Pennington et al., 2014). These techniques produce word embeddings from large unannotated corpora in an efficient, scalable way. Unlike bag-of-words, the resulting vector is several orders of magnitude smaller than the vocabulary of the domain, while at the same time semantic relations are captured in a more powerful way. Word embeddings have been used extensively in subsequent NLP tasks such as speech recognition (Bengio and Heigold, 2014), machine translation (Mikolov et al., 2013a), syntactic parsing (Dyer et al., 2015) and others.

The main drawback of these vectorial representations is that they are not easily interpretable by humans, and therefore it is hard to translate distributed representation outputs into natural language. Furthermore, they still only work at the lexical level.

2.4 Knowledge Acquisition

Language interpretation requires general knowledge of the world that we denote this representation as **background knowledge**. Background knowledge is described with a formal language, known as **knowledge representation**, in order to be effectively manipulated by machines.

A Knowledge Base (KB) is a kind of database that is used for store and manage knowledge, where manage means the capacity to organize and serve that knowledge. A KB contains information that describes a domain, employing for it a representation vocabulary, which is known as representation language.

We focus on computer-oriented databases. They contain information organized for Knowledge Based Systems (KBS) that solve complex problems using artificial intelligence techniques. Encyclopaedic resources such as Wikipedia fall outside this category, although they provide an efficient access to the users that search for texts, usually formative, as manuals and articles. These resources are written in natural language, and therefore intelligent systems cannot use them easily.

Building a KB has been a long time research line since, due to the nature of language interpretation, KBs are required to be very large. Natural language is extensive and can be used to express information about any domain, and at the same time it is precise since it is able to represent specific concepts. However, background knowledge is far from matching both the extension and the precision of natural language to be useful for interpretation.

Roughly, there are three main approaches for knowledge acquisition: Use knowledge engineers to curate data manually (Lenat, 1995; Friedland et al., 2004; Gunning et al., 2010); build a KB by crow-sourcing, employing less trained people, even volunteers, to curate data (Stork, 1999); and finally acquire knowledge automatically (Banko et al., 2007; Ferrucci et al., 2010). We can find also any combination of these approaches. Generally speaking, these methods go from small KBs with high quality data to larger databases with more noise.

So far, the first two methods are considered insufficient to build a background KB since these methods are unable to scale up. Currently, the most promising technique is to automatically gather knowledge, and much of the ongoing effort aims to scale up the process. The rationale behind is that, given enough data, noise would be irrelevant compared to the amount of interesting facts. Textual resources, specially the Web, are currently wide and accessible and are often used for knowledge acquisition. Most of the information available is presented in natural language, and therefore knowledge acquisition from text is potentially the best chance to obtain the most complete KB. Recently, there have been some attempts to automatically acquire knowledge in the context of language interpretation, known as machine reading (Etzioni et al., 2006) or learning by reading (Mulkar et al., 2007).

In the following sections we study the most notable approaches to build knowledge bases considering manually curated databases and four kinds of automatic knowledge acquisition systems: information extraction, knowledge base population, open information extraction and propositional knowledge acquisition.

2.4.1 Manually Curated Databases and Linked Data

Roughly, manual annotation efforts aim to define and populate an ontology. The concept of ontology has been used with different meanings. For our purpose, an ontology can be seen as a predefined graphical structure to arrange data of a domain. Typically it involves the definition of instances, classes and relations. An instance is an element of the domain, and a class is a category of these elements. For instance, *Peter* may be an instance of the class *Person*. Classes represent the nodes of the graph, and can be arranged in a hierarchical structure. For instance, *Person* can be a subset of *Mammal*. Relations compose the edges of the graphs, and define the way that two or more instances relate among them. For example, the relation *married_to* can relate the instances *Peter* and *Mery*. Relations can impose constraints on the allowed classes, for example, *married_to* could accept only instances of the class *Person* as arguments. Classes are often tagged with properties, which are common characteristics of the instances of the class. For example, the class *Birds* could have the property *can_fly*. Ontologies are defined according to an ontology language that states the syntax to represent the knowledge. Common ontology languages include OWL, RDF and RDFS.

A main advantage of ontologies is that it is possible to group related knowledge, making easy to acquire, index and query large amounts of structured knowledge. As ontologies are pre-specified models, they are often tied to a specific task, and perform well on it. However, they are not enough for NLU, because large portions of knowledge do not fit on the model.

First attempts to collect data on ontologies used expert annotators and focused on small domains. Around the middle 80's there were several efforts towards compiling a large common sense KB using human annotators. Most notably, the CYC project (Lenat, 1995) is one of the biggest efforts to encode common sense by hand. The initial idea was to store all the knowledge in 400 sample articles, plus all the implicit background knowledge needed to comprehend them. This project has been running for the past 30 years, and at this point it is distributed through three different platforms, OpenCyc, EnterpriseCyc and ResearchCyc. ResearchCyc 1.1[v], the latest version available, has over 500.000 concepts, 5 million assertions and more than 26.000 relations.

CYC proved to be useful for many tasks such as question answering (Curtis et al., 2005; Lenat et al., 2010) and word sense disambiguation (Curtis et al., 2006), as manual annotations provide accurate data with almost no noise. However, generate a manual KB is very expensive, and therefore they are not updated or the updates are slow and difficult. Eventually, it was clear that it was infeasible to encode enough knowledge to simulate common sense with manual approaches. Besides, ontologies are too rigid to represent the real world.

The acquisition problem found an alternative approach on the Semantic Web and more specifically on its realization through Linked Data. Linked Data is a framework for publishing structured data in format that is manipulable by machines, while at the same time is generic and flexible. The popularity of Linked Data propitiated that many organizations built several large resources, and because of the design, it is possible to integrate data from different repositories. The fact that the data can be integrated from datasets of different domains improve the capabilities of the end systems.

Linked Data is based in several standard Web technologies: Linked Data uses Uniform Resource Identifiers (URIs) (Masinter et al., 2005) to assign a unique identifier to entities or concepts in the world. URIs achieve two goals; first, they enable to decentralise the creation of global names, so every domain can create their own identifiers, and second, URI's provide a way to access to the referenced item using the HyperText Transfer Protocol (HTTP) (Fielding et al., 1999). Also, Linked Data uses the Resource Description Framework (RDF) (Lassila and Swick, 1999) to define a model to represent information in the form of a triple, which is composed by a subject, a predicate and an object. It can be seen as a graphical representation with directed arcs. The subject of a triple is an URI that identifies the described resource.

[v]http://www.cyc.com/platform/researchcyc/

Objects can be URI's or literal values such as strings, dates or numbers. The predicate indicates the relation that exists between subject and object, and it is also identified by an URI. Predicates have to be predefined, and therefore a domain has a limited number of predicates. RDF can be implemented using several different syntax, or serializations, including RDF/XML, Turtle, RDFa or RDF/JSON.

Linked Data databases are devoted to many topics, for example, FOAF (Brickley and Miller, 2012) is a database of people, Geonames (Wick and Vatant, 2012) contains data of locations, Linked Movie DataBase (Hassanzadeh and Consens, 2009) contains information about films, and many others (See Figure 2.1). Several Intelligent systems use linked data for tasks like domain modelling, customization, data integration, improved search and others in different domains such as arts, education and e-learning, e-government, financial, geographical. Specifically for language interpretation, some linguistic resources have been transformed into Linked Data databases: ConceptNet (Liu and Singh, 2004), VerbNet (Schuler, 2005), WordNet (Miller, 1995), SenticNet (Cambria et al., 2014), FrameNet (Baker et al., 1998) and Probase (Wu et al., 2012). Storing factual knowledge we can find DBpedia (Auer et al., 2007), Freebase (Bollacker et al., 2008), WikiTaxonomy (Ponzetto and Strube, 2007), Web Tables (Cafarella et al., 2008) or Wikidata (Vrandečić and Krötzsch, 2014). Some of these databases combine curated data with information extracted automatically, specially in the case of factual knowledge databases.

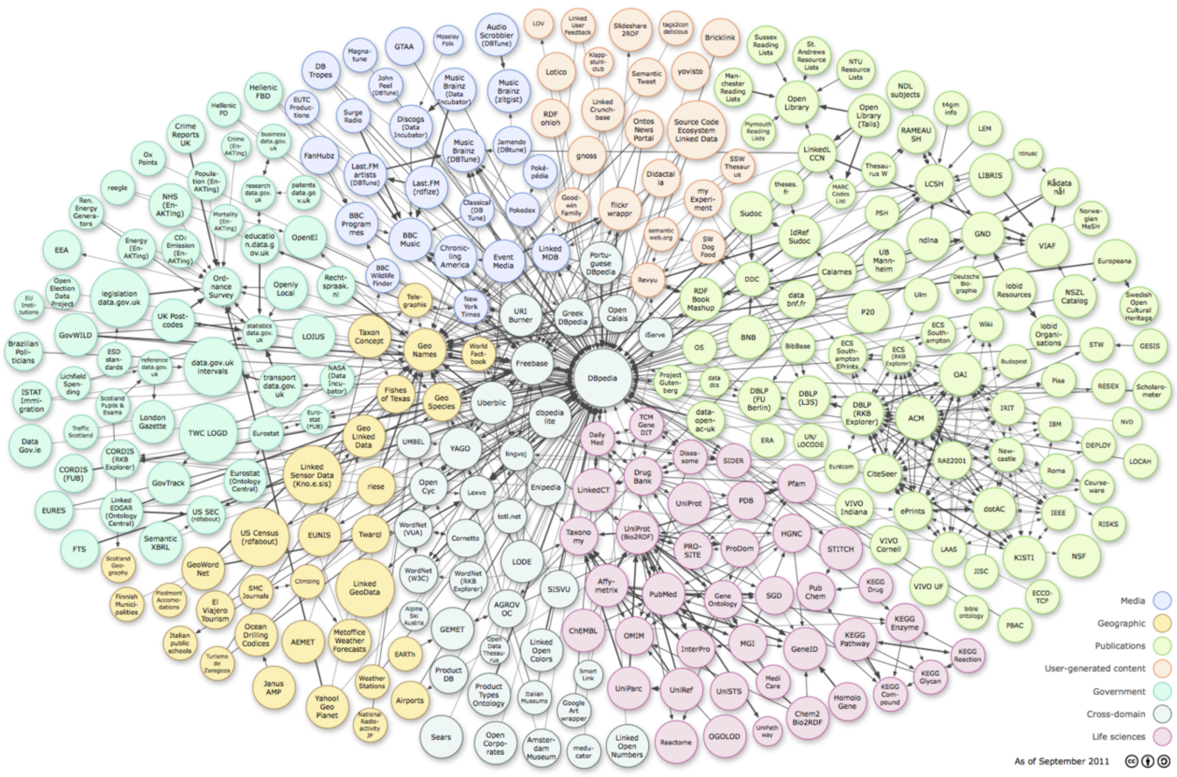


Figure 2.1: LOD Cloud Diagram as of September 2011 by Anja Jentzsch - Own work. Licensed under CC BY-SA 3.0 via Commons

There is a large amount of related open source tools built by the community to create and manage ontologies. For example, one popular tool for storing and serving RDF data is The OpenLink Virtuoso server [vi]. Apache Marmotta [vii] is an open source framework to publish and build custom ontologies.

Ontologies and linked data databases are widely used in research, however, their manual scaling is limited and therefore, some automated paradigms have arisen for ontology population. The following sections show the two main paradigms to automatically extract knowledge tied to a predefined schema. They are micro and macro-reading.

2.4.2 Micro-reading and Information Extraction

Information Extraction (IE) is the task of acquiring structured information from either natural language or semi-structured language. We use the term **Micro-reading** as coined in (Mitchell et al., 2009) to define the traditional IE approach where a single document is considered ant the goal is to distil all the information included and structure the resulting knowledge in a pre-defined ontology.

Information extraction was first studied in the Message Understanding Conference (MUC) (Beth, 1995; Grishman and Sundheim, 1996; Chinchor, 1998) and continued with the program Automatic Content Extraction (ACE) (Maynard et al., 2003) until 2008. These were the first quantitative evaluation of systems of this kind, and they were a major factor for the advance of the state of the art in automatic text processing.

Relation extraction is a subproblem of IE that consists on identify labelled semantic relations between entities. For example, in the sentence *John is married to Mary*, we can extract that there is a relation between two persons, and this relation is labelled as `spouse`.

More formally, a relation is an ordered tagged pair $r(e1, e2)$, where r is the relation name, or label, $e1$ is the target entity (i.e. the entity about we are querying) and $e2$ is the value, the entity that holds the relation with the target entity. A relation instance is a single assignment of entities and relation. In the past example, the relation instance would be $spouse(John, Mary)$.

Relation extraction is very popular, and it has many small competitions related. As an example, the DDI Extraction 2011 challenge[viii] is a Spanish competition that focuses in the biological relation extraction.

Some KBs were developed with information extraction techniques. Some of them are built extracting information from semi-structured text, such as Wikipedia infoboxes.

[vi] http://www.openlinksw.com/dataspace/dav/wiki/Main/VOSRDF
[vii] http://marmotta.apache.org/
[viii] http://labda.inf.uc3m.es/DDIExtraction2011/

These KBs have a precision around 95%. Some examples of these KBs are Yago (Suchanek et al., 2007), Kylin/KOG (Wu and Weld, 2007), and DBpedia (Auer et al., 2007). More recently, Wikidata (Vrandečić and Krötzsch, 2014) arose as a hybrid approach that assemble user and computer contributions in order to get the best of both worlds. Despite the good precision, these approaches depend on sources of semi-structured text, which is not as rich and abundant as natural language.

Regarding information extraction directly from text, most salient systems use bootstraping or other semi-supervised techniques. Bootstraping systems start with a set of seed patterns that are used to collect a set of instances from the document collection. Then, it will search other matches of these instances in different sentences that will be used to define new patterns. This process is executed iteratively to increase the recall of the system. However, each iteration introduces more errors, decreasing the precision. DIPRE (Brin, 1999) and SnowBall (Agichtein and Gravano, 2000) are some examples of iterative knowledge methods.

2.4.3 Macro-reading and Knowledge Base Population

Macro-reading (Mitchell et al., 2009) also relies on a pre-defined ontology to drive the extraction, and the task is actually to populate it, hence the name **Knowledge Base Population** (KBP). Macro-reading appeared as a solution to the decreasing performance of bootstraping systems. Macro-reading aims to extract information at the collection level, instead of at document level. The main idea is that it is not important to extract every piece of information from a single document, but that the salient facts are (probably) stated many times on a corpus with different rewordings, thus making easier the extraction task. This new perspective enabled the idea of coupling the identification of more than one relation at the same time, ensuring that the coupling of two relations is consistent with the ontology makes much more robust the ontology population.

As a result there was a shift of paradigm for knowledge acquisition. On parallel, from 2009, the Text Analysis Conference (TAC) took over the ACE in the evaluation of information extraction and proposed the Knowledge Base Population task (McNamee and Dang, 2009) was proposed in this conference. It may be seen as a combination between information extraction and question answering, where the task is harder because the information must be gathered across different documents. Most salient systems such as CUNY (Artiles et al., 2011) and Stanford (Surdeanu et al., 2011) use standard NLP tools such as tokenization, segmentation, named entity detection, coreference resolution and syntactic dependency parsing.

ReadTheWeb (Mitchell et al., 2009) is the first system based on the macro-Reading paradigm. Never Ending Language Learning (NELL) (Carlson et al., 2010) is a related system. It learns extractors for a set of predefined relations through distant supervision. To do so, it uses a bootstrapping approach, it learns extraction patterns from extraction instances, and then runs the new patterns to extract new instances. As in the case of micro-reading, bootstraping methods tend to degrade after few iterations, so it is common to use different heuristics to reduce the noise.

As with micro-reading, the use of a pre-defined schema facilitates the inference process, but limits the expressiveness and therefore macro-reading is still unable to represent all the knowledge required for language interpretation. In the following sections we review the major paradigms within the unsupervised acquisition systems: open information extraction and propositional knowledge.

2.4.4 Open Information Extraction

The **Open Information Extraction** (OIE) paradigm aims to extract relations from natural language in Web size scale. OIE supposes a new point of view over the traditional information extraction. Whereas information extraction generally learns an extractor for each target relation from labelled training examples, OIE aims to extract and cluster different phrases that express arbitrary relations. The main advantage of this paradigm is that it is completely unsupervised, as it is independent of any dictionary of relations or instances.

However it has some important drawbacks: often the knowledge extracted is incomplete and noisy. Therefore it is necessary to filter and validate the obtained data, complete missing data and deal with uncertainty in some degree. Structured KB capture knowledge in a pre-specified model and rely on *ad hoc* reasoning procedures that are simple, but provide precise inferences, whereas OIE uses propositions encoded with natural language to express facts about the world at the expenses of being less precise.

Generally, OIE systems extract tuples *Arg1, predicate, Arg2* in four steps:

1. Label a set of sentences using heuristics or distant supervision.
2. Learn a relation phrase extractor using some probabilistic model based on sequences, for instance, naïve Bayes, conditional random fields or Markov logic networks.
3. Identify two arguments connected by words in the path between them that express the relation.
4. Some systems do an assessment of the instances in base of the redundancy found.
5. Some systems try to cluster language variations of the same arguments and relational phrases.

To avoid using lexical features, the extractors are trained using POS and NP-chunk features from examples heuristically generated from the Penn Treebank.

DIRT (Lin and Pantel, 2001a) (Discovering Inference Rules from Text) automatically identifies rules of inference from paths in dependency trees. Basically, a dependency parser is applied to a text collection, and then, the different paths between the arguments are gathered and finally the similarity between different paths is calculated. DIRT has been used for textual entailment (Marsi et al., 2007) among others.

KnowItAll (Etzioni et al., 2004, 2005) can be considered the predecessor of the Open Information Extraction paradigm. It starts from a set of predefined relations, manually introduced by the user. Then it is able to extract information without labelled training examples using syntactic patterns. After the extraction phase, it refines the instances obtained using pointwise mutual information on information gathered on search engines. This system only uses a part of speech tagger and dispense with named entity recognizers or dependency parsers. However, it relies on several search engine queries. As a result, it is not scalable and depends on search engines.

TextRunner (Banko et al., 2007) can be considered the first Open Information Extraction system. The relation phrase extractor uses a Naive Bayes model with unlexicalized POS and NP-chunk features. The examples for the training are generated through heuristics from the Penn Treebank. In the labelling phase, TextRunner uses a parser to label a small set of training examples, restricted by three heuristics: there is a dependency chains between the arguments shorter than certain threshold, the paths between the arguments does not contain any sentence boundaries and the arguments are not just a pronoun. These examples are used to generate features to train a Naive Bayes classifier. Some examples of features include number of tokens in the relation, number of stopwords and part of speech tags. Then, the system makes a single pass over its corpus to identify entities and tag words with the part of speech. Each entity that is a suitable candidate is introduced in the classifier to label it as trustworthy or not. Finally, the tuples extracted are normalized and aggregated to assign a probability for each one. The experiments reported use a corpus of 9 million web pages, and claim to extract a total of 60.5 million tuples.

As stated in (Etzioni et al., 2011), the main drawbacks of this approach are the incoherent extractions and uninformative extractions. Incoherent extractions are relations that are not meaningful because the extractor chose the wrong words to describe it. Uninformative extractions are cases where the tuple, besides being informative, misses critical information.

Resolver (Yates and Etzioni, 2007) is an extension of TextRunner. Its goal is to merge different syntactical variances of the same meaning. To do so, they use the data collected by TextRunner and iteratively use cluster algorithms to merge clusters of co-referential names.

WOE (Wu and Weld, 2010) is another system that extracts its features from Wikipedia. They run the general schema where they start with a set of seeds, and then retrieve many examples through bootstrapping, to finally learn a set of patterns.

ReVerb (Etzioni et al., 2011) addresses the problem of incoherent and uninformative extractions. To do so, it uses shallow NLP to tag the words with its part of speech, and then apply a set of simple syntactic and semantic constraints defined by regular expressions. These techniques provided a significant advance compared to TextRunner and WOE.

OLLIE (Mausam et al., 2012) addresses two important flaws in the previous OIE systems. First, it deals with relations not mediated by verbs, mainly those that occur in nominal compounds. Second, it takes into account the context of the sentences to avoid extracting relations that are not asserted as factual. To do so, they start with the popular approach of starting with a set of seeds, in their case, 110.000 tuples and then build a bootstrapping set searching for these seeds in a Web corpus. Once they retrieve a set of sentences, they apply the Malt Dependency Parser (Nivre et al., 2006) and learn a set of syntactic and semantic patterns that encode different ways of expressing the relations. This process is called syntactic scope expansion. Then they apply what they call the context analysis component. This component detects when a sentence does not assert a fact, but only states a conditional truth or an attribution.

A conditional truth is a case where the sentence states that something is true if some condition happens. OLLIE uses the dependence relations to extract the condition and annotate it in an additional field. In the attribution case, OLLIE adds an extra field that indicates who said, suggested, believes, hopes or doubts the information extracted. To do so they train a classifier to assign a confidence to each extraction using a different training set of 1000 instances.

One of the most recent systems of OIE is CSD-IE (Bast and Haussmann, 2013). The main idea behind this system is to extract facts that semantically "belong together". This property is illustrated through an example: *Ruth Gabriel, daughter of the actress and writer Ana Maria Bueno, was born in San Fernando.* In this sentence, they aim to identify that the sentence contains two entities, each one with its own semantic classes and relations. Identify, for example, that *Ruth Gabriel* is a *writer* would be an error, even if those words are in a close context. They denote the process of identify the context of each entity as contextual sentence decomposition. This process is similar to the techniques used in (Mausam et al., 2012).

2.4.5 Propositional Knowledge Acquisition

In contrast with relational knowledge, propositional knowledge is not limited to extract binary relations. Instead, it is structured in **propositions**, which are a realizations of a predicate and its arguments.

Human languages express semantics through predicate-argument structures. A predicate-argument structure denotes the relation between the words that compose a sentence. In order to mimic the human way to represent knowledge, it is desirable that a meaning representation capture semantics with the same structure.

Propositional knowledge is the branch of research that connects this idea with propositional logic. In propositional knowledge, a predicate-argument structure found in text is captured in a tuple that is used as a proposition. The tuple contains the predicate and the arguments expressed in natural language. For instance, the sentence *"John sell a book"* expresses a predicate with the verb *sell* and a pair of arguments *sell* and a book that act as subject and object, respectively, and produce a proposition encoded in the tuple `{John - sell - book}`. These propositions can be used as pieces of knowledge that represent ground truth (or at least evidence towards trueness) and that provide a higher level of abstraction than syntactic trees. The main idea is that proposition stores provide evidence about how the world can be, that is, what is commonplace in the discourse domain.

Therefore, providing text annotated with syntactic dependencies, it is straightforward to extract propositions. This makes ideal the actual scenario where dependency parsers are able to process large amounts of text. The results are large proposition stores that encode meaning without domain restrictions.

This approach is more general, as it allows expressing n-ary relations by default. Also, it provides a more fine-grained representation, where a relation is not limited by an extracted tag, but it can be any word read in the text. The main drawback of propositions is their sparsity, as semantically similar propositions cannot be clustered without further processing.

Schubert (Schubert, 2002) argues that text contain background knowledge in form of assertions that can be exploited by processing large quantities of text. In this way, the focus is in finding redundancies rather than in refining the intra-document analysis. The goal is to extract general relationships from texts, instead of predetermined kinds of facts. For achieve this, they use information about the phrase structure and also compositional interpretive rules. With these tools, they build a large lexical semantics KB.

DART (Discovery and Aggregation of Relations in Text)(Clark and Harrison, 2009) is a work based on the previous idea that tries to exploit the redundancy of instances acquired from patterns in a corpora. In this project, they obtain world knowledge in a semi-formal notation. They have a database with 23 million propositions. The main contributions ahead Schubert's work are the kind of tuples extracted, the amount of data collected and a way to evaluate the results. Their method uses a hand-built parser called SAPIR, which extracts the phrase structure, and then extracts tuples from the parse tree. To test that Dart database was useful, it was used in a parsing test and in recognizing textual entailment.

Prismatic (Ferrucci et al., 2010) is also a KB extracted from large datasets. However the technique employs a more complex knowledge representation, as it is based on frames instead on binary relations or triplets. Besides using a dependency parser, it also uses a coreference resolver and an entity recognizer. The question answering system Watson uses Prismatic as one of the KB.

Peñas and Hovy (2010) suggest to differentiate classes and instances of those classes, and how to use them together with propositions to make explicit the implicit or missed information in the text. To test this system, they do a comparison with DART and TextRunner, and they find that they add new capabilities as management of instances, discovering new relations and being able to fit the KB to a specified domain. They also show that restrict the source collection to a single domain produces better results.

These kind of specific, large sources of knowledge have the potential to help many issues related to natural language understanding. However, how to use propositional knowledge is a research line on an early stage of development. Still, propositional knowledge has shown to be useful for NLP tasks such as textual entailment (Clark and Harrison, 2009), semantic enrichment (Peñas and Hovy, 2010; Peñas and Hovy, 2010) and question answering (Fan et al., 2012).

2.5 Textual Inferences

Despite the great effort in compile large knowledge databases, there are still many open research questions on how to take advantage of the information extracted, specifically, which tasks can be targeted and how can this knowledge be used.

Inference can be defined as the ability of a system to automatically combine meaning representation of the inputs with background knowledge. This capability includes the ability to generate new knowledge, or recover relevant missing background knowledge. In our case, we focus on textual inferences, since we limit both the meaning representations and the background knowledge to textual utterances and their interpretation. Roughly speaking, there are three kinds of inferences, deductive, inductive and abductive.

Deductive inference produce new information using some rules that ensure that the new information is true. Deductive inferences are often used in predicate logic, for example, given that *cats are a mammals* and *mammals are animals*, we can infer that *cats are animals.*

Inductive inference create a new explanatory general rule from a set of particular examples. For example, given that *seagulls*, *magpies* and *pigeons* are *birds*, we all of them *can fly*, we can infer that *birds can fly.* Unlike deductive inference, the new information is not guaranteed to be true, so typically inductive inference generates new rules with an associated probability.

Abductive inference finds the most plausible fact that imply an utterance. For example, if we know that *the grass is wet*, we can assume that *it is raining* with certain probability, although other explanations may be possible. Abductive inference can also use information of the world to modify the strength of the beliefs.

Deductive inferences are highly valuable because they ensure that the new knowledge is true. However, often they are not applicable to real tasks. Due to the ambiguous nature of language, most of the information inferred is abductive, that is, in ambiguous sentences the most plausible meaning is the one that we assume as correct, even when there are several other interpretations.

Many tasks related to language interpretation are heavily knowledge dependent. For example, those tasks which include inferring implicit information contained in the text. Here we provide some examples:

- Ellipses: Enjoy [reading] the book.
- Metonymy: They are drinking [the wine in] a bottle of wine.
- Compound nominals: The bite of a malaria [carrying] mosquito.
- Metaphor: Michael is [as strong as] a beast.
- Lexical ambiguity: Ana put all her money in the [financial] bank.
- Intensional adjectives: The former [and no longer] president has been travelling for eight weeks.

All these linguistic phenomena have been studied mostly independently, but, ultimately, they require to retrieve relevant knowledge in order to be interpreted. Although addressing them is out of the scope of this dissertation, we claim that our proposal is a first step towards a single generic methodology for interpretation that can tackle this problems using proposition stores. In the following sections we take a look to the most salient cases and other language interpretation tasks.

2.5.1 Type Coercion

A precise definition of type coercion is provided by (Pustejovsky, 1991): *"Type coercion is a semantic operation that converts an argument to the type that is expected by a function, where it would otherwise result in a type error"*

Recently it has been redefined as a task (Pustejovsky et al., 2010), where type coercion is *the task of characterizing the type of compositional operation that exist between a predicate and the argument that it selects. Specifically, the goal is to identify whether the type that a verb select is satisfied directly by the argument, or the argument must change type to satisfy the verb typing.*

This definition is based in that, when a predicate takes an argument, this argument may fit in that role only through an adjustment or coercion that makes it belong to the type that the predicate expects.

For example, the ellipsis *enjoy the book* actually means *enjoy reading the book*. Broadly speaking, type coercion would be to identify the accepted argument of *enjoy* would be an event like *reading* instead of an object like *book*, thus tagging the change as *event-for-object*.

A recent example of a work that address this issue is (Roberts and Harabagiu, 2011).

2.5.2 Logical Metonymy

According to the Oxford Dictionary, metonymy is *the substitution of the name of an attribute or adjunct for that of the thing meant, for example suit for business executive, or the turf for horse racing.*

This process is very frequently used by humans to optimize their communication, as it provides a fast way to translate concepts from a sender to a receiver. However, this use implies that systems are not able to retrieve the full meaning of a sentence. To do so, they have to convert the concept stated on the text for the original meaning.

From the point of view of natural language processing, a logical metonymy occurs when a logical argument (i.e. subpart) of a semantic type that is selected by some function denotes the semantic type itself (Pustejovsky, 1991).

In fact, metonymy is a process very related to type coercion. Regarding the example used before *enjoy the book*, one can understand that being readable is part of the semantics of the book, thus inferring that the natural way of enjoy a book is to read it.

Several approaches have been proposed to solve this issue (Fass, 1991; Utiyama et al., 2000; Lapata et al., 2003; Lapata and Lascarides, 2003).

2.5.3 Interpretation of Noun Compounds

Noun compounds are sequences of two or more nouns that act as a single noun (Downing, 1977). The semantic relation between its components is implicit, and this information may be important for understanding the context. For example, *malaria mosquito* can be interpreted as *malaria carrying mosquito*

The use of noun compounds is very frequent. However, a large portion of compounds have few repetitions. Therefore hand made-approaches, like dictionaries or ontologies, are unfeasible for interpretation.

There are two main approaches to model noun compounds: Semantic similarity and paraphrasing.

Systems that use semantic similarity try to describe noun compounds with statistical models trained with features mostly extracted from the context. For example,(Moldovan et al., 2004) define a set of semantic relations, including *temporal*, *topic* and *location*, and then try to classify unseen noun compounds into one of them. To do so, they use different learning models, including semantic scattering, decision trees and naive Bayes, using features based on the semantic classes of the nouns.

On the other hand, paraphrasing systems try to find words, mainly verbs and prepositions that can help to make explicit the implicit relations between the nouns (Nakov and Hearst, 2006, 2008; Peñas and Ovchinnikova, 2012). This task has recently grown in popularity which is shown in several SemEval tasks (Girju et al., 2007; Hendrickx et al., 2009; Butnariu et al., 2009; Jurgens et al., 2012; Hendrickx et al., 2013).

2.5.4 Implicit Semantic Role Labelling

Semantic Role Labelling systems deal with the problem of tagging the arguments of a predicate (See complete description in Section 2.3). However, this approach is limited in cases where the arguments are implicit in the sentence, specially in cases where they appear in a long distance context. As a result, predicates obtained with these systems are incomplete, and eventually can produce noisy predicates.

Implicit Semantic Role Labelling, also known as null-instantiation resolution, aims to recover these predicates. Whereas it has a long standing problem (Palmer et al., 1986), the popularity of this task has been rising recently partially due to subsequent tasks proposed at SemEval-2010 (Ruppenhofer et al., 2010) and SemEval-2013 (Kolomiyets et al., 2013), plus the availability of new knowledge resources.

However, this is a very hard challenge, and systems reported very low F1 measures (F1=0.0014) (Chen et al., 2010). Still, some recent work achieved significant advances (Laparra and Rigau, 2013).

2.5.5 Paraphrase Detection and Textual Entailment

The process where speakers reword a sentence with different syntactic and lexical variations is called paraphrasing. This phenomenon results in big difficulties for machines to assert that two phrases mean the same, that is, they are semantically equivalent. When this relation is not bidirectional, but unidirectional, this is called textual entailment. Between 2006 and 2013, this task was extensively studied in the Recognizing Textual Entailment Challenge (Dagan et al., 2006; Dzikovska et al., 2013), that defines it as: *"Given two text fragments called 'Text' and 'Hypothesis', Textual Entailment Recognition is the task of determining whether the meaning of the Hypothesis is entailed (can be inferred) from the Text"*

There is a long tradition of research in textual entailment. One newer approach, Paralex (Fader et al., 2013a) tries to take advantage of KBs to associate natural language patterns to database concepts, solving the problem of linking input strings with entities, relations and question patterns.

2.6 Conclusions

In this chapter we have reviewed three important areas for language interpretation: meaning representation, knowledge acquisition and textual inference. Although all these three topics have seen a large boost in the past years there is still a long way until machines are able to understand language, in part, because these areas remain as separate fields of research, but also because most of the development is devoted to specific sub-problems.

Typically, meaning representations are built from documents, and then are connected with structured knowledge bases (ontologies) by two means: either the meaning representation is enriched with information from the knowledge base, generally through entity linking, or the meaning representation is used to populate the knowledge base. These processes are very limited for the purposes of language interpretation. Meaning representations and knowledge representations should be closer in order to be able to mutually enrich each other.

At this point, manual approaches to acquire commonsense knowledge are considered insufficient and difficult to apply to language interpretation because they are very distant to actual realizations of text. The current trend is to automatically acquire knowledge from large amounts of text. Open Information Extraction arises as a paradigm that aims to acquire knowledge by representing concepts and relations through simple representations close to natural language, typically subject-relation-object triples. This information is found in ever-increasing text corpora, and gathered in background knowledge bases or proposition stores. These resources have shown to be useful in different applications such as question answering (Fader et al., 2014), textual entailment (Christensen et al., 2013), summarization (Soderland and Mausam, 2014), and others (Mausam, 2016).

Proposition stores suppose a middle ground between meaning representations and structured knowledge bases. Since propositions are acquired from meaning representations, their vocabulary is closer to natural language, while at the same time they are able to represent a different kind of knowledge than predefined ontologies. However, there are many open research questions on how to acquire and use this knowledge and how the different components relate to each other.

Regarding meaning representations for acquisition purposes, currently there are several semantic formalisms that are used to encode the contents of a text. Among them, graphs representations stand out as popular approach. Graph representations can be seen as an evolution from dependency parsers, where the aim is to build a tree with the syntactic dependencies of the elements of a sentence. By definition, trees are limited because every node is connected with a single parent and to the root node by exactly one directed path. This constraint is useful for a dependency analysis, but not for a semantic analysis, where nodes can act as arguments for many relations, therefore having more than one parent.

Besides, graphs provide extra useful capabilities, such as the availability of well known algorithms to traverse them or search for patterns. Besides, their modular structure makes easy to add extra information.

One specific realization of a graph representation is Abstract Meaning Representation, which captures who is doing what to whom in a sentence. However, still there is not an AMR parser with a comparable performance to standard dependency parsers. Many other graph representations have been proposed to improve document representation, but most of them are built at sentence level, thus they lack of cohesion between the elements on distant contexts.

As we have said, we focus on propositional knowledge acquisition. Current proposition extraction systems such as (Clark and Harrison, 2009; Fader et al., 2011; Gordon and Schubert, 2012) ignore some specific information in favour of acquiring general background knowledge. On the one hand, they do not address which named entities participate as an argument of a proposition. On the other hand, they do not generalize entities into their semantic classes and how such semantic classes participate in the propositions. Besides, the acquisition of propositions is limited to sentence-level.

To the best of our knowledge, there is no effort to gather propositions from an agglutinative semantic representation such as AMR. Besides, AMR parsers themselves are not appropriate for proposition extraction, in part for the computational cost, but also because AMR parses sentences instead of documents, thus missing some interesting cross-sentence information.

In order to perform textual inference, it is necessary to find a suitable knowledge representation that is able to encode as much knowledge as possible while at the same time is able to be manipulated by a machine. Often a knowledge representations reaches a solution that satisfies one requisite at the expense of the other. Roughly, highly structured representations are easy to understand, and they enable simple but powerful inferences, such as property propagation in ontologies. However, every piece of information that does not fit in the general schema is lost. For domains like web-size knowledge extraction, or common sense knowledge, build such a detailed schema has not been accomplished even when investing large amounts of effort. In addition, simple inferences often lack of the necessary granularity to deal with language interpretation. On the other hand, looser representations (such as natural language) can express more knowledge, however, automatic systems struggle to handle them to perform inference. As a result, the field has seen a shift from early heavily structured meaning representations into representations closer to natural language.

Summarizing, it seems reasonable to think that the way to make meaning representation, knowledge acquisition and textual inference converge is finding a way to express knowledge in a natural way, closer to meaning representations but also enabling textual inferences.

This thesis proposes a new paradigm that covers these three aspects and study how they influence each other. Figure 2.2 locates the four research areas related to using propositional knowledge for language interpretation that we explore.

1. First, we study how to map documents into a meaning representation for acquisition purposes.
2. Second, we study how to build a proposition store from the meaning representation. The last two challenges are devoted to study how propositional knowledge can be used to perform textual inference.
3. Third, we study how a proposition store can be used to improve the meaning representation, specifically, how can be used to improve syntactic parsing.

4. Finally, we study how can we bridge the gap between propositional knowledge and structured knowledge bases, more precisely, how to map propositions into linked data properties.

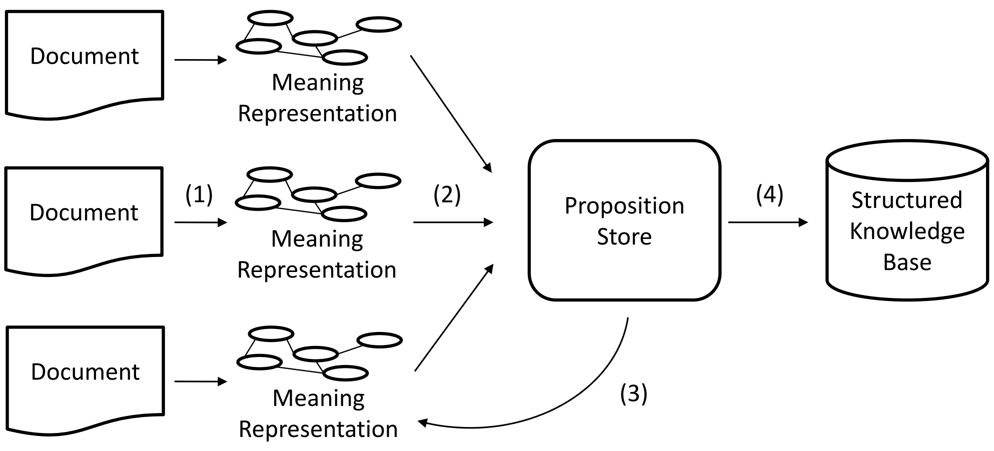


Figure 2.2: Proposal to use propositional knowledge as middle ground to relate meaning representations and structured knowledge bases. (1) Transforming a plain document into a meaning representation. (2) Extracting propositional knowledge from meaning representations. (3) Using propositional knowledge to correct a meaning representation. (4) Map propositional knowledge into structured knowledge.

Regarding the meaning representation study, we explore graphs, which are a compromise solution according to Hobbs criteria and are feasible to build from a standard dependency parsing making use of coreference links. In Chapter 3 we propose a graph based representation that aims to encode the semantic at document-level while at the same time simplifying and standardizing the dependencies so that classifiers could use new features that capture context that might be distant on text.

In (Peñas and Hovy, 2010) a Background Knowledge Base is created where named entities and classes are considered, but the acquisition is limited to a small domain and propositions are extracted at sentence-level. In Chapter 4 we extend this work in three directions: first, we extract propositions from documents represented as graphs; second, we increase the number of patterns captured by the propositions; and finally we increase the number and size of the processed corpora.

We illustrate the usage of the Proposition Stores in two different tasks that are knowledge intensive: syntactic and semantic parsing. For syntactic parsing, we focus on appositives, which are highly ambiguous dependencies where parsers commit mistakes due to the lack of semantic information. Appositives are useful for textual acquisition because they often convey a instance-class relation, but still there is a lack of dedicated literature, specially in comparison with other ambiguous structures such as PP-attachments. In Chapter 5 we study what kind of errors arise when parsing appositives and show how can be corrected using automatically acquired knowledge.

In the semantic parsing task, typically systems build a lexicon, that is, a dictionary where textual occurrences and canonical logical forms are related with an associated probability. There are many possibilities to build such a lexicon automatically, i.e. observation of system behaviour (Chen and Mooney, 2011), conversations (Artzi and Zettlemoyer, 2011), schema matching (Cai and Yates, 2013), questions (Poon, 2013) and especially question-answer pairs (Kwiatkowski et al., 2013; Berant et al., 2013; Berant and Liang, 2014; Fader et al., 2013b, 2014). A comparison between the aligning methods is difficult because it requires an evaluation on an extrinsic task that involves several components. In Chapter 6 we study the contribution of the knowledge acquisition on closing the gap between natural language and database predicates in a question answering over linked data task.

3

Graph Based Representation of Text

In this chapter we explain the characteristics of the representation that we use as the basis for the knowledge acquisition. This is a representation at document level, obtained from plain text to a graph representation through deep text processing techniques. We also measure the contribution of using this representation for feature generation in an information extraction task.

Our feeling is that an effective characterization of knowledge can result in a real understanding system in the not too distant future.

Roger Schank and Robert Abelson, 1975

Contents

3.1 Introduction

Traditionally, semantics were expressed either on conceptual models such as ontologies or frames, with a limited expressive power, or in logic frameworks such as first order logic, which are more expressive but present scalability problems as well as being hard to be created from text (Martin and Jurafsky, 2000).

In the current stage of NLP there is an ongoing effort to design a general purpose meaning representation able to capture as much semantics as possible directly from an open domain text (Oepen et al., 2014; Koller, 2015). Ideally, a sizable corpus of this meaning representation would be useful in subsequent NLP tasks such as textual entailment, question answering, machine translation and others.

The challenge of generating automatically this representation is still unsolved. The first approach is to define an annotation schema, use it to annotate a large enough corpora, and use them to learn how to map natural language into it. The most recent and salient effort of the community to create a general purpose meaning representation in this line is Abstract Meaning Representation (Banarescu et al., 2012). AMR is a set of guidelines to annotate English sentences with their logical meanings, with the final purpose of creating a Treebank that can be used to work in statistical NLU. Although recently some AMR parsers have been published (Artzi et al., 2015; Pust et al., 2015; Peng et al., 2015; Vanderwende et al., 2015; Wang et al., 2015; Flanigan et al., 2014), these are on early stages of development and their performance is still far from syntactic parsers.

On the other hand, a meaning representation can be built from plain text by intensive use of linguistic processing tools. Traditionally, deep processing techniques such as syntactic parsing and coreference resolution were considered computationally too expensive to be applied to large corpora. However, current methods are more mature and optimized, while at the same time computational power is much cheaper, so this task becomes possible.

Our goal is to generate a document-level conceptual representation derived directly from text that captures as much semantics as we can in a machine-readable format. We build this representation by combining existing language processing tools with our own semantic enrichment processes. Let's consider the following example:

(3.1) David's wife, Julia, is celebrating her birthday. She was born in September 1979.

A syntactic dependency parser would produce two parse trees, one for each sentence. Each word become a node in the tree and these words are related by syntactic dependencies. However, relations implied by the text, such as *Julia being born in*

September are not represented. We want to map the basic parse tree into a more meaningful representation that relates all the elements present on the text. Figure 3.1 illustrates the expected graph for the example. It shows a compact representation where all the coreferent mentions of the entity *Julia* are clustered, and as a result it is directly related to the events *bear* and *celebrate*.

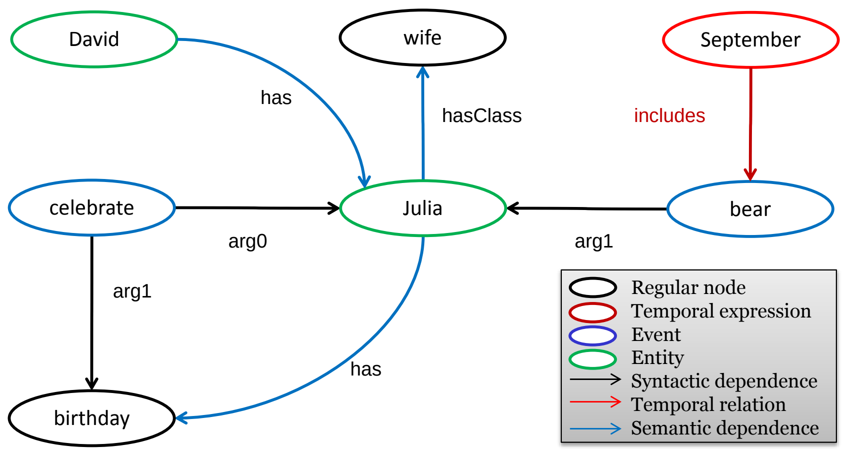


Figure 3.1: Graph-based representation intended of the example document *"David's wife, Julia, is celebrating her birthday. She was born in September 1979."* .

Moreover, consider the following examples

(3.2) Julia is David's wife. She is celebrating her birthday. Julia was born in September 1979.

(3.3) David has a wife, Julia, who was born in September 1979. Julia is celebrating her birthday.

Sentences 3.2 and 3.3 are basically semantically equivalent to 3.1. However, their syntactic parse trees are different. We aim to avoid different syntactic realizations in order to produce similar graphical representations for sentences with the same meaning.

Therefore, the research questions tackled in this chapter are:

- **Research Question 1.1:** *What are the steps to build a graph-based representation from text at the whole document level using off-the-shelf tools?*
- **Research Question 1.2:** *What kind of lexical, syntactic and semantic information can be expressed by these graphs?*
- **Research Question 1.3:** *What other features do the representation provide?*
- **Research Question 1.4:** *What features of an information extraction classifier are affected by the graph representation?*

- **Research Question 1.5:** *Which is the performance of a classifier trained with those features in a task of automatic relation extraction?*
- **Research Question 1.6:** *In the same task, once the graphs are generated, what is the effect of the new semantic information?*

Parts of this chapter have been published in (Cabaleiro and Peñas, 2012), (Garrido et al., 2011) and (Garrido et al., 2012). In particular, research questions 1.5 and 1.6 were addressed in a joint work led by Guillermo Garrido.

This chapter is divided into the following sections: First, in Section 3.2 we motivate the need of a graph-based representation and explain the elements of the graphs. Then, in Section 3.3 we propose a methodology to build the representation and in Section 3.4 we show the results of applying the methodology to diverse textual collections. Section 3.5 shows an extrinsic evaluation of the graph-based representation in the Temporal Slot Filling task. Finally in Section 3.6 we summarize the conclusions of this chapter.

3.2 Graph Based Representation at Document Level

In order to interpret language, computers have to replicate the human ability to build conceptual models of the meaning of the text. Semantic analysis, which is building such a model from plain text, has been a long standing research line, and a central problem in artificial intelligence.

The Meaning Text Theory (Mel'cuk and Polguère, 1987) proposes that interpretation is a mapping from subsequent layers of representation: phonetic representation, morphological representation, syntactic representation and semantic representation. Given that syntactic parsing is a very developed research field, it seems natural to try to map from syntactic tree parsers into meaning representations.

Trees are useful for syntactic parsing because they can be efficiently derived from grammars. However, they impose hard constraints as being rooted and acyclic, which are problematic for meaning representations. In general, a semantic representation do not require a root node, and more importantly, elements of the semantic representation should be allowed to participate as arguments of an unbounded number of predicates.

For these reasons, a more general graph based representation is more convenient to represent meaning. Crucially, graphs can be enriched with linguistic information that is not expressed in the text but it is useful or even necessary for language interpretation. Besides, graphs are a well known mathematical formalism, easy to traverse by automatic means. They also can alleviate the problems caused by distant context in text by relating far items in text with direct syntactic or semantic edges.

Some graph representations have been proposed for document representation (Banarescu et al., 2012; Martins and Almeida, 2014). The intuition is that these representations are capable of capturing the relations of the elements present in the original text. However, they often are built at sentence-level, so they lack of cohesion between the elements on distant contexts. Our hypothesis is that a graph representation at document level will provide a better solution at least in some cases.

Therefore, the main goal of this chapter is to study the generation of a representation able to express the whole structure of one document in a simplified, condensed way, capturing lexical, syntactic and semantic relations.

We show in the next chapters that graphs provide a suitable framework to search for syntactic-semantic patterns for knowledge acquisition purposes.

As a subproduct of this work, we feed information extraction systems with this representation instead of just plain text in order to use new features that allow classifiers to capture context that might be distant in text.

3.2.1 Initial Representation

In order to build such graph-based representation, we take a bottom-up approximation. Following Ferrucci et al. (2010) we denominate **deep processing of text** to those techniques that are commonly used to transform a plain text to a representation which includes structured syntactic and semantic features in a machine readable format that allow further inference. It can be seen as finding a computational representation closer to a logical representation.

Roughly, there are two general ways to process text: shallow processing and deep processing. Shallow processing, i.e. statistic approaches based on bag-of-words, are popular methods to characterize text. Shallow processing can be fast and computationally cheap, but the result is a vector representation that can only express the surface semantics of sentences.

In contrast with shallow processing, deep linguistic processing provides a rich, expressive, and structured representation that captures long-distance dependencies (Baldwin et al., 2007). The major drawback of deep linguistic processing is the high computational effort that it takes, specially for syntactic parsing. As a result, many processes cannot be applied to large text corpora. However, recent advances in processing techniques, such as dependency analysis in linear time, make affordable deep processing of text with the current resources.

In our case, we use off-the-shelf tools to perform deep processing of text to structure documents into a graph-based representation that we denominate **initial representation**.

Formally, each document D is represented by a graph, $G_D = (N, E)$, comprising a set of nodes N and a set of directed edges E.

3.2.1.1 Nodes

A node $n \in N$ represents a unit of information, generally a word, except in two cases: a multi-word named entity or a verb with its auxiliaries.

Every node is tagged with a descriptor. A descriptor is a representative string. For the nodes that are not entities the string is composed by the lemmas of the words in the node. Otherwise we choose the string just as it is found on the text. Nodes are divided in the next types:

- Regular Nodes: Every node without an special class is considered a regular node. A node contains the following set of attributes: Words, lemmas and morphosyntactic tags. These attributes are included in every type of node.
- Events: Events are verbs or nouns that describe an action or a process. Besides the regular attributes, events are also annotated with time, aspect and polarity. A detailed explanation of the events can be found in (Saurí et al., 2005).
- Temporal Expressions: Used to identify words or multi-words that refer to a concrete moment or to a temporal period. It has a normalized temporal value according to the TimeX3 standard.
- Named Entities: Entities recognized in text. Annotated with entity type, for example *organization*, *location*, *person*, etc.

3.2.1.2 Edges

An edge $e \in E$ is a directed relation between nodes, that is $e = (n_1, n_2 \in NxN : n_1 \neq n_2)$. An edge can represent three types of relations:

- Syntactic: Syntactic edges represent a syntactic dependence between two nodes. The labels of these edges correspond to the Stanford dependencies tags (De Marneffe and Manning, 2008).
- Coreference: A coreference edge indicates that two nodes are mentions of the same discourse referent.
- Temporal: Temporal edges represent a temporal relation between an event and a temporal expression. Temporal relations belong to the next types: *before*, *after*, *within*, *throughout*, *beginning* and *ending*.

3.2.2 Enriched Representation

The **enriched representation** aims to convert the initial representation (i.e the syntactic tree) into a semantic graph representation. To do so we take advantage of coreference, the linguistic relation that is established between two or more expressions that refer to the same entity. We use the term *discourse referent* coined by (Karttunen, 1968) to refer to the set of all of these expressions for an entity. We denote the process of creating a discourse referent by grouping the mentions as *collapsing*. This procedure is similar to file cards used by (Heim, 1983) or baskets in (Recasens, 2010).

Collapsing creates a big graph with many different grammatical relations between its components. This would produce features with high sparsity, which is not desirable for automatic learning techniques or for knowledge extraction. Our solution is to simplify the morphosyntactic relations with a naïve semantic role labeling. We normalize different expressions that are semantically equal. We consider grammatical voice, genitives expressed in different ways, nominal compounds, etc. Section 3.3.2 details the normalization process.

Besides, collapsing provokes the merging of nodes tagged with different attributes. These attributes can reinforce an evidence or supply with extra data. However, occasionally contradictory data can be grouped. The criteria to solve inconsistencies is detailed in Section 3.3.2.

We transform the graph by defining a set of operations that are executed when some prerequisites are met. Under these circumstances, and taking into account the nature of the graphs that do not have a beginning or an ending by definition, the standard procedural/object-oriented computation conveys two main problems:

First, the control flow is not linear. Depending on each document and graph, different operations would be executed and in different order. To code every possible control flow in procedural programming is a waste of effort and makes the code very redundant, making it difficult to maintain. Second, adding new operations is expensive, because they have to be included directly on the code. Besides, a sideline effect is that coding is more difficult to understand, because each operation is not coded independently. To solve these problems we opt to use declarative programming.

Formally, graphs with initial representation $G_D = (N, E)$ are transformed to create enriched representation graphs $EG_D = (R, A)$, characterized by a set of nodes that we denominate discourse referents R and a set of edges that we denominate enriched edges A. The next sections describes these elements:

3.2.2.1 Discourse Referents

A discourse referent $r \in R$ is built when a set of nodes $n_0, \ldots, n_k \in N$ are related by coreferences, thus creating a new set of discourse referents R. Discourse referents are in fact the new nodes of the graph in the enriched representation.

Discourse referents keep the types of the original nodes: Regular discourse referents, events, temporal expressions and named entities. Discourse referents also keep the attributes of their nodes (i.e. the nodes collapsed into the same discourse referent). However, collapsing may arise some inconsistencies, that we detail in Section 3.3.2.

3.2.2.2 Enriched Edges

Besides the types of edges of the initial representation, we add extra edges with semantic information. As a result, we get three types of edges:

- Syntactic: Syntactic dependencies simplified through syntactic patterns. We add the tags *arg0*, *arg1* and *arg2*, which roughly correspond to subject, direct object and indirect object, but considering the different voices, subordinate clauses, etc.
- Temporal: Same edges that are in the initial representation
- Semantic: Indicates that there is a semantic relation between two nodes. We distinguish between three subspecified semantic tags: *is*, *has* and *hasClass*. These tags aim to group every dependence that denotes a copulative relation, a genitive relation that expresses a notion of possession or an instance-class relation, respectively.

3.3 Methodology

In this section we describe the methodology to build the graph-based representation from a plain text. We divide the process into two steps, which yield the two different representations: the initial representation and the enriched representation. These representations are used as source to obtain features for an information extraction classifier. Their contribution is measured in Section 3.5.

Figure 3.2 shows the steps to produce the initial and the collapsed representation from a plain text. The left side represent the processes to create the initial representation, which is an aggregation of text processing tools. Then, the right side corresponds to the enriched representation, which refines the initial representation, mainly by adding extra semantic information and normalizing syntactic relations. Next sections details these processes.

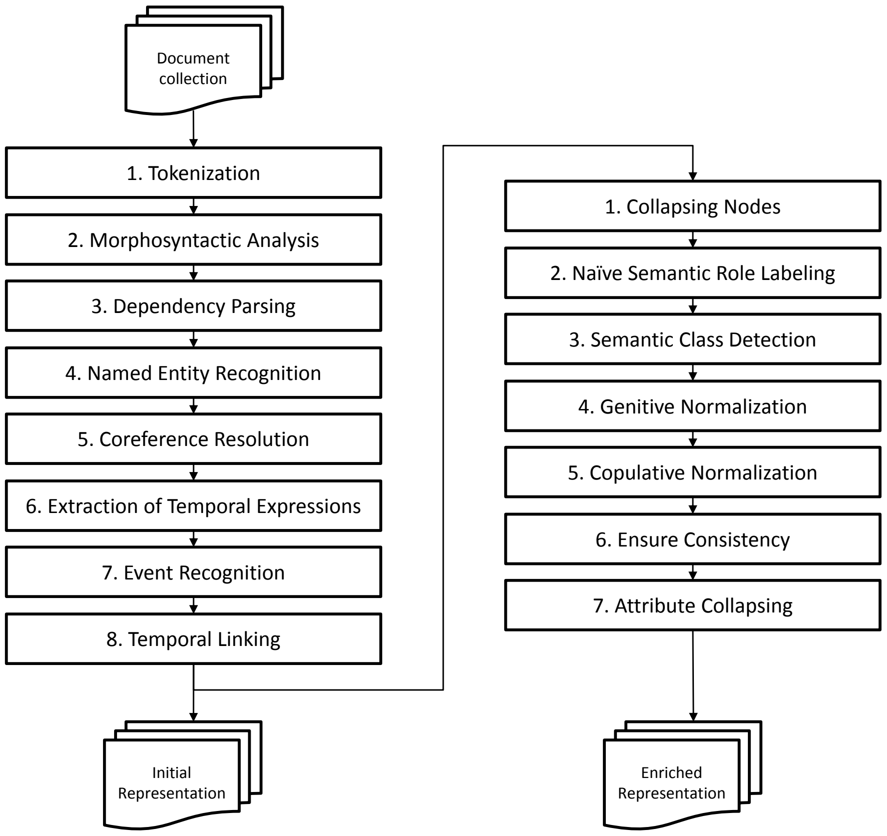

Document collection
1. Tokenization
2. Morphosyntactic Analysis
3. Dependency Parsing
4. Named Entity Recognition
5. Coreference Resolution
6. Extraction of Temporal Expressions
7. Event Recognition
8. Temporal Linking
Initial Representation
1. Collapsing Nodes
2. Naïve Semantic Role Labeling
3. Semantic Class Detection
4. Genitive Normalization
5. Copulative Normalization
6. Ensure Consistency
7. Attribute Collapsing
Enriched Representation

Figure 3.2: Methodology for the tranformation from a plain text into the initial and the enriched representations.

3.3.1 Initial Representation Generation

In order to build the initial representation, we depart from a syntactic dependency tree whose nodes are annotated with morphological information, and with edges annotated with syntactic information, coreferences and temporal relations. Specifically, we perform the deep text processing methods:

1. Tokenization: A tokenizer first divides documents into sentences, evaluating if a sentence ending character such as '.' is acting as a delimiter. Then, it splits sentences into tokens. In English, a token roughly corresponds to a word with some exceptions regarding punctuation. We use the Stanford Tokenizer [i].

[i]http://nlp.stanford.edu/software/tokenizer.shtml

2. Morphosyntactic analysis: This step performs part of speech tagging and lemmatization. Part of speech tagging, or POS tagging, is one of the most common and studied text enriching processes. POS taggers assign a grammatical descriptor to each word in a sentence (Voutilainen, 2003).

   In many cases, POS taggers also provide inflectional and lexico-semantic information, such as the distinction between common and proper nouns, or verb tenses. However, in most languages each word can play different roles depending on the semantics of the sentence. Moreover, POS tagging is language dependent, so it is common to train language-specific POS taggers.

   There are two common approaches to develop POS taggers, one is rule-based, where some hand-defined or automatically acquired rules are used to tag words, and the other is stochastic, where statistical techniques are used. The latter achieves the best results, with reports over 97% accuracy (Toutanova et al., 2003; Shen et al., 2007; Hajič et al., 2009).

   A lemmatizer extracts the lemma for a word. This is the canonical form of a word, which is the particular form that, chosen by convention, represents the lexeme. It is useful because it allows grouping all the different inflected forms of a word under the same representation.

   In some occasions, it is important to disambiguate the word according to the context, and in this case the task is not trivial because it requires understanding the context that surrounds the word.

   For both steps we use the Stanford Log-linear Part-Of-Speech Tagger (Toutanova et al., 2003).

3. Dependency parsing: A dependency parser structures a sentence into a dependency tree. In a dependency tree, nodes represent tokens, and are related through directed labelled edges that represent syntactic dependencies. We use the Stanford PCFG Parser (Klein and Manning, 2003). Dependency parsing and more generally syntactic parsing is discussed in detail in Chapter 2.3.2. The dependency labels are described in (De Marneffe et al., 2006).

4. Named entity recognition: Named Entity Recognition (NER) is, as defined at the Sixth Message Understanding Conference (Grishman and Sundheim, 1996), the task to identify the names of all the people, organizations and geographic locations in a text. In this context, to identify is equivalent to determine the words that belong to the entity and label such words with the category, or named entity type. This is frequently extended to a few other types such as times, currencies and percentage expressions, but it is possible to find systems that distinguish between many more categories (Sekine and Nobata, 2004). Besides, named entities are tagged with the named entity type. NER is mostly considered

as a solved task and most recent research is centered around specific domains or topics such as Twitter (Derczynski et al., 2015).

There are two main approaches to solve this task, handcrafted rule-based algorithms and machine-learning approaches. The latter achieve better results, and are commonly used, for example in (Finkel et al., 2005; Ratinov and Roth, 2009).

We use the Stanford Named Entity Recognizer (Finkel et al., 2005).

5. Coreference resolution: Coreference relations occur between different mentions that refer to the same real world entity. These mentions can be written with different linguistic expressions, so relate them is not a trivial process.

   However, knowing that two mentions refer to the same entity is important to correctly comprehend a text. This makes coreference resolution a very wide research field, but despite all efforts, the performance of the coreference resolvers in the state of the art is around 60%-70% F1 measure (Clark and Manning, 2015).

   In the first place, most of the coreference research tries to apply heuristic approaches, but from the 90s most of the systems use machine learning techniques (Raghunathan et al., 2010; Lee et al., 2011a). Recently, Lee et al. (2013) showed how a constrained set of rules could achieve the state of the art.

   We use the Stanford Multi-pass Sieve Coreference Resolution (Raghunathan et al., 2010).

6. Extraction of temporal expressions: Temporal expressions are lexical elements used to denote temporal semantics such a date or an interval. In order to automatically map temporal expressions into meaning representation it is necessary to determine the temporal expression meaning. This process is not trivial as temporal expressions may be ambiguous, and the same temporal expression can be conveyed by different lexical expressions. Moreover, some temporal expressions require require a reference time to be computed (i.e. today, tomorrow, yesterday, next month, etc.).

   In order to perform this task some systems were developed to locate and normalize temporal expressions. We use the TempEx tagger (Mani and Wilson, 2000), GUTime tagger (Verhagen et al., 2005) or SUTime (Chang and Manning, 2012). All of them are rule-based systems.

   The most accepted standard for temporal representation is TIMEX3, contained within the TimeML language (Pustejovsky et al., 2003). The previous mentioned systems use this standard.

7. Event recognition: Event recognition is a close field of named entity recognition. In this case, the goal is to find events in the text, which consists in states, actions and processes together with their properties. It is a very important

task, especially for question-answering systems, as events are frequently a key part of the input question, and in general for tasks related to natural language understanding.

Traditional event recognition systems use a pre-defined set of relation patterns (Brill et al., 2002; Hovy et al., 2002), but there are some other approaches that combine linguistic and statistic knowledge to improve the results (Saurí et al., 2005; Bethard and Martin, 2006). For our representation we use Evita (Saurí et al., 2005).

8. Temporal linking: In many NLP tasks it is important to know which events occur after or before a concrete date (or other event). Here we denominate Temporal linking to the task of relate events to temporal expressions and infer the relative order of the events that appear in the text.

   Unlike other systems discussed here, there is not much research on this area. Some of the most notable systems rely on hand-written rules, and their performance is not as good as desirable. The Tarsqi Toolkit (Verhagen and Pustejovsky, 2008) includes some tools to perform temporal linking, for example GUTenLINK (Mani et al., 2003) anchors events to time expressions and S2T creates temporal links from modal relations in the text. We use these tools in the representation.

These tools are grouped in two NLP frameworks. Steps 1-5 use the Stanford CoreNLP package (Manning et al., 2014), while steps 6-8 apply the Tarsqi Toolkit (Verhagen et al., 2005). In order to ensure consistency, we manually fed the Tarsqi Toolkit with the tokenization produced by the Stanford CoreNLP. Moreover, we limit the length of the documents to 10000 characters to avoid memory problems. After these processes, we obtain a syntactic tree annotated with extra temporal information.

3.3.2 Enriched Representation Generation

As we have explained in Section 3.2.2.2, the standard procedural computation is not desirable for graphs processing. Therefore, we opt for declarative programming. In declarative programming, only the initial state is given (i.e. the graph) and the operations are defined with a set of rules. We use JESS[ii], a Java rule engine. Due to processing requisites (e.g. compute direct objects before indirect objects), we define priorities in the rules. In JESS, this concept is denoted as salience, being -1 the lowest priority and 1 the highest.

We follow the next steps:

[ii]http://www.jessrules.com

1. Collapsing nodes: First we group all nodes related by coreference dependencies into a discourse referent. We refer to this process as collapsing. Given an initial graph $G_D = (N, E)$, the goal is to generate a enriched graph $EG_D = (R, A)$. A discourse referent $r \subset \{n_0, \ldots, n_k \in N : k \geq 1\}$ is composed by a set of nodes $n_0, \ldots, n_k$ related by coreference dependencies. Given a second discourse referent $r' \subset \{n'_0, \ldots, n'_{k'} \in N\}$, we tie both referents with an enriched edge $a'(r, r')$ if any of their nodes were tied, that is: $\exists e(n_i, n'_{i'}) \rightarrow \exists a'(r, r')$

2. Naïve semantic role labeling: Then we group syntactic dependencies into three new dependencies, *arg0*, *arg1* and *arg2* that correspond to subject, direct object and indirect object. Table 3.1 shows the rules used. These rules target active and passive subjects and objects, plus clausal complements. The objective is to normalize the syntactic dependencies so that sentences with the same meaning yield the same representation.

Naïve semantic role labeling - arg0, arg1 and arg2		
Input	Output	Salience
$V \rightarrow nsubj \rightarrow N$	$V \rightarrow arg0 \rightarrow N$	0
$V \rightarrow xsubj \rightarrow N$	$V \rightarrow arg0 \rightarrow N$	0
$V \rightarrow csubj \rightarrow N$	$V \rightarrow arg0 \rightarrow N$	0
$V \rightarrow agent \rightarrow N$	$V \rightarrow arg0 \rightarrow N$	0
$V \rightarrow nsubjpass \rightarrow N_1$, $V \rightarrow arg1 \rightarrow N_2$	$V \rightarrow arg2 \rightarrow N_1$, $V \rightarrow arg1 \rightarrow N_2$	-1
$V \rightarrow nsubjpass \rightarrow N$	$V \rightarrow arg1 \rightarrow N$	0
$V \rightarrow dobj \rightarrow N$	$V \rightarrow arg1 \rightarrow N$	0
$V \rightarrow iobj \rightarrow N$	$V \rightarrow arg2 \rightarrow N$	0
$N \rightarrow partmod \rightarrow V$	$V \rightarrow arg1 \rightarrow N$	0
$V \rightarrow xcomp \rightarrow N$	$V \rightarrow arg1 \rightarrow N$	0
$V \rightarrow ccomp \rightarrow N$	$V \rightarrow arg1 \rightarrow N$	0
$V \rightarrow xcomp \rightarrow N_1$, $V \rightarrow arg1 \rightarrow N_2$	$V \rightarrow arg2 \rightarrow N_1$, $V \rightarrow arg1 \rightarrow N_2$	-1
$V \rightarrow ccomp \rightarrow N_1$, $V \rightarrow arg1 \rightarrow N_2$	$V \rightarrow arg2 \rightarrow N_1$, $V \rightarrow arg1 \rightarrow N_2$	-1

Table 3.1: Patterns used to perform naïve semantic role labeling. Each entry corresponds to a dependency $governor -\ dependency -\ dependent$. N_i corresponds to a noun and $ne(N)$ denotes that N is a named entity and V is a verb. Salience refers to the priority of the rule, being -1 the lowest priority and 1 the highest.

3. Semantic class detection: We add edges that denote a semantic class-instance relation between common nouns and named entities. This semantic relation is denoted as *hasClass*, and is produced through a set of patterns that detect lexical structures with genitives, nominal compounds and appositives. Table 3.2 shows the full list of rules that drive the transformation. These patterns target linguistic phenomena such as appositives, abbreviations, and nominal

compounds that often are used to express that an entity belongs to a semantic class, similar to (Hearst, 1992).

Semantic class detection - hasClass		
Input	Output	Salience
$ne(N_1), N_1 \rightarrow nn \rightarrow N_2$	$N_1 \rightarrow hasClass \rightarrow N_2$	0
$ne(N_1), N_1 \rightarrow appos \rightarrow N_2$	$N_1 \rightarrow hasClass \rightarrow N_2$	0
$ne(N_1), N_1 \rightarrow abbrev \rightarrow N_2$	$N_1 \rightarrow hasClass \rightarrow N_2$	0
$ne(N_2), N_1 \rightarrow appos \rightarrow N_2$	$N_2 \rightarrow hasClass \rightarrow N_1$	0
$ne(N_2), N_1 \rightarrow abbrev \rightarrow N_2$	$N_2 \rightarrow hasClass \rightarrow N_1$	0
$ne(N_1), N_1 \rightarrow nsubj \rightarrow N_2$	$N_1 \rightarrow hasClass \rightarrow N_2$	0
$ne(N_2), N_1 \rightarrow prep\_such\_as \rightarrow N_2$	$N_2 \rightarrow hasClass \rightarrow N_1$	0
$ne(N_2), N_1 \rightarrow prep\_like \rightarrow N_2$	$N_2 \rightarrow hasClass \rightarrow N_1$	0

Table 3.2: Patterns used to perform semantic class detection. Each entry corresponds to a dependency *governor – dependency – dependent*. N_i corresponds to a noun and $NE(N)$ denotes that N is a named entity. Salience refers to the priority of the rule, being -1 the lowest priority and 1 the highest.

4. Genitive normalization: Our next step is to normalize genitive relations. Our purpose is to select genitives that denote some notion of possession. Table 3.3 shows the patterns that we use to transform structures like nominal compounds and straight possessives into the semantic relation *has*.

Genitive Normalization - has		
Input	Output	Salience
$ne(N_2), N_1 \rightarrow nn \rightarrow N_2$	$N_1 \rightarrow has \rightarrow N_2$	0
$ne(N_2), N_1 \rightarrow poss \rightarrow N_2$	$N_2 \rightarrow has \rightarrow N_1$	0
$N_1 \rightarrow poss \rightarrow N_2$	$N_2 \rightarrow has \rightarrow N_1$	-1
$ne(N_2), N_1 \rightarrow prep\_of \rightarrow N_2$	$N_2 \rightarrow has \rightarrow N_1$	0
$ne(N_2), N_1 \rightarrow nsubj \rightarrow N_2$	$N_2 \rightarrow has \rightarrow N_1$	0

Table 3.3: Patterns used to normalize genitives. Each entry corresponds to a dependency *governor – dependency – dependent*. N_i corresponds to a noun and $ne(N)$ denotes that N is a named entity. Salience refers to the priority of the rule, being -1 the lowest priority and 1 the highest.

5. Copulative normalization: Copulas are used in language to link subjects and predicates. For our purpose, copulatives are an intermediate node that does not convey a strong meaning, and result in extra distance for other nodes that should be closer in the representation. Therefore we simplify the relations that involve a copulative by transforming these nodes into relations. Table 3.4 shows the relevant rule.

Copulative Normalization - is		
Input	Output	Salience
$N_1 \rightarrow nsubj \rightarrow N_2$, $N_1 \rightarrow cop \rightarrow N_3$	$N_2 \rightarrow is \rightarrow N_1$	0

Table 3.4: Pattern used to normalize copulatives. Each entry corresponds to a dependency *governor – dependency – dependent*. N_i corresponds to a noun. Salience refers to the priority of the rule, being -1 the lowest priority and 1 the highest.

6. Ensure consistency: The process of collapsing may produce some errors. Table 3.5 shows the rules that we have designed to correct four kind of mistakes. First, we consider inappropriate coreferences, such as the ones between and entity and its semantic class. Second, we consider a problem where an entity was incorrectly attached to a temporal expression, instead of the relevant event. Third, we correct instances where a noun is related to another noun through its semantic class, instead of directly by the noun. Finally, we correct cases where a nominal compound involving several common nouns and a named entity would only assign the closest noun as semantic class of the named entity.

7. Attribute collapsing: Collapsing may cause that one referent contains several annotations of the same type. We denote as **attribute collapsing** to the process of choosing the final attributes of the discourse referents.

   We pick the representative named entity type and descriptor for the discourse referents. With descriptors, the objective is to find the string that better characterizes the node. To do so, we take the longest string of every original node. In the future, it would be a better idea to create a knowledge base with every coocurrence of the descriptors, and from it to select the right one, similar to disambiguation systems.

   Regarding the morphosyntactic tag, we assign to every named entity the value N and to every event V, while for the rest of the words we maintain the same of the original node. This is a simple approximation that takes advantage of the fact that named entities and events are the nodes that can have coreference relations. With this decision we seek to normalize the nodes that usually contain the most important information.

   In some cases, due to the collapsing a discourse referent can possess more than one attribute of one type. In these cases, we opt for the next solutions:

   - Regular Discourse Referents: Like regular nodes, regular discourse referents are defined exclusively with the attributes that are common to every discourse referent. These attributes are slightly different from the ones of the nodes. Discourse referents retain descriptors and morphosyntactic tags, but lose lemmas and words.

- Events: If several events collapse in a single discourse referent we keep all the attributes, i.e. time, aspect and polarity.
- Temporal Expressions: When several temporal expressions coincide on a discourse referent, we store all the attributes.
- Named Entities: Often, several named entities with contradictory types are grouped in a discourse referent. In these cases, we retain only the most

Coreference		
Input	Output	Salience
$N_1 \rightarrow rcmod \rightarrow N_2$, $N_2 \rightarrow argX \rightarrow N_3, N_3 = wh*, that$	$N_1 \rightarrow coreference \rightarrow N_3$, $N_2 \rightarrow argX \rightarrow N_3$	0
$N_1 \rightarrow hasClass \rightarrow N_2$, $N_3 \rightarrow coreference \rightarrow N_2$	$N_1 \rightarrow hasClass \rightarrow N_2$	1
$N_1 \rightarrow hasClass \rightarrow N_2$, $N_2 \rightarrow coreference \rightarrow N_3$	$N_1 \rightarrow hasClass \rightarrow N_2$	1
$N_1 \rightarrow is \rightarrow N_2$, $N_3 \rightarrow coreference \rightarrow N_2$	$N_1 \rightarrow is \rightarrow N_2$	1
$N_1 \rightarrow is \rightarrow N_2$, $N_2 \rightarrow coreference \rightarrow N_3$	$N_1 \rightarrow is \rightarrow N_2$	1
Temporal expressions		
Input	Output	Salience
$V \rightarrow X \rightarrow N$, $N \rightarrow Y \rightarrow DATE$	$V \rightarrow X \rightarrow N$, $V \rightarrow Y \rightarrow DATE$	0
Semantic class governor		
Input	Output	Salience
$N_1 \rightarrow hasClass \rightarrow N_2$, $N_3 \rightarrow X \rightarrow N_2$	$N_1 \rightarrow hasClass \rightarrow N_2$, $N_3 \rightarrow X \rightarrow N_1$	-1
hasClass		
Input	Output	Salience
$ne(N_1), N_1 \rightarrow nn \rightarrow N_2$, $N_1 \rightarrow nn \rightarrow N_3$	$N_1 \rightarrow nn \rightarrow N_2$, $N_2 \rightarrow nn \rightarrow N_3$	1

Table 3.5: Patterns used to ensure the consistency of the enriched graphs. Each entry corresponds to a dependency $governor - dependency - dependent$. N_i corresponds to a common name, $ne(N)$ denotes that N is a named entity and V is a verb. X and Y refer to any kind of dependency. Salience refers to the priority of the rule, being -1 the lowest priority and 1 the highest.

frequent types. As with the descriptors, this process could be improved by storing every coocurrence in a database and using it as a feedback for the type assignment. *Persons* can be also tagged with gender and age.

3.4 Application to Textual Collections

This section details the sources of documents for applying the representation method, the output format intended and the resulting graphs obtained.

3.4.1 Datasets

We applied our graph-based representation to texts from diverse corpora, both in size and in domain. The following list details the used corpora ordered by number of documents.

- AIDS: A collection of 28,863 articles of the medical domain of AIDS.
- NYTFootball: A collection of 33,886 New York Times articles about US football.
- Webtext: The English subset of 1 million documents of the TAC 2012 KBP Source Corpus Additions Web Documents (LDC2012E23). This is a collection of web documents extracted from the GALE web corpus and added to the TAC KBP Source Corpus in 2012.
- KBP: The TAC 2012 KBP Source Corpus (LDC2012E22) is a collection of about 1.7 million documents established at the KBP 2010. It is composed by newswire documents, web documents, and a small subset of other documents all written in English.
- WebClue09: We create a corpus of 4.2 million documents that contain single sentences in English with two or more links to entities. We explain the purpose of this corpus in Chapter 6. The WebClue09 Dataset (Callan et al., 2009) contains about 100 million web pages in ten languages, with about 50% of the documents written in English. Web pages were collected in 2009 as a part of the Lemur Project. We use the Freebase Annotation extension provided by Google where named entities are linked to their Freebase id (Gabrilovich et al., 2013).
- Gigaword: The English Gigaword Fifth Edition (Parker et al., 2011) is a collection of newswire text data in English that includes four sources of news: The Agence France Presse English Service, The Associated Press Worldstream English Service,

The New York Times Newswire Service and The Xinhua News Agency English Service. In total it contains near 10 million documents.

3.4.2 Results

We produce two different outputs. The first output is a JSON-like representation, where we annotate the relevant attributes for each node and relation. We have used this representation for the relation extraction task (see Section 3.5).

For the second output we use DOT[iii], a standard graph description language. It is compatible with many tools for graph visualization such as Graphviz[iv], Canviz[v] or Gephi[vi].

Table 3.6 reports the number of documents processed for each corpus, both for the initial and the enriched representation.

Corpus	Documents	Graphs	%
AIDS	28,863	26,065	90.31
NYTFootball	33,886	32,746	96.64
Webtext	996,960	831,400	83.39
KBP	1,774,494	1,754,248	98.86
WebClue09	4,256,350	4,255,644	99.98
Gigaword	9,876,086	4,745,191	48.05
Total	16,966,639	11,645,294	68.64

Table 3.6: Number of documents of each corpus and number of documents converted into graphs.

As a matter of example, Figure 3.3 shows the transformation from the initial representation to the enriched representation for the running example. As it can be seen, the enriched version is a more compact meaning representation for the initial input.

[iii] http://www.graphviz.org/doc/info/lang.html
[iv] http://www.graphviz.org/
[v] http://code.google.com/p/canviz/
[vi] https://gephi.org/

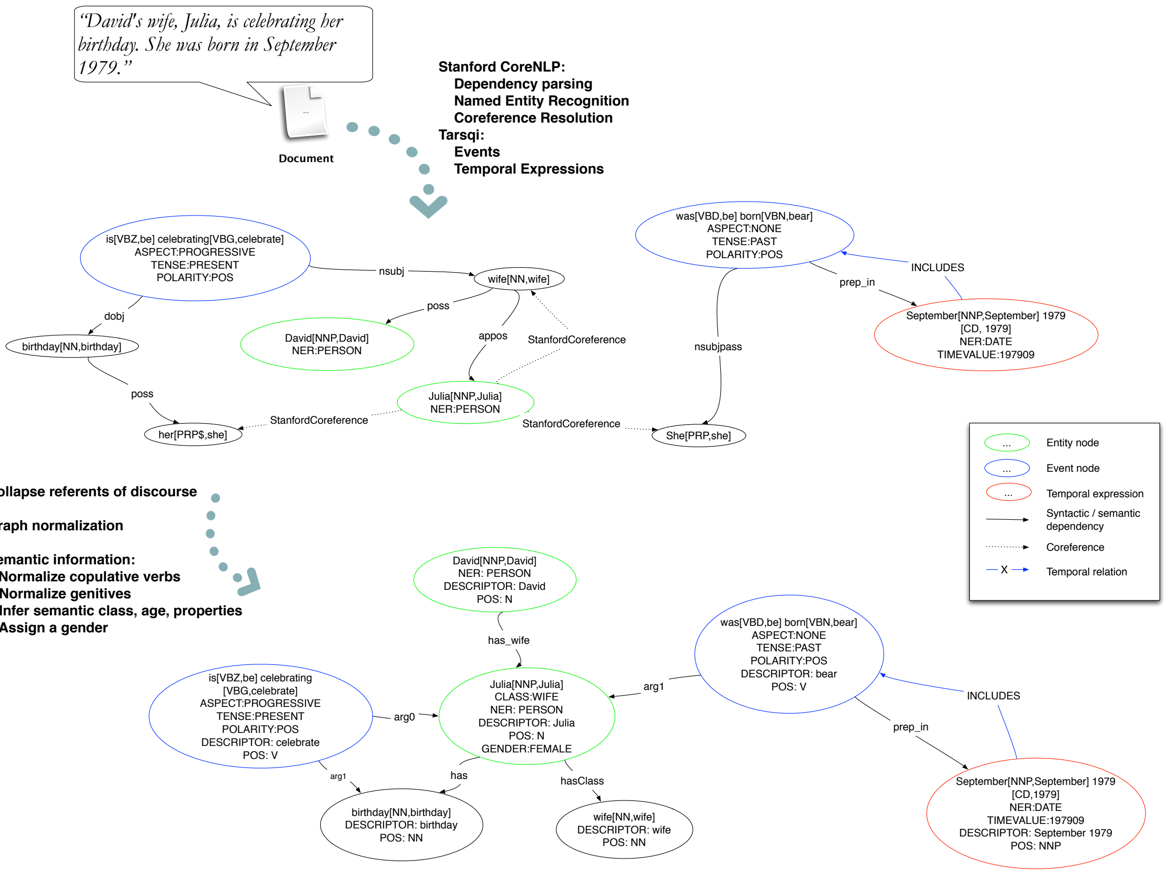

"David's wife, Julia, is celebrating her birthday. She was born in September 1979."
Document
Stanford CoreNLP:
Dependency parsing
Named Entity Recognition
Coreference Resolution
Tarsqi:
Events
Temporal Expressions
is[VBZ,be] celebrating[VBG,celebrate] ASPECT:PROGRESSIVE TENSE:PRESENT POLARITY:POS
nsubj
wife[NN,wife]
dobj
birthday[NN,birthday]
poss
David[NNP,David] NER:PERSON
appos
StanfordCoreference
Julia[NNP,Julia] NER:PERSON
poss
her[PRP$,she]
StanfordCoreference
StanfordCoreference
She[PRP,she]
nsubjpass
was[VBD,be] born[VBN,bear] ASPECT:NONE TENSE:PAST POLARITY:POS
prep_in
INCLUDES
September[NNP,September] 1979 [CD, 1979] NER:DATE TIMEVALUE:197909
Entity node
Event node
Temporal expression
Syntactic / semantic dependency
Coreference
Temporal relation
Collapse referents of discourse
Graph normalization
Semantic information:
Normalize copulative verbs
Normalize genitives
Infer semantic class, age, properties
Assign a gender
David[NNP,David] NER: PERSON DESCRIPTOR: David POS: N
has_wife
is[VBZ,be] celebrating [VBG,celebrate] ASPECT:PROGRESSIVE TENSE:PRESENT POLARITY:POS DESCRIPTOR: celebrate POS: V
arg0
Julia[NNP,Julia] CLASS:WIFE NER: PERSON DESCRIPTOR: Julia POS: N GENDER:FEMALE
arg1
was[VBD,be] born[VBN,bear] ASPECT:NONE TENSE:PAST POLARITY:POS DESCRIPTOR: bear POS: V
INCLUDES
prep_in
September[NNP,September] 1979 [CD,1979] NER:DATE TIMEVALUE:197909 DESCRIPTOR: September 1979 POS: NNP
arg1
has
hasClass
birthday[NN,birthday] DESCRIPTOR: birthday POS: NN
wife[NN,wife] DESCRIPTOR: wife POS: NN

Figure 3.3: Transformation from initial representation, G_D, to enriched representation, G_C, for the example document: "David's wife, Julia, is celebrating her birthday. She was born in September 1979".

3.5 Extrinsic Evaluation on Information Extraction

This chapter describes how we took advantage of the graphical representation for the Temporal Slot Filling task in the TAC Knowledge Base Population evaluation.

3.5.1 The Temporal Slot Filling Task

The main objective of the transformation of the documents in graphs is to improve the performance of the systems that actually use the representation. To test this goal, we perform an extrinsic evaluation in the 2011 Slot Filling task of the Knowledge Base Population track (Ji et al., 2011).

The Slot Filling task is composed by two subtasks: Regular Slot Filling (RSF) and Temporal Slot Filling (TSF). The goal of RSF is, given a set of entities and a set of slots, extract from a corpus of millions of documents the correct values for each combination of entity and slot. This is, for a set of queries composed by the pair *(entity, attribute)* the systems have to answer with a tuple *(entity, attribute, value)*. For instance, for the query *(Barack Obama, spouse)* the answer should be *(Barack Obama, spouse, Michelle Obama)*.

The correct answer to a slot may consist in a list of values. Only the right values are scored, and the redundant answers are ignored.

Entities are divided in two categories, *person* and *organization*. Each category has its own set of slots (See Table 3.7).

Person	Organization
per:alternate_names	org:alternate_names
per:date_of_birth	org:political/religious_affiliation
per:age	org:top_members/employees
per:country_of_birth	org:number_of_employees/members
per:stateorprovince_of_birth	org:members
per:city_of_birth	org:member_of
per:origin	org:subsidiaries
per:date_of_death	org:parents
per:country_of_death	org:founded_by
per:stateorprovince_of_death	org:founded
per:city_of_death	org:dissolved
per:cause_of_death	org:country_of_headquarters
per:stateorprovinces_of_residence	org:city_of_headquarters
per:cities_of_residence	org:shareholders
per:schools_attended	org:website
per:title	
per:member_of	
per:employee_of	
per:religion	
per:spouse	
per:children	
per:parents	
per:siblings	
per:other_familiy	
per:charges	

Table 3.7: Slot names for the two entity types in the RSF task

The organizers distribute a knowledge database built from the Wikipedia Infoboxes. It contains noisy information about entities, values and relations, useful to train the systems. Moreover The TAC 2012 KBP Source Corpus, a 1.7 million documents collection, is provided as solutions search space. It contains documents from different sources, such as newswire, web and others.

The TSF adds the temporal component. In this subtask, the tuples *(entity, relation, value)* have to be anchored in temporal interval. A fuzzy temporal interval is defined as a tuple of four values (t_1, t_2, t_3, t_4), that denote that the slot started in a point between t_1 and t_2, and it ended between t_3 and t_4. If the value of t_1 or t_3 is not defined it means that the value is $-\infty$, while for t_2 or t_4 is $+\infty$.

Furthermore, the number of target slots is reduced for this subtask. Table 3.8 shows the TSF slots.

Person	Organization
per:spouse	org:top_employees/members
per:title	
per:employee_of	
per:member_of	
per:cities_of_residence	
per:stateorprovinces_of_residence	
per:countries_of_residence	

Table 3.8: Slot names for the two entity types in the TSF task

The organization provides with training data annotated with temporal anchors, and also intermediate local information regarding temporal constraints. This information consists on a temporal expression located on a document and the temporal relation (See Table 3.9) between the slot value and the temporal expression.

3.5.2 Experimental Design

In order to answer our research questions we participated in the RSF and TSF tasks (Garrido et al., 2011). We used a single multi-class classifier and a battery of binary classifiers for each task respectively. Regardless, in each task we follow the next steps:

1. First we retrieve the relevant documents given for the given queries, up to 100 documents per query.
2. Then, we transform the documents into the initial and enriched representation

3. Within the graphs, we search for the node or nodes that match the query entity.
4. Finally, each of the nodes at 10 or less length from the node is a candidate to be the slot value. With this information we produce an unlabelled example *(entity node, value node, document)*, which is processed to the classifier to label it as either positive for any of the slot type or with the negative label.

Furthermore, the classifier takes advantage of the representation to extract the features to characterize each example. Table 3.10 shows the features used in RSF. X stands both for *entity* and *value*.

TSF uses three extra features detailed in Table 3.11. These are Verb features, which are generated from the verbs, V, identified in the path between *entity* and *value*. Features provided by our representation are underlined.

We submitted two different runs to evaluate the performance of the enrichment and collapsing. Both follow the process described, but the first one uses the initial representation, whereas the second uses the enriched representation.

3.5.3 Results

Table 3.12 shows the results of the Regular Slot Filling Task, compared with the manual annotation (LDC) and the top teams in the competition. Our system clearly underperforms compared with the other systems. These results were expected as many

Relation	Role of temporal expression	**Example**
Beginning	the start time for the slot value	Rob joined GE in 1999
Ending	the end time for the slot value	Rob left GE in 1999
Beg_and_end	the slot value is true exactly for the specified time	Rob was named linguist of the month for June 1999
Within	the slot value is true for at least some portion of the specified time	Rob worked for GE in 1999
Throughout	the slot value is true for all of the specified time	Rob commuted to work from his home in Denver for all of the 1999
Before_start	a moment before the start time for the slot value	In 1999, before Rob joined GE, ...
After_end	a moment after the end time for the slot value	By 1999 Rob had already left GE

Table 3.9: Temporal relations in the TSF training data

Feature family	Feature name	Description
Syntactic dependency	path	path between *entity* and *value* in the sentence [represented with the unigrams and bigrams of dependency labels, POS tags and NE tags]
Placeholders	X-annotation	NE annotations for the sentence fragment X
	X-pos	Part-of-speech annotations for the sentence fragment X
Lexical context	X-gov	Governor of X in the dependency path
	X-mod	Modifiers of X in the dependency path
Properties	X-has_age	X is a NE, and we have identified it has an age attribute.
	X-has_class-C	X is a NE, and we have identified it has a class C.
	X-has_property-P	X is a NE, and it has a property P
	X-has-Y	X is a NE, and it is in a relationship to-have with another NE, Y
	X-is-Y	X is a NE, and it is in a relationship to-be with another NE, Y
	X-gender-G	X is a NE, and it has gender G

Table 3.10: Features included in the model for RSF. The underline denotes a unique feature extracted from the representation.

teams have mature systems whereas our system is in an early stage of development. However, it is notable how results improve using the enriched representation.

Table 3.13 shows the general results of the Temporal Slot Filling Task. It can be observed that the performance of the system is similar to other systems on the state of the art, and achieving the highest precision among all the participants. Moreover, despite the low recall, the system gets the third best F_1 measure.

This implies that the graphical representation is a promising chance in the task of

Feature family	Feature name	Description
Properties of the verb V	V-tense	Tense of the verb V in the path.
	V-aspect	Aspect of the verb V in the path.
	V-polarity	Polarity (positive or negative) of the verb V in the path.

Table 3.11: Features included in the model for TSF. The underline denotes a unique feature extracted from the representation.

	LDC	Top-1 Team	Top-2 Team	Median Team	Initial	Enriched
Precision	0.8618	0.3502	0.4917	0.1031	**0.0259**	**0.0343**
Recall	0.7259	0.2550	0.1259	0.1650	**0.0455**	**0.0529**
F1	0.7880	0.2951	0.2005	0.1269	**0.0330**	**0.0416**

Table 3.12: Regular Slot Filling task results

automatic relation extraction, because, despite being in the first steps of development, it allows to compete with the systems on the state of the art.

System	Precision	Recall	F1
BLENDER2	0.1789	0.3030	0.2250
BLENDER1	0.1796	0.2942	0.2231
BLENDER3	0.1744	0.2976	0.2199
IIRG1	0.2457	0.1194	0.1607
Initial	**0.2996**	**0.0703**	**0.1139**
Enriched	**0.2596**	**0.0609**	**0.0986**
Stanford 12	0.0233	0.1680	0.0409
Stanford 11	0.0238	0.1453	0.0408
USFD20112	0.0152	0.0070	0.0096
USFD20113	0.0079	0.0014	0.0024

Table 3.13: Final results on the Temporal Slot Filling task

The system is made by several components pipelined: Information retrieval, document representation, semisupervised learning, relation extraction and in the case of TSF, temporal anchoring. This kind of system is very sensitive to error propagation, and therefore is interesting to study precision and recall taking into account the maximum bound imposed by the information retrieval component.

Table 3.14 shows the results in the Temporal Slot Filling task until the relation extraction phase. It contains the precision and recall taking into account the number of tuples *(entity, attribute, value)* successfully obtained.

Representation	Initial	Enriched
Recall	0.08	0.08
Precision	0.42	0.45
F1	0.14	0.14

Table 3.14: Temporal Slot Filling task results until the relation extraction phase

Data shows that both precision and recall obtain very similar results in both types of representations, being slightly better in the enriched graphs case.

A manual inspection of the results shows that the generated graphs contain many mistakes. After the participation in the task we have identified many programming mistakes that possibly worsen the results, especially in the enriched representation. However, the enriched graphs represent a slight improvement against the initial graphs.

3.6 Conclusions

In this chapter we presented a method to transform documents into a document level representation consisting in semantically enriched morphosyntactic graphs. We have performed an extrinsic evaluation in the regular and temporal slot filling task. This representation is designed to facilitate future downstream NLP tasks such as feature extraction, knowledge acquisition.

We use simple techniques in every step because our goal force us to group heterogeneous information. Still, the representation supposes a promising start and provides a large margin for improvement.

Moreover, since the representation system is based on interchangeable external tools, we have a flexible and modular application that facilitates future changes.

Regarding our research questions:

1. **Research Question 1.1:** *What are the steps to build a graph-based representation from text at the whole document level using off-the-shelf tools?*

   In this chapter we have shown how to build a graph-based representation combining existing language processing tools with our own semantic enrichment processes. First we build the initial representation, which corresponds to a syntactic tree for each sentence in the document, annotated with extra information such as POS, named entity types, temporal relations and coreference. Then, we build the enriched representation using a declarative procedure that on the one hand collapses nodes into discourse referents taking advantage of the coreference relations and on the other hand refines the resulting graphs with a rule-based system that employs four types of rules: naïve semantic role labelling, semantic class detection, genitive normalization, copulative normalization, ensure consistency and attribute collapsing.

2. **Research Question 1.2:** *What kind of lexical, syntactic and semantic information can be expressed by these graphs?*

The graph-based representation contains standard lexical information such as the words lemma, POS and named entity type, events annotated with tense, aspect and polarity and temporal relations between entities and temporal expressions. Discourse referents also include a representative descriptor. Moreover graphs encode syntactic information condensed in the *arg0*, *arg1* and *arg2* relations, that express a notion of subject, direct object and indirect object respectively. We add extra semantic information including semantic class-instance relations *hasClass* and genitive (possessive) *has* and copulative relations *is*. These relations are obtained by identifying and tagging dependencies that convey the same semantics.

3. **Research Question 1.3:** *What other features do the representation provide?*

   The main unique features are provided by the extra semantic information. Specifically, we have used in RSF the following unique features: *X-has_age*, *X-has_class-C*, *X-has_property-P*, *X-has-Y*, *X-is-Y* and *X-gender-G*. In TSF we also added the following features related to event detection; *V-tense*, *V-aspect* and *V-polarity*.

4. **Research Question 1.4:** *What features of an information extraction classifier are affected by the graph representation?*

   The collapsed representation enables the propagation of information across the graph. For instance, pronouns participating as arguments of a predicate may be replaced by a named entity related by a coreference relation, and, as a result, path features are shortened. The naïve role labelling and the genitive/copulative/class-instance normalization aim to group dependencies whose semantics are equivalent, thus features extracted should be less sparse and therefore learning should be easier.

5. **Research Question 1.5:** *Which is the performance of a classifier trained with those features in a task of automatic relation extraction?*

   Slot filling is a challenge out of the scope of this work. We only run a preliminary system to compare the effect of both representations. Results show that the graph-based representation is promising for feature extraction for classifiers in the automatic relation extraction task. The many opportunities of improvement of the representation leads us to think that future versions can achieve better results.

6. **Research Question 1.6:** *In the same task, once the graphs are generated, what is the effect of the new semantic information?*

   We have got slightly better results using the enriched graphs. With these data we can say that the enrichment phase is useful for the automatic relation extraction task, as the balance between the gain of collapsing the graphs and the loss of the added errors introduced by the process is positive.

4

Automatic Capture of Propositional Knowledge

This chapter is devoted to general-purpose knowledge acquisition. Our approach extracts propositions at document level from large corpora of documents represented with the previously presented graph based representation. These propositions express background knowledge that are the base for further textual inferences.

If knowledge can create problems, it is not through ignorance that we can solve them

Isaac Asimov

Contents

4.1 Knowledge Capture

Achieving human intelligence in machines would require the acquisition of human-levels of knowledge. In other words, we need to provide large amounts of knowledge to machines as a prerequisite for enabling understanding. Presumably it involves several types of knowledge, such as factual knowledge (e.g. Barack Obama is married with Michelle Obama), procedural knowledge (a cooking recipe) or mathematical knowledge ($a + b = b + a$) among a large list. In our case, we are concerned with **background knowledge**, the knowledge that any ordinary person is expected to know. In order to be useful for machines, this knowledge has to be fine-grained and extensive which makes it hard to obtain. This problem is known as the acquisition bottleneck, and it has been a long term problem in the AI field. Specifically, we focus on the knowledge needed to guide systems for NLP tasks. For example, we want to know that *a quarterback throwing a pass* is an assertion more probable than *a quarterback swimming fast*, and study how this background knowledge is valuable in NLP task such as dependency parsing, word sense disambiguation or question answering.

Current computational capabilities plus the explosion of the textual resources in the Web have advocated researchers to tackle this task with automatic techniques that aim to extract knowledge from large sources of text. Classical approaches are based on Information Extraction (IE), where knowledge is tailed to a pre-specified template or ontology. However pre-specifying the target relations poses two main problems: first, develop an IE system requires a significant effort to learn the mapping to the target relations and the patterns to extract them; and second, IE is too coarse to be able to capture the full complexity of natural language. Both problems get worse as the target domain grows.

Addressing this issue, Banko et al. (2007) introduced Open Information Extraction (OIE) as a paradigm for knowledge acquisition. It uses surface patterns to extract an unbounded number of relations which are further refined with a probabilistic model. Even with the subsequent improvements over the original work, OIE is still limited: most of the OIE systems only extract verb-based relations, obviating other relations like nominal, possessive and copulative. However, for our purposes, the main drawback of OIE is that it is generally good acquiring facts (e.g. `<Steve Walsh - throw - pass>`) but not powerful enough to generalize this knowledge (e.g. *"C - throw - pass"* where C is a probability distribution for entity classes.).

In turn, **proposition extraction** aims to acquire general knowledge following the ideas of (Schubert, 2002), who hypothesized:

"[there is] a largely untapped source of general knowledge in texts, lying at a level beneath the explicit assertional content. This knowledge consists of relationships implied to be possible in the world, or, under certain conditions, implied to be normal or commonplace in the world."

Using his own example, the sentence *"He entered the house through the open door"* suggests that it is possible for males to enter in houses, that houses have doors that can be opened, and so on. This knowledge should be useful for tasks related to language interpretation.

Some examples of salient propositions extraction systems include (Clark and Harrison, 2009; Fader et al., 2011; Gordon and Schubert, 2012). Instead of binary relations, these systems aim to extract predicate structures with a variable number of arguments. Although proposition extraction is devoted to extract background knowledge, these systems do not generalize neither entities into their semantic classes nor how such semantic classes participate in the propositions. Besides, they limit the acquisition of propositions to sentence-level. This implies that information available in the text as a result of coreference resolution or pronominal anaphora is excluded from the propositions. Peñas and Hovy (2010) create a Background Knowledge Base where named entities and classes are considered. However, the acquisition is limited to a small domain and propositions are extracted at sentence-level. We extend this work in three directions: first, we extract propositions from documents represented as graphs; second, we increase the number of patterns that capture propositions; and finally we increase the number and size of the processed corpora.

In this context, we formulate the following research questions:

- **Research Question 2.1:** *Is it feasible to build knowledge bases from documents represented as graphs? What does this kind of representations provide?*
- **Research Question 2.2:** *What problems arise from an automatically generation of propositional knowledge bases?*

The application to Named Entity Disambiguation has been published in (Barrena et al., 2014).

This chapter is structured as follows. Section 4.2 presents our method to build a proposition store, with the particularities of semantic classes and typed propositions explained in Sections 4.3 and 4.4 respectively. Section 4.5 shows the methodology and implementation details. The resulting proposition stores are presented in Section 4.6. We evaluate a subset of the proposition stores in a named entity disambiguation task in Section 4.7 and we finish showing the conclusions in Section 4.8.

4.2 Proposition Stores

For our purposes, a **proposition** is identified as a tuple of words that fit in a predefined syntactic pattern or structure commonly use to express assertions, i.e. predicate structures. Each tuple describes an occurrence in the world, i.e. the corpus. From a propositional logic point of view, it can be seen as a statement that is either true or false.

The first step for proposition extraction is to annotate text with the syntactic dependencies. Previous systems use dependency parsers SAPIR (Clark and Harrison, 2009) or Stanford Parser (Peñas and Hovy, 2010). In our case, we use the enriched graph-based representation presented in Chapter 3.

Our graphs provide several advantages. Since the dependencies are normalized, a simple set of patterns can extract most of the information conveyed in the text. At the same time, some sentences with the same meaning produce the same propositions, even if they are syntactically different, for example, active-passive voice. Thus, Proposition Stores contain less noise by having less variability of propositions and higher frequencies. The next advantage is that some implicit relations such as the ones conveyed by semantic classes or genitives are explicitly represented. For example, the apposition present in the sentence *"David supports the team of his wife, Julia"* is directly expressed as `<Julia - hasClass - wife>`. Finally, since the named entity information is propagated through coreferences, patterns can capture information beyond the limits of a sentence.

Propositions are acquired through syntactic patterns that cover three types of linguistic realizations: dependencies between nouns (noun-noun compounds and possessives), dependencies between nouns and verbs (predicate-argument structures) and prepositions with two nouns as arguments. Thus, the main types of propositions are governed by:

- Verb (V): A verb is used to denote the relation between arguments. In order to reduce sparseness in the aggregation step, we use the lemma instead of the inflected form of the verb and we dispense with auxiliaries.

  We assume that sentences are constituted by predicates and their arguments, where a predicate expresses the relations between the arguments. We consider three types of arguments, subjects, objects and complements. For simplicity, indirect objects are included as complements.

- Preposition (P): A preposition can act as a relation between two nouns.

- Genitives (has): Grammatical relation that marks a noun as modifier of another noun. Genitives have many semantic interpretations, including possessor-possessee (John's house), temporal (yesterday's party), part-hole (John's leg) and kinship (John's brother).
- Semantic class (hasClass): Denotes a relation between a semantic class and an entity.

Note that the `has` and `hasClass` relations are represented with an edge in the graph based representation.

Arguments are divided in the following types:

- Noun (N): Since we aim to retrieve both general knowledge and specific knowledge about instances, we allow arguments to be named entities, common nouns or pronouns (i.e. `<Steve Walsh - throw - pass>`, `<player - throw - pass>` and `<he - throw - pass>`, respectively). Pronouns include personal pronouns, possessive pronouns and interrogative pronouns. Note that collapsed nodes may have several realizations for an entity. This is particularly common in named entities grouped with pronouns. For our propositions, we select a canonical descriptor of the node, according to the rules described in Section 3.3.2.
- Preposition (P): A combination of preposition and noun PN denotes a complement of a proposition. For simplicity, an indirect object is denoted also with PN, but in this case the preposition is omitted.
- Semantic Class (C): A semantic class is a noun that acts as a category of a named entity. We explain the semantic class acquisition in Section 4.3.

We denote a proposition with a concatenation of its elements. For example, a NVNPN proposition is an instantiation of <Subject - Verb - Object - Preposition - Complement>. For readability, we attach the preposition to the verb in the form <Subject - Verb:Preposition - Object - Complement>. For example, the sentence *"John write a book about science"* yields the proposition `<John - write:about - book - science>`. Table 4.1 shows the patterns that we consider. The most notable differences compared with previous work (Clark and Harrison, 2009; Peñas and Hovy, 2010) are the addition of the genitive relations and the omission of relations that involve quantifiers, adjectives or adverbs.

The aim is to select highly productive patterns to acquire a large number of propositions. However, we do not pretend to obtain every proposition expressed in the collection. What is important is to acquire a representative and sufficient number of propositions to evaluate the plausibility of new text occurrences.

Consequently, some patterns target partial information on purpose, for example, in the sentence *the quarterback throws a ball*, we can capture the proposition `<quarterback - throws>`, obviating some information in the process.

Pattern	Example	Proposition
NV	*As DNA replicates mistakes in the replication occur.*	`<DNA - replicate>`
NVN	*Four Rooms was released by Miramax in December, 1995.*	`<Miramax - release - Four Rooms>`
NVPN	*An opportunity has arisen for a payroll clerk to join this legal firm.*	`<opportunity - arise:for - clerk>`
VNPN	*Research is undertaken on leading-edge management issues through membership surveys.*	`<research - undertaken:on - issue>`
NVNPN	*Don Knotts won five Emmys as Barney Fife.*	`<Don Knotts - win:as - Emmys - Barney Five>`
NPN	*Frank Welker was the voice of Megatron.*	`<voice - of - Megatron>`
NHasN	*It is the policy of the Department to continue to ensure so far as is reasonably practicable.*	`<Department - has - policy>`

Table 4.1: Syntactic patterns used to extract propositions from the graphical representation. NVNPN is equivalent to Subject - Verb - Direct Object - Preposition - Complement

Also, in sentences with a coordinate structure (e.g. several subjects or complements), we acquire every possible combination of the arguments. For example, the sentence *John and Mary visit the zoo and the museum* produces the propositions `<John - visit - zoo>`, `<John - visit - museum>`, `<Mary - visit - zoo>` and `<Mary - visit - museum>`.

4.3 Semantic Class Propositions

Semantic classes are a popular topic of research in natural language. Determining that *wife* is a class of *person*, whereas *Spanish* is a class of either *person*, *organization* or *location*, is a very valuable knowledge for many tasks, including information extraction, question answering and many others.

There are multiple ontologies or dictionaries with information about semantic classes that can be used as background knowledge, either created by hand as WordNet (Miller, 1995) or in semi-supervised ways, such as DBPedia (Auer et al., 2007). However, manual and semi-supervised databases have important drawbacks: For many tasks its coverage is insufficient, especially for open classes. Moreover either they are static or their mechanism to include new knowledge is limited, and often the granularity provided is inappropriate for some purposes.

Unsupervised semantic class acquisition methods address these problems. The task is known as **semantic class learning**, semantic class induction or hyponym acquisition (Lin and Pantel, 2001b). The most popular approach is to process the text with a syntactic parser and select one or more surface patterns to extract a set of semantic classes (Hearst, 1992; Kozareva et al., 2008). These results are augmented with bootstrapping techniques that produce new patterns from the initial set of semantic classes. This iterative process is very productive, but the quality of the new semantic classes is reduced as the number of iterations rise. This problem is known as semantic drift.

Our work is closer to Probase (Wu et al., 2012), in which the initial patterns are fixed and the aim is to apply the method to a large number of documents so that relations between classes and instances are extracted with an associated frequency that is representative of how common is the relation in the world. For example, the instance *Obama* is frequently associated with the semantic classes *president* or *democrat*. We also see a long tail of classes associated with instances with less frequency. For example, *Obama* is related to a lesser degree to semantic classes like *husband* or *citizen*. As with the other propositions, some errors arise, for example we retrieve examples like *Obama* related to the class *daughter*. Unlike Probase, we dispense with a post-processing to refine the results. As we use well-known high-precision patterns, we expect that mistakes have low frequency and thus have low impact for textual inference purposes.

The graph based representation enables a straightforward semantic class proposition extraction considering the `NhasClassN` relation. This relation is generated using syntactic patterns that consider appositives, copula verbs and other patterns, similar to Hearst (1992) (See Table 4.2).

Syntactic pattern			Example
NE	nn	NN	*spokesman Baba Mohammed*
NE	appos	NN	*Umaru Yar, member of ...*
NN	appos	NE	*... capital, Abuja*
NN	such_as	NE	*spokesmen such as Phillip Umeadi*
NN	like	NE	*a ruler like Olusegun Obasanjo*
NE	be	NN	*Hamas is a group*

Table 4.2: Patterns for the assignment of semantic classes. Each entry belongs to a tuple *governor − dependency − dependant* where NN is a common name and NE is a named entity.

Instead of looking for the relation `NhasClassN`, we retrieve every entity even if it is not related with a semantic class. The purpose is to be able to estimate the probability of an entity acting as a specific class with respect to the frequency of the entity. Moreover, the tuples that we store are different. In this case, we force the first argument to be a named entity, and the second is a common noun that denotes the semantic class. Entities are stored with its entity type (i.e. person, organization, location, etc.), in order to gather information about the relation between entity types and classes. For example, a *person* is more likely to have classes as *teacher*, *player* or *mother*, whereas organizations would have classes such as *company*, *holding* or *library*. We also store information about the entity gender. We use the information contained on discourse referents that merge a named entity and a gender pronoun (i.e. he, she, his, hers).

4.4 Typed Propositions

Selectional preference is a long standing topic in NLP (Katz and Fodor, 1963; Zapirain et al., 2013) that evaluates the plausibility of an argument for a linguistic element. For example, in semantic role labelling it is necessary to evaluate how appropriate is an entity to fit a role in a verb. That is, we want to know that the object of the verb *drive* is probably a vehicle. This knowledge is valuable in other tasks such as relation extraction, where it is desirable to know if an argument can be accepted as a part of a relation. For example, we may discard any subject of the relation `married_to` that is not a person. However, selectional preferences are more complex for ambiguous elements, for example, it is possible to drive a company or to drive sheep. This knowledge is useful in tasks such as the mentioned semantic role labelling and relation extraction but also to others like question answering or word sense disambiguation.

Early efforts on selectional preferences were based on hand-made lexical resources such as WordNet (Resnik, 1997), as with semantic classes, but these methods lack of the necessary scalability to deal with open domains. The trend has evolved to the use of distributional similarity based on text collections (Agirre and Martinez, 2001; Zapirain et al., 2013).

We produce typed propositions as propositions with arguments tagged as semantic classes. In this case, the argument is replaced with the semantic class. Table 4.3 shows the patterns that we consider to extract typed propositions, which are analogous to propositions.

Typed propositions have been studied in other articles (Anaya-Sánchez and Peñas, 2015; Anaya-Sánchez et al., 2015), however, the evaluation is out of the scope of this thesis.

4.5 System Overview

In this section we show the steps that our method follows to build a Proposition Store from a corpus of documents. First we describe the methodology and then we show the details of the implementation, together with the datasets used.

Pattern	Example	Proposition
CV	*Kristen Anderson, a third year student won the annual award ...*	`<student - win>`
CVC	*Manchester United and Chelsea will watch Parma defender Matteo Ferrari.*	`<ORGANIZATION - watch - defender>`
CVPC	*Wright & Teague will donate 15m to WaterAid.*	`<jeweller - donate:to - ORGANIZATION>`
VCPC	*Vincent Van Gogh was born in Groot-Zundert, Netherlands, son of a small town preacher.*	`<son - born:in - LOCATION>`
CVCPC	*Investigative food journalist Joanna Blythman has made her name with her hard-hitting articles and books about the UK food industry..*	`<journalist - make:with - name - article>`
CPrepC	*... with goals from captain Michael Ballack.*	`<goals - from - captain>`
CHasC	*Huw Lewis, the editor of Jesus Life, the magazine of the Jesus Fellowship.*	`<magazine - has - editor>`

Table 4.3: Syntactic patterns used to extract typed propositions from the graphical representation. CVCPC is equivalent to Subject Class - Verb - Direct Object Class - Preposition - Complement Class

4.5.1 Methodology and Implementation

Our process is divided in the next steps:

1. Process a large number of documents using the representation proposed in Chapter 3. First we use the graph representation shown in Chapter 2. Summing up, we start from the documents parsed with Stanford Parser (Klein and Manning, 2003). We also use other features of The Stanford CoreNLP package: POS tagger (Toutanova and Manning, 2000), named entity recognizer (Finkel et al., 2005) and the coreference resolution system (Lee et al., 2011a). Then we follow the next steps: (1) Collapse coreferents into single nodes, (2) Perform a naive semantic role labelling that labels subjects, direct objects and complements with `arg0`, `arg1` and `arg2`, respectively, (3) Replace the genitive dependencies for the lexical relation `has`, and (4) Normalize lexical relations that denote that a named entity belongs to a semantic class into the relation `hasClass`.

2. Generate a set of propositions from the graphs applying syntactic patterns to a dependency tree. Once the graphs are generated, we use a Java application to apply a set of patterns (See Table 4.2). The output of this step is a set of Tab Separated Values (TSV) files with the following fields:

   - *Pattern*: Indicates what the type of the pattern is, i.e. NVN, NVNPN, etc.
   - *Document ID*: Identifier of the document where the instance was found.

- *Number of sentence*: Number of the sentence where the relation takes place.
- *Relation*: Lemma of the relation.
- *Argument [0-2]*: Descriptor of the argument.
- *Argument type [0-2]*: Indicates the syntactic type of the argument, i.e. noun, pronoun, named entity.

The number of arguments ranges from one to three depending on the syntactic pattern.

3. Generate a set of semantic class propositions searching for named entities on the graph. The output of this step is a new TSV file with instances with the following fields:

   - *Document ID*: Identifier of the document where the instance was found.
   - *Number of sentence*: Number of the sentence where the relation takes place.
   - *Named entity*: Named entity descriptor.
   - *Named entity type*: Named entity type, i.e. *person*, *organization* or *location.*
   - *Semantic class*: Semantic class gathered through the patterns.
   - *Gender*: If the discourse referent contains a gender pronoun, a gender is assigned.

4. Generate a set of typed propositions applying the same syntactic patterns for named entities on the graph, but considering only propositions with at least one argument tagged with a semantic class. The output of this step is a new TSV file with instances with the following fields:

   - *Pattern*: Indicates what the type of the pattern is, i.e. CVC, CVCPC, etc.
   - *Document ID*: Identifier of the document where the instance was found.
   - *Number of sentence*: Number of the sentence where the relation takes place.
   - *Relation*: Lemma of the relation.
   - *Argument class [0-2]*: Semantic class of the argument.
   - *Argument class type [0-2]*: Indicates the named entity type of the semantic class argument, i.e. *person*, *organization* or *location.*

   If an argument is not tagged with a semantic class, the output is the regular *Argument* and *Argument Type* fields.

5. After obtaining the TSV files, we aggregate the information for each pattern to obtain the frequency of each proposition. We use a bash script to get the frequency of each instance deleting the fields *pattern*, *Document Id* and *Number of sentence*.

4.5.2 Datasets

We acquire knowledge from the enriched graph-based representation introduced in Chapter 3. We have already presented the corpora in Section 3.4.1, but we repeat here the details for completeness.

- AIDS: A collection of 28,863 articles of the medical domain of AIDS.
- NYTFootball: A collection of 33,886 New York Times articles about US football.
- Webtext: The English subset of 1 million documents of the TAC 2012 KBP Source Corpus Additions Web Documents (LDC2012E23). This is a collection of web documents extracted from the GALE web corpus and added to the TAC KBP Source Corpus in 2012.
- KBP: The TAC 2012 KBP Source Corpus (LDC2012E22) is a collection of about 1.7 million documents established at the KBP 2010. It is composed by newswire documents, web documents, and a small subset of other documents all written in English.
- WebClue09: We create a corpus of 4.2 million documents that contain single sentences in English with two or more links to entities. We explain the purpose of this corpus in Chapter 6. The WebClue09 Dataset (Callan et al., 2009) contains about 100 million web pages in ten languages, with about 50% of the documents written in English. Web pages were collected in 2009 as a part of the Lemur Project. We use the Freebase Annotation extension provided by Google where named entities are linked to their Freebase id (Gabrilovich et al., 2013).
- Gigaword: The English Gigaword Fifth Edition (Parker et al., 2011) is a collection of newswire text data in English that includes four sources of news: The Agence France Presse English Service, The Associated Press Worldstream English Service, The New York Times Newswire Service and The Xinhua News Agency English Service. In total it contains near 10 million documents.

4.6 Results

Table 4.4 shows the number of graphs with at least one proposition extracted for each collection. The percentage of graphs with a proposition exceeds 90% in every case.

Corpus	Documents	Graphs	Graphs with propositions	%
AIDS	28.863	26.065	25.973	99,65
NYTFootball	33.886	32.746	32.686	99,82
Webtext	996.960	831.400	812.995	97,79
KBP	1.774.494	1.754.248	1.740.822	99,23
WebClue09	4.256.350	4.255.644	4.145.891	97,42
Gigaword	9.876.086	5.002.263	4.655.503	93,07
Total	16.966.639	11.902.366	11.413.870	95,90

Table 4.4: Number of documents and graphs of each corpus and number and percentage of graphs with at least one proposition extracted.

Table 4.5 shows the number of propositions extracted for each corpus. In total, we have acquired more than 1,970 million propositions. In the following sections we divide the number of propositions considering each kind of propositions.

Corpus	Propositions
AIDS	8,687,584
NYTFootball	10,873,176
Webtext	141,566,456
KBP	295,738,283
ClueWeb09	37,773,752
Gigaword	1,475,436,907
Total	1,970,076,158

Table 4.5: Number of propositions extracted for each corpus.

4.6.1 Proposition Stores

Table 4.6 shows the number of propositions extracted for each corpus. In the table we can see that shorter patterns produce more individual propositions. However, when considering unique propositions, the NVN pattern is able to retrieve more instances.

	AIDS		NYTFootball		Webtext		KBP		ClueWeb09		Gigaword	
	propositions	unique	propositions	unique	propositions	unique	propositions	unique	propositions	unique	propositions	unique
NV	1.429.273	133.260	1.990.118	694.886	30.865.746	5.374.721	49.087.949	8.907.098	4.492.388	1.447.928	226.490.338	21.945.736
NVN	1.208.118	788.291	1.581.382	1.208.375	23.267.475	10.622.106	42.508.591	22.338.885	3.702.372	2.106.088	201.618.591	60.009.481
NVPN	735.142	527.952	996.914	831.946	11.860.626	6.294.845	24.492.137	15.283.352	3.176.616	1.857.689	120.855.433	41.402.859
VNPN	936.374	675.659	917.302	726.169	12.405.735	5.894.365	28.132.157	16.376.826	3.117.574	1.835.158	142.390.867	44.562.936
NVNPN	459.753	396.734	607.293	530.584	6.741.477	3.738.542	16.125.089	11.259.073	1.920.314	1.196.582	82.533.488	31.860.039
NprepN	840.079	526.696	764.332	429.014	11.002.842	3.560.957	19.602.862	6.830.806	2.859.793	1.289.597	128.114.040	19.831.749
NhasN	711.559	466.021	944.705	455.403	10.947.547	3.301.547	24.876.439	7.703.324	3.765.882	1.453.837	121.598.897	19.099.810
Total	6.320.298	3.514.613	7.802.046	4.876.377	107.091.448	38.787.083	204.825.224	88.699.364	23.034.939	11.186.879	1.023.601.654	238.712.610

Table 4.6: Number of propositions extracted for each corpus and unique propositions.

Entities	entities	unique	classes	unique
AIDS	1,773,853	370,578	86,634	73,515
NYTFootball 2014	1,735,772	229,331	171,743	92,321
Webtext 2013	29,989,974	3,203,632	1,114,575	545,412
KBP 2013	66,494,985	5,834,629	4,021,686	1,688,716
ClueWeb09	13,592,358	1,028,352	739,889	325,189
Gigaword	358,156,581	11,599,521	21,564,363	4,381,156
Total	471,743,523	-	27,698,890	-

Table 4.7: Number of entities and semantic classes extracted for each corpus.

4.6.2 Semantic Class Propositions

Table 4.7 shows the number of entities found for each collection, plus the number of entities with a semantic class associated.

4.6.3 Typed Propositions

Table 4.8 shows the number of typed propositions acquired for each corpus. As it is expected, the number of propositions is much lower than compared to non-typed propositions. Moreover it is noticeable the high sparsity of the typed propositions, as about 50% of them are only found once.

	AIDS		NYTFootball		Webtext		KBP		ClueWeb09		Gigaword	
	propositions	unique	propositions	unique	propositions	unique	propositions	unique	propositions	unique	propositions	unique
CV	185.311	4.804	498.182	105.777	1.055.662	288.547	6.407.258	894.363	234.784	77.825	33.974.969	1.832.466
CVC	69.688	53.904	209.765	141.082	1.006.070	530.859	6.637.594	3.084.098	172.993	89.570	14.684.552	4.267.753
CVPC	60.095	48.569	194.722	148.317	563.241	328.234	3.213.074	2.073.667	132.979	75.048	11.893.351	4.074.101
VCPC	29.306	24.964	64.893	50.718	290.553	158.738	1.233.779	801.845	89.218	49.357	4.729.912	1.651.196
CVCPC	22.976	20.773	80.797	67.283	419.249	251.801	2.585.425	1.870.099	60.148	35.849	5.627.100	2.290.145
CprepC	23.569	18.964	52.670	34.457	342.268	161.821	818.607	403.789	99.768	47.294	4.821.879	1.471.828
ChasC	202.488	86.114	234.329	84.972	807.991	333.808	3.522.337	1.131.420	356.565	80.021	17.946.909	3.149.261
Total	593.433	258.092	1.335.358	632.606	4.485.034	2.053.808	24.418.074	10.259.281	1.146.455	454.964	93.678.672	18.736.750

Table 4.8: Number of typed propositions extracted for each corpus and unique propositions.

4.7 Application to Named Entity Disambiguation

The One Sense Per Discourse and the One Sense Per Collocation hypotheses are useful rules of thumb for word sense disambiguation. One Sense Per Discourse (OSPD) (Gale et al., 1992) refers to the tendency of each polysemous word to adopt only one specific meaning in every mention on a discourse. OSPD has been tested across several collections (Gale et al., 1992; Krovetz, 1998). One Sense Per Collocation (OSPC) (Yarowsky, 1993) states that a word act with the same sense when occurring in the same collocate, whether the collocate is positional or syntactic. This hypothesis is estimated to hold in around 70% of the cases (Martinez and Agirre, 2000).

In (Barrena et al., 2014) the goal is to study if these hypothesis hold for named entities and if this information can be used in a Named-Entity Disambiguation (NED) system. In a NED task, the aim is to map an entity mention in context into a specific canonical entity in a knowledge base.

This task requires a dataset with annotated entities. This is relatively straightforward for OSPD, since there are several corpora for entity recognition and disambiguation such as AIDA (Hoffart et al., 2011). However, there is not such a resource annotated with collocations and disambiguated named entities.

To build the new corpus, we first identify a total of 138 string mentions, which are ambiguous entities. Then, for each string mention we select at random a proposition in the KB collection where the string mention is related to an argument. For instance, we select the proposition `<ABC - has - radio>`. Then, we manually disambiguate five random occurrences of each proposition. This process was performed by a single person and later reviewed by the rest of the authors. The dataset, among others produced for the same paper, are publicly available[i].

Due to the limited number of mentions filling the desired properties, the result is 61 mention-collocation pairs with a total of 279 occurrences. From these pairs, only 1

[i]http://ixa2.si.ehu.es/OEPDC

corresponds to an ambiguous mention, which means that OSPC holds for 98.4% of the cases.

4.8 Conclusions

In this chapter we have presented our method for propositional knowledge acquisition. This method can be framed into the proposition extraction trend. Specifically, we start from a graph based representation that includes syntactic and semantic dependencies and we search for syntactic patterns to obtain propositions, semantic class propositions and typed propositions.

Our design choices are orientated to acquire a large amount of propositions so we can have evidence for textual plausibility. With this goal on mind, we admit as arguments named entities, common nouns and pronouns, and we apply a large set of patterns. Some of our extractions can be incorrect or incomplete by themselves because they do not capture the full context of the sentence, or because they contain pronouns which are a less representative form of their coreferent. However, this problems are minor considering that the high amount of evidence collected across big corpora minimizes the effect of the incorrect extractions.

We have applied our method to several corpora, both of general and specific domains. The result is a proposition store that considers the frequency of the propositions, which enables the assignment of a confidence to the knowledge and thus allows a probabilistic model of diverse problems related to language. We have acquired more than 1,970 million propositions from more than 11 million graphs. Graphs are automatically built from documents that belong to six different corpora. Propositions are divided into three kinds of propositions, regular propositions, semantic class propositions and typed propositions. In the following chapters we show how propositions can be used to perform textual inference, specifically in tasks related to language interpretation.

Regarding our research questions:

- **Research Question 2.1:** *Is it feasible to build knowledge bases from documents represented as graphs? What does this kind of representations provide?*

  The graph-based representation directly affects to the extraction of propositions. A long-distance dependence successfully captured becomes an interesting proposition acquired. Besides, other features included such as temporal and semantic relations allow extracting newer and unique propositions.

- **Research Question 2.2:** *What problems arise from an automatically generation of propositional knowledge bases?*

Propositional knowledge bases rely on several steps, each one of them introduces some limitations in the acquisition. The conceptual representation of text greatly influences the extracted propositions. In our case we have chosen to use syntactic patterns to extract semantic class knowledge. These syntactic patterns, although being very productive, impose a limit on the relations that can be extracted. Moreover, we rely on hand-made patterns.

5

Improving Parsing with Automatically Acquired Semantic Classes

In this chapter we define our method to exploit semantic class knowledge to improve parsing on appositive structures. To do so, we select the grammatical option that is semantically more compatible. We define a set of measures of compatibility and evaluate its performance.

A mistake is to commit a
misunderstanding.

Bob Dylan

Contents

5.1 Introduction

Information extraction systems typically preprocess text through a syntactic parser. However this step is an important bottleneck because parser errors are hard to solve or even identify at the extraction phase.

We can see in literature that parsers achieve overall performances over 90%, but it is well known that their performance drop in highly ambiguous dependencies such as PP attachments (Olteanu and Moldovan, 2005) or long distance dependencies (Bender et al., 2011).

Here we focus on **appositions**, which are grammatical constructions with two contiguous noun phrases where one defines or modifies the other. They are one of the most useful and productive dependence given by parsers because they are frequently used to denote the semantic class of a particular instance (Meyer, 1992). We develop a method to improve the apposition dependence detection that impact the retrieval of instance-class relations.

Consider the following example:

*David supports the team of his wife, **Julia**.*[i]

Dependency parsers often determine that there is an apposition between the common noun *wife* and the named entity *Julia*, leading to interpret that *Julia* belongs to the class wife. People would accept this interpretation using their background knowledge where *Julia* is a common name for a female person. Now consider the following example with exactly the same syntactic structure:

*David supports the team of his wife, **The Vikings**.*

Here it is not possible to determine the scope of the apposition without semantic information. However we know that *The Vikings* is not an usual name for a *wife*, whereas in the close context there is a different candidate, the common noun *team*, which is semantically more compatible. Thus, we want parsers to reproduce this behaviour and link *The Vikings* and *team*. Figure 5.1 illustrates the problem with an example that shows the original output given by a state of the art parser and the intended result. In the original output, the parser determines that there is an apposition that relates *Diskin* and *agency*. However, the correct interpretation of the sentence is to assign the apposition to the common noun that refers to the named entity, in this case, *chief*.

Given this scenario, we formulate the next research questions:

[i]We denote in bold the named entity and underline the candidate common nouns.

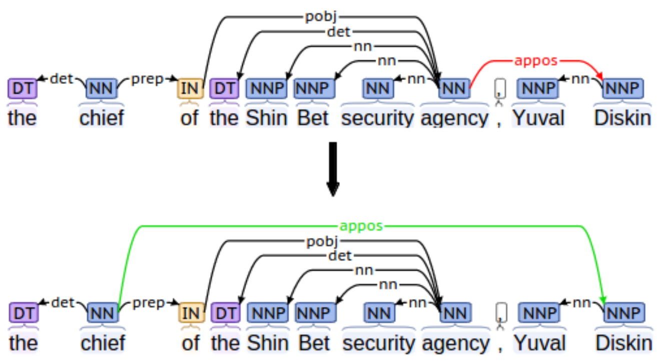


Figure 5.1: Correction from the typical parser output to the expected behaviour. Images generated with the Brat Tool.

- **Research Question 3.1:** *What kind of errors do parsers commit on appositions because of the missing semantic information?*
- **Research Question 3.1:** *Is it possible to overcome these errors considering information captured previously from text collections? What evidence can it provide to characterize the named entity?*
- **Research Question 3.1:** *What is the most effective way to measure the semantic compatibility between the candidates and the named entity?*
- **Research Question 3.1:** *What configuration of evidence and measures achieves the best results?*

To answer these questions we have parsed a large text corpus with two objectives: first to have a base to acquire background knowledge, and second, to get a sample of appositions where we compare the behaviour of two different dependency parsers in the state of the art.

The research conducted in this chapter has been published in (Cabaleiro and Peñas, 2013) and (Cabaleiro and Peñas, 2015).

This chapter is structured as follows. In Section 5.2 we define the scope of our study and check the source of errors in parsing. In particular we focus on cases where there are many candidates to govern an apposition. Our hypothesis is that these errors can be overcome by considering some background knowledge automatically extracted from large text collections. In Section 5.3 we explore how to gather this knowledge. We assume that the most semantically compatible candidate is the correct one. To measure this compatibility we study different configurations to combine the available evidence in Section 5.4, and show the results on Section 5.5. In Section 5.6 we discuss related work, and finally we provide some conclusions in Section 5.7.

5.2 Parsing Appositives: Building a Gold Standard

Our goal is to study appositions where dependency parsers have a chance to make a mistake selecting a governor because there are several grammatically correct candidates. In particular, we focus on cases where errors come from not taking into account the semantic compatibility between the two parts involved.

One structure where these errors can be found is a phrase where one side of the apposition is a named entity, and the other one has two or more common nouns that can be the nucleus of the noun phrase, and therefore a semantic class of the named entity. We refer to those common nouns as **candidates**.

We search for this structure in a large collection of textual data belonging to the TAC KBP task (Ji et al., 2011), composed by around 1.5 million documents that belong to different categories including newswire, blogs and phone calls transcriptions. We parse the collection with The Stanford Parser (Klein and Manning, 2003).

In this collection we obtain a total of 41,285,844 sentences, with 691,394 apposition dependencies, where 240,392 (34.7%) have more than one candidate.

We took a sample of 300 sentences to build a gold standard for evaluation. The following subsections show the qualitative analysis of this gold standard. According to the semantic compatibility between candidate classes an instances, we can distinguish the following cases:

5.2.1 One Valid Candidate

In the simplest case, such as the examples in the introduction, the common noun that acts as nucleus of the first noun phrase should be the governor of the apposition and it is more suitable than the other. Consider the next sentence:

. . . the <u>leader</u> of its largest rebel <u>group</u>, ***Manuel Marulanda****, . . .*

We target these cases as the ones to solve. Ideally, background knowledge should be able to discriminate suitable candidates. For example, a person as *Manuel Marulanda* is more frequently a *leader* than a *group*.

5.2.2 Several Valid Candidates

However, sometimes there are multiple common nouns that are valid candidates to be the governor of the relation. This occurs when either there is an implicit or explicit conjunction. An example of explicit conjunction would be:

. . . a prominent Jewish writer and Holocaust survivor, ***Ralph Giordano*** *. . .*

What we assume is that there are different simultaneous classes of the named entity. In some cases it could be interesting to add an apposition for each of the candidates.

An implicit conjunction occurs when there are noun compounds where common nouns act as modifiers, like in the next examples:

. . . by the IOC 's chief Beijing organiser, ***Hein Verbruggen****, . . .*

In these cases, we perform a linguistic test where we check if the nouns are a semantic class of the named entity. Following the example, we check if *Hein Verbruggen* belongs to the classes *chief* and *organiser*. As it happens, we say that both candidates are valid [ii].

In other cases the noun phrase includes a subordinated sentence that also has a common noun candidate to govern the apposition, for example:

Another passenger who gave only his surname, ***Chen*** *. . .*

5.2.3 Undecidable Candidates

In some sentences there are two classes referred to two different entities, but some extra-linguistic knowledge is needed to decide how they are related. For example:

. . . at least one brother of another defendant, ***Ali Dayeh Ali****.*

In the previous example there are two classes, *brother* and *defendant*, that refer to two different entities, *Ali Dayeh Ali* and an unknown entity. Without external knowledge it cannot be decided if *Ali Dayeh Ali* belongs to the class *brother* or to the class *defendant*.

[ii]Recall that the first noun of a noun-noun compound should not be a candidate. Even so, our method is robust enough to consider these candidates

5.2.4 No-apposition Case

In the manual inspection we have found multiple examples where the parser detects an incorrect apposition relation. To correct these errors is a line of work by itself, but here we limit ourselves to show what cases we have found. These are: Conjunction between two sentences, wrong apposition relation between a noun phrase and a verbal phrase, structures that denote the relation location-region, enumerations and nonsense text.

5.2.5 Summing up

We have shown how parsers depend on semantic knowledge to solve apposition dependencies with several grammatical alternatives. We have classified the sentences according to the semantic compatibility of the candidates and the named entity. Table 5.1 shows the number and percentage of appositions in each case.

In general, there are two main sources of errors in apposition parsing: When actually there is no apposition and the parser makes a bad choice of the dependency type, and when there are several candidates to govern the apposition and the parser makes a bad identification of the governor of the apposition. We tackle the latter, which include cases with one valid candidate, several valid candidates and undecidable candidates. These cases represent the 78.6% of the appositions of the sample.

We compose a Gold Standard with these appositions where we annotate the right apposition dependence. When there are several valid candidates, or it is undecidable which candidate is valid, we consider all valid choices as correct. Even if this cases are easier to solve, we include them to maintain generalization (i.e. there are cases with several valid candidates but at the same time another one that it is not). For our experimentation, we ignore the non-apposition cases. The result is a Gold Standard of 236 appositions.

One valid candidate	212 (70.6%)
Several valid candidates	22 (7.3%)
Undecidable candidates	2 (0.6%)
Total	236 (78.6%)
No apposition	64 (21.3%)

Table 5.1: Classes of appositions.

5.3 Background Knowledge Acquisition

In the previous section we saw that there are appositions with several candidates and the parser has to choose among them. We assume that the correct candidate is the most semantically compatible with the named entity. In this section we explain how we gather evidence to get a measure for this compatibility. Our hypothesis is that we can get it from large sources of text following the Open Information Extraction paradigm (Banko et al., 2007).

The goal in this phase is to extract relations between classes and instances with an associated probability. To acquire this information we use a graphical document representation G_d generated from a document d in the set of documents D.

Each document d is processed with Stanford CoreNLP (see Section 2) to obtain part-of-speech annotations (Toutanova and Manning, 2000), named entities (Finkel et al., 2005), syntactic dependencies (Klein and Manning, 2003) and coreferences (Lee et al., 2011b).

Graphs G_d are defined by a set of nodes V_d and set of edges E_d. A node $v \in V_d$ contains lexical information about words and multiwords. Edges $e \in E_d$ are typed syntactic dependencies obtained by the parser. Finally, coreferences are used to collapse different mentions of the same entity into a single node.

We obtain a set of classes from the graphs using very simple hand-crafted patterns based on syntactic dependencies where a common noun is a class for a named entity. Named entities have an entity type associated (*person*, *organization* or *location*) given by the named entity recognizer. When we find a match, we assign the common noun as semantic class of the named entity and we get an instance $NE - Class - Type$. Table 5.2 details the patterns and provides with some examples of instances collected.

Syntactic pattern			Example
NE	nn	NN	spokesman Baba Mohammed
NE	appos	NN	Umaru Yar, member of ...
NN	appos	NE	... capital, Abuja
NN	such_as	NE	spokesmen such as Phillip Umeadi
NN	like	NE	a ruler like Olusegun Obasanjo
NE	be	NN	Hamas is a group

Table 5.2: Patterns for the assignment of semantic classes. Each entry belongs to a tuple $governor - dependency - dependant$ where NN is a common name and NE is a named entity.

We do not pretend to obtain all the relations instance-class expressed in the collection with these patterns, but to acquire a representative and sufficient number of classes to evaluate their compatibility with named entities. We found a total of 4,410,293 instances.

After obtaining the semantic class assignments, we aggregate the information from all the collection to obtain the frequencies of named entities, classes and types. As a matter of example, Table 5.3 contains some tuples extracted with the associated joint probability.

NE name	Class	NE type	Frequency	$p(ne, c, t)$
Baba Mohammed	spokesman	person	17	3.8E-6
Umaru Yar	member	person	20	4.5E-6
Abuja	capital	location	356	8.1E-5
Phillip Umeadi	spokesman	person	13	2.9E-6
Olusegun Obasanjo	ruler	person	57	1.2E-5
Hamas	group	organization	1159	0.0002

Table 5.3: Sample of relations instance-class obtained.

To measure the semantic compatibility we use the joint probability between the classes and entity names, and also between classes and entity types (see Section 5.5.2). To get them we use a maximum likelihood estimator and marginalize:

$$p(c, ne) = \sum_{t} p(ne, c, t) \tag{5.1}$$

$$p(c, t) = \sum_{ne} p(ne, c, t) \tag{5.2}$$

Where c is the candidate, ne the entity name and t is the entity type. Table 5.4 shows a sample of some of the higher probabilities obtained.

Class	NE type	$p(c, t)$	Class	NE name	$p(c, ne)$
spokesman	person	0.031	president-elect	Barack Obama	0.001
president	person	0.023	spokesman	Sean McCormack	0.001
capital	location	0.009	president	Mahmud Abbas	0.0009
group	organization	0.007	spokeswoman	Dana Perino	0.0006
company	organization	0.006	mainland	China	0.0003

Table 5.4: Probabilities of Class and NE type and Class and NE name.

Algorithm 1 sums up the process of acquiring the background knowledge. $D = \{d_0, \ldots, d_n\}$ is document collection, $P = \{p_0, \ldots, p_6\}$ is the set of patterns and

$KB = \{t_0, \ldots, t_m\}$ is the knowledge base extracted. G_d is the graph of the document d, and it is defined by a set of nodes $V_d = \{v_0, \ldots, v_n\}$ and a set of edges $E_d = \{e_0, \ldots, e_m\}$. C_d represent the set of coreferences between pairs of nodes.

Algorithm 1 Knowledge base acquisition

```
Require: $D = \{d_0, \ldots, d_n\}, P = \{p_0, \ldots, p_6\}$
Ensure: $KB = \{t_0, \ldots, t_m\}$ where $t_i = \langle$ne name, class, ne type, frequency$\rangle$
  for each $d \in D$ do
    //transform $d$ into $G_d$:
    $< V_d, E_d, C_d > \leftarrow parse(d)$
    //collapse co-referent nodes
    for each coreference $(v_i, v_j) \in C_d$ do
      for each edge $(v_j, v_k) \in E_d$ do
        $E_d \leftarrow E_d \cup \{< v_i, v_k >\}$
        $E_d \leftarrow E_d \setminus \{< v_j, v_k >\}$
    $V_d \leftarrow V_d \setminus \{v_j\}$
    //apply extraction patterns
    for each $p \in P$ do
      $t \leftarrow match(G_d, p)$
      $KB = KB \cup \{t\}$
```

5.4 Experiment Design

In this section we explain our approach to identify and select the candidates to govern an apposition relation. We define the following variables in our method:

$S = \{s_0, \ldots, s_n\}$ represents the set of sentences, each one of them is composed by several candidates $C_{s_i} = \{c_0, \ldots, s_m\}$; $E_{s_i} = \{ne, t\}$ is the named entity with annotated evidence. Finally $F = \{f_1, \ldots, f_8\}$ denotes the set of configurations. We explain them and their use in more detail in the following sections.

5.4.1 Method

We gather sentences s_i that contain an appositive dependence formed by two noun phrases that fulfil two premises. The first one is that the first noun phrase must have more than one common noun. Each one is a candidate to be the governor of the apposition dependence $C_{s_i} = \{c_0, \ldots, s_m\}$.

We use as working example the sentence: *"the chief of the Shin Bet security agency, Yuval Diskin"*. In this sentence, the candidates are: c_0 =*agency*, c_1 =*chief*, c_2 =*security.*

The second premise is that the nucleus of the second noun phrase has to be a named entity. The named entity is the dependent part of the apposition. It is defined by two aspects, the entity name *ne*, which is the proper string that forms the named entity, and its entity type t (*person*, *organization* or *location*). We will refer to these values as *evidence* (E). Following our example, $E = \{ne = \textit{Yuval Diskin}, t = \textit{person}\}$.

Once we have candidates to govern an apposition relation, we select the most suitable by measuring their semantic compatibility with the named entity.

5.4.2 Measures

We use three measures of compatibility: normalized pointwise mutual information between candidate and evidence $npwmi(c_i, E)$; conditioned probability of the class given the evidence $p(c_i|E)$; and smoothed conditioned probability $p_{JM}(c_i|E)$.

The formula that describes the normalized pointwise mutual information we use is:

$$npmi(x;y) = \frac{pmi(x;y)}{-\log p(x,y)} \tag{5.3}$$

$$pmi(x;y) = log\frac{p(x,y)}{p(x)p(y)} \tag{5.4}$$

$p(x, y)$ is the probability estimated in the previous phase, and $p(x)$ and $p(y)$ is the result of marginalization. Normalized pointwise information scales the results to the range $(-1, 1)$, where -1 is the value where there are not joint observations and 1 is the value where the observations are always together.

In the third measure we introduce a smoothing factor to deal with data sparsity, as we expect to find cases with no evidence, i.e., entities that do not co-occur with some semantic classes. We use the Jelinek-Mercer algorithm with a default value of $\alpha = 0.99$.

$$p_{JM}(e|c) = \alpha p(e|c) + (1 - \alpha)p(e) \tag{5.5}$$

When $npwmi(e; c) = -1$ for each candidate $c_0, \ldots, c_n$ we say that we have no evidence. We deal in the same way with $p(c|e) = 0$ and $p_{JM}(c|e) = 0$. In these cases, we consider that our method fails.

5.4.3 Baselines

Our baselines are the results obtained in this Gold Standard by two parsers on the state of the art, the Stanford Parser (Klein and Manning, 2003) and the Fanse Parser

(Tratz and Hovy, 2011).

Stanford Parser has 191 (80.9%) correct appositions over the 236 instances in the Gold Standard, whereas Fanse Parser only reaches 70.7%. Stanford Parser has better results, but still there is a margin of improvement of 19.1%. We estimate that in the whole collection there are around 30.000 mistakes that can be overcome with our method.

5.4.4 Configurations

In order to evaluate the contribution of each evidence and each measure, we define a *configuration* as a pair of measure and evidence. Table 5.5 shows the configurations considered.

Entity Name (ne)	Entity Type (t)	Both (We assume conditional independence $ne \perp t \vert c$)
f_1: $npwmi(ne;c)$	f_4: $npwmi(t;c)$	
f_2: $p(c \vert ne)$	f_5: $p(c \vert t)$	f_7: $p(c \vert ne,t) \propto p(ne \vert c) * p(t \vert c) * p(c)$
f_3: $p_{JM}(c \vert ne)$	f_6: $p_{JM}(c \vert t)$	f_8: $p_{JM}(c \vert ne,t) \propto p_{JM}(ne \vert c) * p_{JM}(t \vert c) * p(c)$

Table 5.5: Configurations with evidence and measures.

For each configuration f we will select the candidate that maximizes the compatibility score between a candidate c and a entity E with entity name ne and type t. More formally, given the set of candidates $C = \{c_0, \ldots, c_n\}$, and the evidence $E = \{ne, t\}$ we select the best candidate c_f such as:

$$c_f = \arg\max_{c \in C} f(c, E) \tag{5.6}$$

5.5 Results

Our method always returns the highest scored apposition. Given the gold standard, this decision can be just right or wrong. Thus, we evaluate the experiments using accuracy, i.e., the proportion of correct appositions.

Figure 5.2 represents the results obtained for each configuration. Dotted lines show the two baseline systems. Five configurations outperform baseline systems significantly[iii]. But more interestingly, we find some regularities in the effect of the evidence and measures considered:

[iii]We apply the McNemar's test with *p-value* < 0.0001 in each case

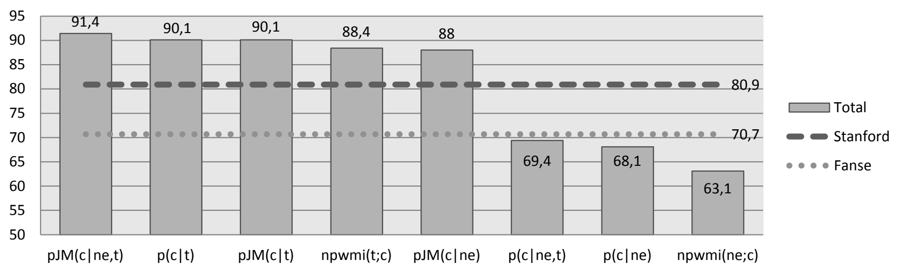


Figure 5.2: Percentage of correct appositions according to each configuration.

5.5.1 Evidence

We can see how, when considering the same measure, combining both evidence performs better than the use of entity types alone, which in turn performs better than considering only the entity name. This confirms the intuition that entity types are useful to generalize entity names, but combining both is even better to measure semantic compatibility.

As it is expected, there are many cases with no evidence for the entity name. This fact explains the low performance of $npwmi(ne, c)$, $p(c|ne, t)$ and $p(c|ne)$ when they are used without smoothing.

5.5.2 Measures

With respect to the measures, it is noticeable how conditional probability outperforms normalized pointwise mutual information regardless the evidence considered.

Smoothing is useful in cases where there is no evidence, because it will favour the most probable candidate classes. For example, if we take into account the entity name, there are 64 instances with no evidence in the sample. Smoothing these cases helps to push the results from 68.1% to 88%.

However, when considering the entity type as evidence, smoothing is not useful because in the sample considered there are no instances without evidence. Thus, the results remain equal.

Considering both entity name and entity type, smoothing it is also very useful. It allows to consider 59 instances that previously had no evidence and classify them better than the baseline, improving the results from 69.4% to 91.4%.

5.5.3 Best Configuration

The best performance is reached using both sources of evidence (entity name and entity type) in a smoothed conditional probability (Configuration 8: $p_{JM}(c|ne, t)$). With this configuration results rise from 80.9% accuracy in the baseline to 91.4%, which represents a relative improvement of 12.9% with respect to the parser.

5.5.4 Examples

Table 5.6 shows the results of our working example. In this case every evidence and measure considered chooses the correct candidate. As we said, *Yuval Diskin* is the entity name and its entity type is *person*. Candidates are *agency*, *chief* and *security*. Both Stanford parser and Fanse parser have chosen *agency* in the first place, but the correct candidate is *chief*.

	$f_1 =$ $npwmi(ne; c)$	$f_2 =$ $p(c\|ne)$	$f_3 =$ $p_{JM}(c\|ne)$	$f_4 =$ $npwmi(t; c)$	$f_5 =$ $p(c\|t)$	$f_6 =$ $p_{JM}(c\|t)$	$f_7 =$ $p(c\|ne, t)$	$f_8 =$ $p_{JM}(c\|ne, t)$
c_0 =agency	0.13	0.02	0.02	-0.22	4.2E-4	4.5E-4	5.5E-8	5.8E-8
c_1 =chief	**0.28**	**0.33**	**0.32**	**0.09**	**0.01**	**0.01**	**8.9E-6**	**8.8E-6**
c_2 =security	-1	0	1.4E-6	-0.25	1.1E-4	1.1E-4	0	2.0E-11

Table 5.6: Example of apposition corrected. *. . . the chief of the Shin Bet security agency, Yuval Diskin, . . .*

For the sake of completeness Table 5.7 shows the relevant input probabilities used to calculate all configurations in our running example.

ne	$p(ne)$	t	$p(t)$	c	$p(c)$	$p(c, ne)$	$p(c, t)$
				agency	0.0033	6.8E-7	2.7E-4
Yuval Diskin	2.8E-5	*person*	0.64	*chief*	0.0117	9.2E-6	0.011
				security	0.0001	0	7.3E-5

Table 5.7: Relevant frequencies for the working example.

Table 5.8 shows one case where our method worsens the baselines. In this case, *Virginia Casey* is the entity name and its entity type is *person*. Candidates are *daughter* and *cousin*. Both Stanford parser and Fanse parser have chosen *cousin* in the first place, and it is the correct candidate.

The reason of the failure is that we have gather more evidence of persons being *daughter* than *cousin*, and we have no evidence of the name *Virginia Casey* with these classes.

	$f_1 =$ $npwmi(ne;c)$	$f_2 =$ $p(c\|ne)$	$f_3 =$ $p_{JM}(c\|ne)$	$f_4 =$ $npwmi(t;c)$	$f_5 =$ $p(c\|t)$	$f_6 =$ $p_{JM}(c\|t)$	$f_7 =$ $p(c\|ne,t)$	$f_8 =$ $p_{JM}(c\|ne,t)$
$c_0 =$ *daughter*	-1	0	**2.7E-5**	**0.06**	**0.003**	**0.003**	0	**5.7E-12**
$c_1 =$ *cousin*	-1	0	4.9E-6	0.05	7.2E-4	7.2E-4	0	1.0E-12

Table 5.8: Example of apposition incorrectly changed. *. . . daughter Kim Boyer and Boyer's cousin, Virginia Casey . . .*

5.6 Related Work

The relation between syntactic ambiguity and semantic analysis, and also their application to syntactic disambiguation is a research field that has received attention for many years (Church and Patil, 1982; Resnik, 1993).

More recently, (Ciaramita, 2007) shows how to use semantic features in a syntactic parser to improve its performance. To do so, they tag the named entities with a named entity recognizer and consider the entity types as parts of speech. In (Agirre et al., 2008, 2011) the focus is in using WordNet (Miller, 1995) to generalize related words. For example, take the class *tool* instead of its instances *scissors* or *knife*, to improve syntactic parsers. Instead of WordNet, we use class-instance relations directly gathered from a reference collection, expecting that its coverage and granularity would be better adjusted to the topics and domains involved, and permitting also to calculate prior probabilities. Unlike them, our scope is not to build a parser with new features, but to test our hypothesis about the usefulness of semantic classes on the parsing of appositives.

Our approach is similar to other unsupervised methods for semantic class acquisition, also known as semantic class learning or semantic class induction (Lin and Pantel, 2001b). Many other authors have processed text with a morphosyntactic parser and then selected one or more surface patterns to extract a set of candidate semantic classes, that in turn are refined (Hearst, 1992; Kozareva et al., 2008).

In a different direction, (Schubert, 2002) argues that text contains general knowledge in form of assertions and they may be exploited after aggregating big amounts of data, like in KNEXT (Schubert, 2002), TextRunner (Yates et al., 2007) or DART (Clark and Harrison, 2009).

We use both ideas going one step beyond by combining induced classes with named entity types. In this way we find evidence of, for example, that classes *spokesman* and *leader* are related with more frequency with entities of type *person*, whereas *group* and *company* are classes that relate to *organization*.

Regarding appositions, although they are a really useful source of semantic knowledge, most of the research has been done in the context of coreference resolution (Soon et al., 2001; Ng and Cardie, 2002) or textual entailment (Roth and Sammons, 2007). Recently some efforts have been done in apposition extraction (Favre and Hakkani-Tür, 2009; Radford and Curran, 2013). In (Radford and Curran, 2013) they also use as a feature the semantic compatibility in appositions between named entities and nouns. To do so, they generalize the noun to a named entity type using WordNet. However, besides the dependence on WordNet, they do not address the problem of choosing between different candidates. Our approach allows to use the countings gathered from the text collection to solve this issue.

5.7 Conclusions

In this chapter we have studied how semantic classes can be used to solve structural ambiguities in apposition dependencies.

To do so we have answered the following research questions

- **Research Question 3.1:** *What kind of errors do parsers commit on appositions because of the missing semantic information?*

  Parsers fail when they lack of the semantic information to choose between several grammatically correct candidates. We have estimated that these cases represent the 78.6% of errors on apposition parsing, and have categorized them on three classes: appositions with one valid candidate (70.6%), with several valid candidates (7.3%) or with undecidable candidates (0.6%). We have focus on solving these three cases.

  The remaining 21.3% errors correspond to bad choices of the dependency type.

- **Research Question 3.2:** *Is it possible to overcome these errors considering information captured previously from text collections? What evidence can it provide to characterize the named entity?*

  We have used automatically acquired semantic classes as background knowledge to measure the semantic compatibility of candidates and named entities. This knowledge was divided in two different evidence, one relating semantic classes with entity names and other relating semantic classes with entity types.

  The results obtained reinforce our hypothesis that considering semantic compatibility between the two parts of the apposition can help to overcome parsing errors.

- **Research Question 3.3:** *What is the most effective way to measure the semantic compatibility between the candidates and the named entity?*

  We have used two different evidence (entity name and entity type) and three different measures (normalized pointwise mutual information, conditional probabilities and smoothed conditional probabilities).

  Using the entity name as evidence alone does not improve the results, since many names could be missing in the reference collection. On the other hand, the main problem of taking entity type alone as evidence is that some classes are very dominant (*chief*, *business*), and tend to be overassigned. According with the results, it is more effective to combine both entity name and entity type to get an accurate measure of semantic compatibility.

- **Research Question 3.4:** *What configuration of evidence and measures achieves the best results?*

  Regarding the configurations tested, the best results are obtained when combining both sources of evidence (relation between semantic classes and entity names, and relation between semantic classes and entity types) with smoothed conditioned probabilities. We reach a 91.4% accuracy which is a 12.9% of relative improvement with respect to the best baseline (80.9%), which corresponds to the Stanford Parser.

6

Grounding Proposition Stores for Question Answering over Linked Data

We show how grounding propositions within a distant supervision framework can improve the performance of question answering: Our approach converts utterances into graphs, which in turn are used to extract propositions. The propositions are aligned with Freebase labels using distant supervision through entity linking. The output is used to feed a semantic parser for question answering over linked data.

Any sufficiently advanced technology
is indistinguishable from magic.

Arthur C. Clarke

Contents

6.1 Introduction

Linked Data (LD) refers to *a set of best practices for publishing and connecting structured data on the Web* (Bizer et al., 2009). It establishes the bases for the Web of Data, an effort from the community of web users to create large amounts structured, machine-friendly knowledge, preserving the structure and semantics of the relations between elements. Although there are plenty of linked data databases (e.g. Freebase (Bollacker et al., 2008), DBPedia (Auer et al., 2007) or Yago2 (Hoffart et al., 2013)), common web users lack of the necessary know-how to use them.

Question Answering (QA) can be viewed as one human friendly method for accessing linked data since it alleviates the need to learn query languages such as SPARQL. QA systems typically employ semantic parsing to map natural language into a predicate-argument meaning representation. The map can easily be translated into knowledge base query languages.

We define **grounding** as the procedure for expressing natural language in terms of the target knowledge base language. More specifically, the task is to map an unbounded number of expressions (natural language) into a small set of entities and properties (linked data). For example the constructions *What does John do for a living?*, *What is John's profession?*, and *Who is John?* are be mapped to the same property `{John - Profession - X}`.

Grounding provides two key benefits. On the one hand, it alleviates the problem of logic form annotation by providing data for indirect supervision (Poon, 2013). Secondly, if the logic forms share the same vocabulary with the target knowledge base the querying step becomes trivial.

Semantic Parsing methods require a lexicon to enable the mapping between text and the labels of the knowledge base. A **lexicon** captures and ranks the candidate mappings between predicates in natural language and properties in the linked data database. For instance, solving the previous example would require an entry *living* → `profession`. However building these lexicons is not trivial and the contribution to the full system remains unmeasured because the final score is given by the complete system and involves other processes, e.g. choosing the appropriate entry of the lexicon.

Recent work proposes a method to build a lexicon by acquiring knowledge from large text corpora (Reddy et al., 2014). This process relies on distant supervision to build a lexicon that then is used to feed a semantic parser. Our goal is to study the contribution of this process of knowledge acquisition on closing the gap between natural language and linked data properties. Specifically, it is unclear which syntactic structures should be aligned and what is the impact of each one.

We use our methods of graph-based representation introduced in Chapter 3 and the acquisition method presented in Chapter 4 to transform natural language utterances into logic forms composed by a set of propositions, which are triples with the form `<argument 1 - predicate - argument 2>`. A propositions is mapped into a linked data triple `{argument 1 - property - argument 2}` to build a grounded proposition, which is a proposition expressed with the linked data vocabulary. We build a lexicon that we denote Grounded Proposition Store (henceforth, GPS) by grounding a large number of propositions automatically extracted from text. Finally, we combine the GPS with the method proposed in (Reddy et al., 2014) to create a scenario where grounding can be evaluated in isolation to study how different grounding configurations affect semantic parsing. Figure 6.1 shows how the utterance *"Carrie Fisher is the actress who played Princess Leia"* is transformed to a logic form composed by two propositions and then grounded into linked data properties. We explain this process and some related concepts in Section 6.4.

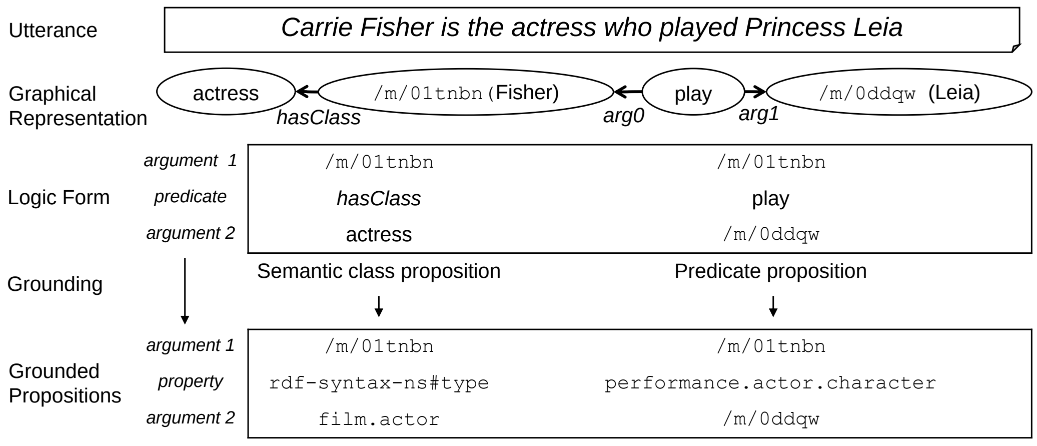


Figure 6.1: Example of acquisition of a grounded proposition. For simplicity, we represent the property `performance.actor.character` as a single triplet. In Freebase, this property is expressed with two triplets related by an intermediate entity.

We structure our research around the following research questions. In the context of a Semantic Parser trained using raw text for distant supervision:

- **Research Question 4.1:** *What are the methodological steps to build a GPS?*
- **Research Question 4.2:** *What is the impact of the GPS when used to feed a semantic parser for question answering?*
- **Research Question 4.3:** *What linguistic phenomena (syntactic-semantic relations) should be considered in the knowledge acquisition step?*
- **Research Question 4.4:** *Are external linguistic resources useful for enriching the GPS?*

This chapter is structured as follows: In Section 6.2 we motivate the choice of distant supervision using raw text for QA over LD. Section 6.3 details the architecture of the semantic parser, Section 6.4 studies the grounding step and presents our approach to build a GPS. In Section 6.5 we evaluate the effect of GPS in QA over LD and we present the results in Section 6.6. We finish with some conclusions in Section 6.7.

6.2 Semantic Parsing over Linked Data

Early works on semantic parsing for question answering were done on domains with controlled language and small predefined domains such as baseball (Green et al., 1961) and geography (Zelle and Mooney, 1996). However, these approaches cannot be scaled to general-domain knowledge bases.

As semantic parsers scaled to answer a wider range of queries, several problems arise. Firstly, systems have to deal with the lexical variability of the utterances, a problem that grows as domains become less restricted. Secondly, knowledge bases become bigger and richer, so the potential to give wrong answers increases.

Finally, dealing with the variability of knowledge bases also introduces additional challenges since semantic parsers have to adapt to different structures and vocabularies. Currently, many efforts point to linked data databases like DBPedia or Freebase as a source of general domain knowledge. The main reason is that they are a compromise solution between the high quality data that provide the hand-labelled databases and the extension of the automatically generated databases. Linked data databases are often structured in triples that denote relations between two entities, which are named properties. Properties are labelled with a name close to natural language. For example, an instance of the database may be `{John - profession - teacher}`, although these labels are arbitrary and, in fact, properties are defined extensively by their members.

Early approaches were too dependent on hand-labelled logic forms (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Hakimov et al., 2015), and hence were unable to scale up. More recent work aims to alleviate the supervision problem by using forms of distant supervision, i.e. observation of system behaviour (Chen and Mooney, 2011), conversations from dialog systems (Artzi and Zettlemoyer, 2011), schema matching (Cai and Yates, 2013), questions (Poon, 2013) and question-answer pairs (Kwiatkowski et al., 2013; Berant et al., 2013; Berant and Liang, 2014; Fader et al., 2013b, 2014).

GraphParser (Reddy et al., 2014) is a method for distant supervision that hypothesizes that a natural language predicate found in a text expresses a Freebase property. The idea is to identify pairs of entities connected through a predicate in a large document collection and look for the Freebase properties that connect both entities. For example, given the sentence s = *Cameron is the director of Titanic* one of the properties in Freebase between e_1 = `Cameron` and e_2 = `Titanic` is r = `film.directed_by`. Thus, we assume that $\{e_1 - r - e_2\}$ corresponds to the natural language expression s.

Distant supervision provides a noisy method to learn weights for each predicate-property pairs. For this purpose, the starting point is to take as prior the frequencies observed in a large text collection to build a lexicon and use it to feed the learning process.

GraphParser tackles this task by pairing reified logic forms derived from a Combinatory Categorial Grammar (CCG) parser (Clark and Curran, 2004) with Freebase properties. Each logic form corresponds in turn to a predicate-argument relation. Instead, we show how to obtain similar logic forms from a standard dependency parser. Dependency trees are transformed into graphs, which are then used to extract propositions. Then, propositions are aligned with Freebase to produce a new lexicon.

This setting allows us to measure the effect that different configurations of our Propositions Stores produce on semantic parsing when they are grounded to build the lexicon the system requires.

6.3 System Architecture

In this section we revise the architecture of the Question Answering system. The system is divided in three main layers: Text Processing, Learning and Inference. Figure 6.2 illustrates the architecture diagram.

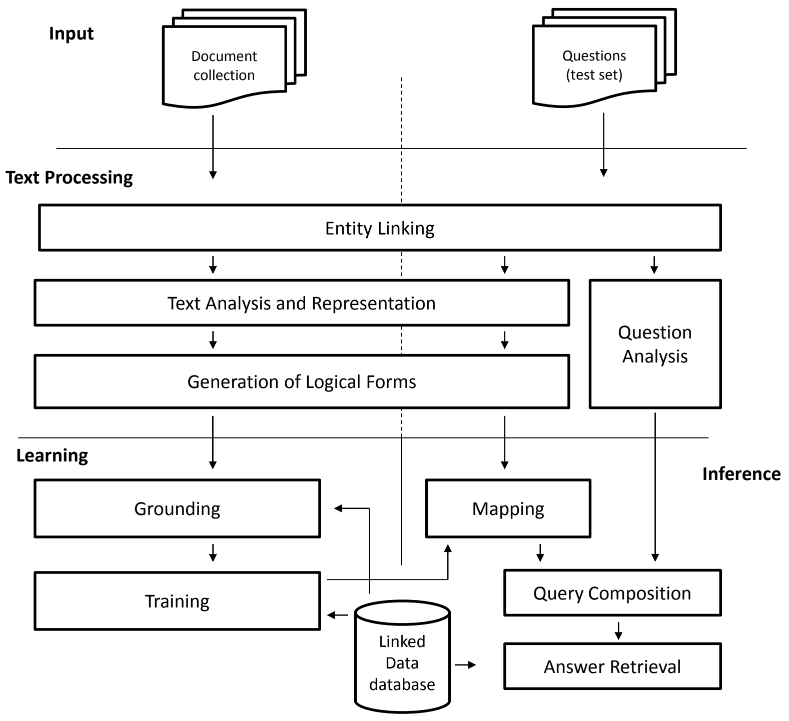


Figure 6.2: Architecture of the Question Answering system.

6.3.1 Text Processing

The purpose of text processing is to structure each document into a machine-friendly representation. This includes both sentences from the document collection and questions for the test. Questions are further analysed to extract the focus. The main tasks are the following:

1. Entity Linking: The entity linking component maps natural language entities to their canonical form in the linked data database, which is often done with a software tuned for the target database.

   For our experimentation we use the ClueWeb09 Corpus (Callan et al., 2009), which is already automatically tagged with Freebase entities (Gabrilovich et al., 2013). This is an automatic process whose authors estimate that has between 80-85% precision and 70-85% recall.

   Performance on this step has a double impact: On the one hand, unlinked entities are lost for the grounding step. This is a small problem, because the goal of the grounding step is to collect a wide sample of linguistic phenomena, not to ground every entity. On the other hand, mistakes on this phase introduce noise on the system, which we cannot detect on posterior steps.

2. Text Analysis and Representation: In this step, the system takes a sentence annotated with entities and produces a structured representation. It is expected that the structured representation is closer to the meaning of the sentence, and therefore mapping it to a property should be easier. GraphParser uses in this step a CCG parser. In our case, we rely on the graph-based representation presented in Chapter 3.

3. Generation of Logic Forms: This step is devoted to flatten the graphical representation into a set of propositions. In GraphParser, this step is equivalent to generate a set of predicates denoted as ungrounded graphs. In our approach we obtain neo-davidsonian reified logic forms from the graph-based representation, and from them we select predicate structures in the form of propositions as in Chapter 4. Section 6.4 gives more details about this processing.

4. Question Analysis: In question analysis, the main goal is to find the question focus, which is the part of the question that, if replaced by the answer, makes the question a single statement. For example, in the question *Who is the director of Titanic?*, the focus is given by *Who*, as it can be replaced by *David Cameron* to produce the affirmative statement *David Cameron is the director of Titanic.*

   Moreover, special operators like `count` and `argmax` are created by searching for special keywords like *How many* and *most* respectively.

6.3.2 Learning

The learning layer creates a model that evaluates a proposition to produce the most probable grounded proposition.

1. Grounding: In Grounding we take the pre-processed documents in order to build the GPS. In GraphParser, this step corresponds to the building of the alignment lexicon. We explain this step in detail in Section 6.4.

2. Training: We use the Maximum Weighted Graph (MWG) of GraphParser to replace each new proposition with the highest weighted grounded proposition given by the GPS.

   The default configuration of GraphParser for training is far more complex. It uses a Structured Perceptron that employs several kinds of features from the alignment between logic forms and properties. Features weights are tuned using denotation as a form of distant supervision. In GraphParser, training pushes the results from 36.5% to 39.3%.

6.3.3 Inference

The inference layer applies the model generated in the learning layer to the questions of the test set in order to generate an answer. Our configuration follows (Reddy et al., 2014). This process is subdivided in three components: Mapping, Query Composition and Answer Retrieval.

1. Mapping: Mapping is devoted to adapt the vocabulary and structure of the logic forms of the text processing step into the logic forms grounded on Freebase (As previously presented in Figure 6.1). It uses the model created in the learning step to decide which the most promising grounded proposition is.

2. Query Composition: The query composition step combines the information given by question analysis with the logic forms in order to create an executable SPARQL query. As grounded propositions are expressed with Freebase vocabulary, the composition is straightforward. Besides, it adds extra metadata about prefixes, domains and optional language filters.

3. Answer Retrieval: The last step is to retrieve the answer given the SPARQL query. We use Virtuoso[i] as an open source, free-available server to allocate the database and enable querying.

[i]http://virtuoso.openlinksw.com/

6.4 Grounding

Despite the need of a full QA system, our goal is to get insights on the effects of the grounding step in overall system performance. The challenge is to build a map between natural language utterances and the properties in a linked data database and measure the effect on the QA task.

In our case, we turn natural language utterances into propositions in the acquisition step and then we build a map from propositions to properties, which is the Grounded Proposition Store. The generation of GPSs is divided into three steps: Proposition Store Building, Proposition Store Grounding and Lexical Expansion.

6.4.1 Proposition Store Building

As presented in Chapters 3 and 4 Sentences are processed with Stanford CoreNLP (Klein and Manning, 2003) to obtain dependency trees which are also annotated with part-of-speech and coreferences (Lee et al., 2011a). We collapse multi-words nodes such as named entities into single nodes. Coreferences are also used to replace pronouns with the correspondent named entity. We depart from the Stanford syntactic dependencies (De Marneffe et al., 2006), and then we perform a naive semantic role labelling to normalize subjects, direct objects, indirect objects, copulatives, genitives and class-instance relations with a new set of semantic dependencies (See Table 6.1). This is an automatic process that relies on a predefined set of patterns. As a matter of example, we show some patterns in Table 6.2. We aim for two advantages: First it allows us to define a simple set of patterns to extract propositions, and second, this normalization reduces the sparsity of the extraction.

Role	Semantic dependency
Subject	subject
Direct object	dobject
Indirect object	iobject
Copulative	is
Genitive	has
Semantic class	hasClass
Prepositions	prep

Table 6.1: Semantic dependencies introduced by the semantic role labelling.

Input	Output
$ne(N_1), N_1 \longrightarrow nn \longrightarrow N_2$	$N_1 \longrightarrow hasClass \longrightarrow N_2$
$ne(N_2), N_1 \longrightarrow poss \longrightarrow N_2$	$N_2 \longrightarrow has \longrightarrow N_1$
$V \longrightarrow subj \longrightarrow N$	$V \longrightarrow subject \longrightarrow N$
$V \longrightarrow agent \longrightarrow N$	$V \longrightarrow subject \longrightarrow N$
$N_1 \longrightarrow nsubj \longrightarrow N_2$	$N_1 \longrightarrow is \longrightarrow N_2$

Table 6.2: Examples of patterns for naive semantic role labelling. N represent nouns and V are verbs. $ne(N)$ means that the noun N is a named entity.

Then, we extract a set of propositions applying patterns based on semantic dependencies. A proposition is composed by a predicate with two arguments, and is denoted as `<arg1 - predicate - arg2>`. We distinguish two kinds of propositions: Semantic class propositions, where the predicate denotes a type relationship (See Table 6.3), and predicate propositions, where the predicate denotes any other relationship (See Table 6.4).

Pattern	Example	Proposition
NhasClassN	*Beatty scored a double-win by casting Madonna as chanteuse Breathless Mahoney.*	`<Madonna - hasClass - chanteuse>`
NisN	*War of the Worlds is a movie with Tom Cruise.*	`<War of the Worlds - is - movie>`

Table 6.3: Syntactic patterns used to extract semantic classes from the graphical representation.

6.4.2 Proposition Store Grounding

We ground propositions in the following manner:

1. Select sentences with two or more entities present in Freebase.
2. Extract the set of propositions from the sentence that involve the entities present in Freebase.
3. Retrieve all possible types and properties from Freebase that link the entities found.
4. Pair propositions and retrieved properties to build the mapping lexicon. Semantic classes are mapped to types, and predicate propositions are mapped to properties.

Pattern	Example	Proposition
NhasClassN+NPN	*Robby Benson, a surprising choice for The Beast, is excellent.*	`<Robby Benson - choice:for - The Beast>`
NisN+NPN	*War of the Worlds is a movie with Tom Cruise.*	`<War of the Worlds - movie:with - Tom Cruise>`
NhasN	*Likewise, Jackman's Drover is a surprising bore.*	`<Jackman - has - Drover>`
NhasN+NPN	*Main Hoon Na is a Bollywood's film of 2004 starring Shahrukh and Sushmita Sen.*	`<Main Hoon - film:of - 2004>`
NPN	*Frank Welker was the voice of Megatron.*	`<voice - voice:of - Megatron>`
NNV	*Nichols and Koenig played Uhura and Chekhov, respectively.*	`<Nichols - play - Koening>`
NVN	*Four Rooms was released by Miramax in December, 1995.*	`<Miramax - release - Four Rooms>`
NVPN	*Don Knotts won five Emmys as Barney Fife.*	`<Don Knotts - win - Barney Five>`
VNN	*The Incredible Hulk also starring Liv Tyler, Tim Roth and William Hurt.*	`<Liv Tyler - star - Tim Roth>`
VNPN	*Tarantino is scheduled to begin shooting Death Proof in Austin in August.*	`<Death Proof - shoot:in - Austin>`
VPNPN	*Under Berg, Hancock was filmed in Los Angeles.*	`<Los Angeles - film:in:under - Berg>`

Table 6.4: Syntactic patterns used to extract propositions from the graphical representation. NVNPN is equivalent to Subject - Verb - Direct Object -Indirect Object

5. Compute the join probability of each pairing between a predicate r with a property p as a way to rank the most probable properties for a given predicate. The joint probability is calculated as:

$$p(r,p) = \sum_{(arg1,arg2)\in ARG} p(p \mid arg1, arg2) \cdot p(arg1, arg2 \mid r) \cdot p(r) \tag{6.1}$$

   where $p(p \mid arg1, arg2)$ is estimated as $\frac{1}{|P|_{arg1,arg2}}$ being $|P|_{arg1,arg2}$ the number of properties retrieved for a pair of arguments $arg1$ and $arg2$ that belong to the set of all arguments ARG, $p(arg1, arg2 \mid r)$ is estimated as $\frac{\#r(arg1,arg2)}{\#r}$ where $\#r(arg1, arg2)$ corresponds to the number of times where a proposition `<arg1 - r - arg2>` is derived from the corpus and $\#r$ is the number of times that the predicate r is derived from the corpus, and $p(r)$ is estimated as $\frac{\#r}{|R|}$ where $|R|$ is the total number of propositions.

The result is a GPS with a total of 3,416 semantic classes aligned to an average of 40.52 types, plus 10,799 predicates aligned to an average of 3.82 properties. Tables 6.5 and 6.6 show some of the most frequent pairs extracted for semantic classes and predicates respectively.

Semantic class	Type	$p(r,p)$
president	`organization.organization_founder`	5.96E-4
	`business.board_member`	4.23E-4
	`people.person`	3.43E-4
son	`people.person`	6.63E-4
	`people.deceased_person`	2.84E-4
	`people.family_member`	5.20E-5
founder	`organization.organization_founder`	1.68E-3
	`business.board_member`	1.54E-4
	`people.person`	1.37E-4

Table 6.5: Highest probability pairings between semantic classes and types in Freebase.

Predicate	Property Argument 1	Property Argument 2	$p(r,p)$
bear	`person.place_of_birth.1`	`person.place_of_birth.2`	0.014
	`place_lived.person`	`place_lived.location`	0.008
	`person.nationality.1`	`person.nationality.2`	0.006
has	`person.nationality.1`	`person.nationality.2`	0.005
	`employment_tenure.person`	`employment_tenure.company`	0.004
	`organization.geographic_scope.1`	`organization.geographic_scope.2`	0.001
die:in	`deceased_person.place_of_death.1`	`deceased_person.place_of_death.2`	0.007
	`deceased_person.date_of_death.1`	`deceased_person.date_of_death.2`	0.001
	`person.date_of_birth.1`	`person.date_of_birth.2`	0.001

Table 6.6: Higher probability pairings between predicates and properties. We removed the domains of the properties for readability.

6.4.3 Lexical Expansion

Alignment through examples is very sensitive to lexical variability. For instance, we may not find the verb film in a proposition like `<Cameron - film - Titanic>`. However, if instead we have found the verb *direct* like in `<Cameron - direct - Titanic>` we could easily expand our lexicon by replacing the verb *direct* with the related verb *film*. We explore the use of synonyms in WordNet (Miller, 1995), a hand-made lexical database. Lexical expansion through external resources has proven to be useful in Semantic Parsing both with WordNet (Walter et al., 2012) and other sources (Kwiatkowski et al., 2013).

The lexical expansion process takes a predicate paired with the properties and obtains every synonym given by WordNet. Then, the final weight of each predicate is divided among the number of synonyms retrieved. We compute the joint probability of each pairing between a synonym s with a property p, that is:

$$p(s,p) = \sum_r p(s,r,p) = \sum_r p(s \mid r,p) \cdot p(r,p) \tag{6.2}$$

where $p(s \mid r,p)$ is estimated as $\frac{1}{|S|_r}$ being $|S|_r$ the number of synonyms retrieved for the predicate r and $p(r,p)$ is calculated as in Equation 6.1.

The resulting expanded GPS extends the predicates up to 72,130, with an average of 5.42 properties paired.

6.4.4 Working example

We illustrate the full process through a working example. Consider the sentence *Spurlock is the creator of the film Supersize Me*. Our method is decomposed in the following steps:

1. Proposition Store Building
   a) Select a sentence s: The working example is selected because it contains two Freebase entities, *Spurlock* and *Supersize Me*. Entities are annotated with their Freebase id, which is `m.035sc2` and `m.022prxf` respectively.
   b) Transform the sentence into a graph g_s: Figure 6.3 shows the resulting graph.

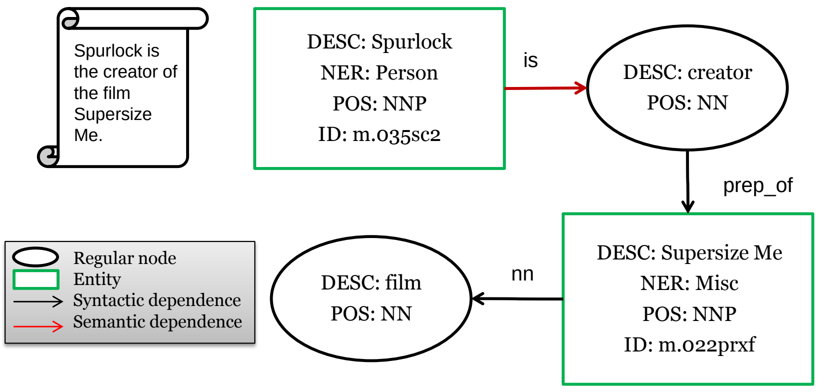


Figure 6.3: Graph extracted from the sentence *Spurlock is the creator of the film Supersize Me.*

c) Extract propositions `<arg1-r-arg2>`: We flatten the graph representation applying syntactic patterns to extract propositions, composed by a predicate r and a pair of arguments $arg1, arg2$. Table 6.7 shows the patterns found in the working example and the resulting propositions.

Pattern	Proposition
NPN	`<creator - of - m.022prxf>`
NisN	`<m.035sc2 - is -creator>`
NisN+NPN	`<m.035sc2 - creator:of - m.022prxf>`

Table 6.7: Propositions extracted from the sentence *Spurlock is the creator of the film Supersize Me.* Note that entities are replaced with their Freebase id.

2. Grounding Proposition Stores

a) Ground propositions: A grounded proposition `{arg1 - p - arg2}` is built by replacing the original predicate r of a proposition with a property p. We search for the properties $P_{arg1,arg2} = p_0, \ldots, p_n$ that connect the entities $arg1, arg2$ in Freebase.

b) Build final lexicon: Compute the probability $p(r, p)$. Tables 6.8 and 6.9 show the result of $p(\text{"}creator\text{"}, p)$ for semantic class propositions and predicate propositions, respectively

Semantic Class	Type	$p(r,p)$
creator	`organization.organization_founder`	3.45E-5
	`business.board_member`	7.30E-6
	`people.person`	6.68E-6
	`film.writer`	5.90E-6
	`film.director`	4.30E-6
	...	

Table 6.8: Relevant probabilities for the semantic class *creator*. Probabilities are obtained after processing the whole collection.

Predicate	Property Argument 1	Property Argument 2	$p(r,p)$
creator	`film.written_by.1`	`film.written_by.2`	9.06E-6
	`organization.founders.1`	`organization.founders.2`	7.77E-6
	`employment_tenure.person`	`employment_tenure.company`	7.77E-6
	`film.directed_by.1`	`film.directed_by.2`	6.47E-6
	...		

Table 6.9: Relevant probabilities for a proposition with the predicate *creator*. We removed the domains of the properties for readability. Probabilities are obtained after processing the whole collection.

3. Lexical Expansion

   a) Create new pairs: Search in WordNet for synonyms of a predicate. For example, for the predicate *create* we find the synonyms *make* and *produce*. Properties that would be paired with the original predicate are paired with the synonyms $S_r = \{s_0 = make, s_1 = produce\}$.

   b) Compute the probability $p(s, p)$. For instance, computing the probability $p(\text{"}creator\text{"}, p)$ require to add up the probabilities of each pair (r, p) whose predicate r yields "*make*" as synonym.

6.5 Experiment Design

In this section we explain the semantic parsing task and the datasets that we have used both for the creation of the GPS and for testing them for the task of semantic parsing.

Formally, our goal is to learn a function to map an utterance u to a query q over a database D. The database is defined by a schema that contains properties $p \in P$ and entities $e \in E$. Both properties and entities are human-readable strings like `film.directed_by` or `David_Cameron`. The database contains a set of triples $\{e_1 - p - e_2\}$. For each utterance (natural language question) the system gets set of SPARQL queries $Q_u = q_0 \dots q_n$. Each query executed over the database obtains a set of answers $A_q = a_0, \dots, a_m$.

6.5.1 Implementation

We take advantage of (Reddy et al., 2014) evaluation module to evaluate our approach against the test collections. We use the Maximum Weighted Graph (MWG) configuration, a baseline that replaces each predicate with the highest weighted property without any further training. With the goal of measuring the contribution of each syntactic pattern, we perform an ablation test where we remove one by one a syntactic pattern in the GPS building process and compare the resulting GPS with the full building process. Moreover, we experiment with the expansion of propositions using WordNet and perform an additional ablation test on the expanded GPS. Then, we try to maximise our results by removing the harmful patterns, and finally, we perform a comparison with a state of the art system.

6.5.2 Evaluation Measures

Following (Kwiatkowski et al., 2013; Berant et al., 2013; Reddy et al., 2014) and many others, the system is evaluated using precision, recall and F1-measure.

$$Precision = \frac{\text{number of correct system answers}}{\text{number of system answers}} \tag{6.3}$$

$$Recall = \frac{\text{number of correct system answers}}{\text{number of questions}} \tag{6.4}$$

These measures consider only the first answer of the ranking. An answer of a query is correct if contains exactly the same responses that the gold standard. Partial answers are considered mistakes. Results are ranked according to the F1-measure.

6.5.3 Dataset

Our dataset is a subset of WebQuestions (Berant et al., 2013) as defined in (Reddy et al., 2014). The scope is reduced to three domains: film, business and people. The final dataset is composed by 200 questions devoted for development and 570 questions for testing. Note that, with the MWG configuration, the development dataset is not used.

6.6 Results

We compare the effect of our lexicon against having no lexicon at all to highlight how determinant is the grounding step. With a GPS, results are pushed 25.3 points with respect to the empty lexicon, showing the high impact that knowledge acquisition has in the task (Table 6.10). We also show that the baseline proposed in GraphParser is already informed. MWG uses the default lexicon of GraphParser to ground logic forms without any training. Compared with the empty lexicon, results are pushed 28.7 points, which indicates that the lexicon contributes with a 78.6% of the total result of MWG. When compared to our regular GPS, the contribution is similar, with a 76.4% of the total result.

Ablation Test: Table 6.11 shows the results of the ablation test, divided between regular and expanded system. The GPS row corresponds to the full system, while the remaining rows correspond to individual ablations where a single proposition type is

	Prec	Rec	F1	difference
Empty lexicon	8.40	7.30	7.80	
GPS	35.74	30.95	33.14	+25.34 (76.46%)
GraphParser - MWG	39,4	34,0	36,5	+28.70 (78.63%)

Table 6.10: Comparison between an empty lexicon, the regular GPS and GraphParser's baseline.

removed. We observe that in both cases the F1-measure of the full system is close to the highest result. Moreover, the difference between these systems and most of ablations is small, a range of [-0,31,+0,16] in the regular case and [-0,35,+0,12] for the expanded case. The VNPN, NVN y NVPN patterns are the exception, with higher loses, up to 12.64 points. This means that these structures are essential to acquire knowledge in the context of linked data, as they gather information that is unavailable for other patterns. Considering individual ablations, the best result for the regular case is to ignore hasClass+NPN structures and for the expanded case the best results are obtained by ignoring NNV or VNN structures in the construction of the proposition stores. These results show that these patterns introduce more noise than useful information.

	Regular			Expanded		
	Prec	Rec	F1	Prec	Rec	F1
GPS	35.74	30.95	33.14	38.08	32.99	35.38
-NNV	35.82	31.02	33.22 (+0.08)	**38.20**	**33.10**	**35.50 (+0.12)**
-VNN	35.68	30.89	33.09 (-0.05)	38.16	33.06	35.46 (+0.08)
-NPN	35.46	30.7	32.88 (-0.26)	38.08	32.98	35.38 (0.00)
-NisN	35.48	30.71	32.90 (-0.24)	38.06	32.97	35.36 (-0.02)
-NhasClassN+NPN	**35.90**	**31.10**	**33.30 (+0.16)**	38.04	32.94	35.34 (-0.04)
-NhasN+NPN	35.82	31.02	33.22 (+0.08)	38.04	32.94	35.34 (-0.04)
-NhasN	35.82	31.02	33.22 (+0.08)	38.02	32.93	35.32 (-0.06)
-VPNPN	35.74	30.94	33.14 (0.00)	37.84	32.77	35.15 (-0.23)
-NisN+NPN	35.67	30.46	32.83 (-0.31)	37.8	32.74	35.11 (-0.27)
-NhasClassN	35.48	30.71	32.89 (-0.25)	37.96	33.39	35.03 (-0.35)
-VNPN	28.58	24.69	26.48 (-6.66)	30.92	26.76	28.72 (-6.66)
-NVN	26.74	23.14	24.74 (-8.40)	27.54	23.84	25.54 (-9.84)
-NVPN	23.48	20.38	21.78 (-11.36)	25.10	20.97	22.74 (-12.64)

Table 6.11: Experimental results for the ablation test. We report the difference between the ablation and the GPS with and without expansion.

Lexical Expansion: Table 6.12 shows that the expanded GPS consistently outperform the regular GPS both in the full system and in every ablation, with a contribution of 0.8% in the worst case and 2.5% in the best. This confirms that GPS can be effectively expanded using WordNet synonyms in order to reduce the lexical gap, and points out that external resources can be a good complement to distant-supervised methods to acquire knowledge. In other words, the more knowledge we inject into the system the better performance it shows.

	Regular	Expanded	Difference
GPS	33.14	35.38	+2.24
-NPN	32.88	32.98	+2.50
-NisN	32.90	32.97	+2.46
-VNN	33.09	33.06	+2.37
-NNV	33.22	33.10	+2.28
-NisN+NPN	32.83	32.74	+2.28
-VNPN	26.48	26.76	+2.24
-NhasClassN	32.89	33.39	+2.14
-NhasN+NPN	33.22	32.94	+2.12
-NhasN	33.22	32.93	+2.10
-NhasClassN+NPN	33.30	32.94	+2.04
-VPNPN	33.14	32.77	+2.01
-NVPN	21.78	20.97	+0.96
-NVN	24.74	23.84	+0.80

Table 6.12: Comparison between F1 measure for the Regular and Expanded GPS.

Best Configuration: Table 6.13 shows the results of the configurations that remove harmful patterns, which are NhasClassN+NPN, NhasN, NhasN+NPN and NNV for the regular configuration and NNV and VNN for the expansion. Results indicate that these patterns can be omitted with a further small boost on the results. More significant, we can achieve the same performance with less patterns, which in turn means less computational cost of building the lexicons.

	Regular			Expanded		
	Prec	Rec	F1	Prec	Rec	F1
GPS	35.74	30.95	33.14	38.08	32.99	35.38
(1)	**35.86**	**31.06**	**33.26 (+0.12)**	**38.22**	**33.12**	**35.52 (+0.14)**
(2)	**35.86**	**31.06**	**33.26 (+0.12)**	38.18	33.08	35.48 (+0.1)

Table 6.13: Experimental results removing the harmful ablations. (1) Corresponds to NNV and VNN patterns which are harmful for the expanded GPS and (2) corresponds to NhasClassN+NPN, NhasN, NhasN+NPN and NNV, which are harmful for the regular GPS.

Comparative Evaluation: We compare our salient configurations with GraphParser's baseline (MWG) and system with training (GraphParser). Table 6.14 shows how we achieve similar results, which is promising considering that we limited the acquisition to the sentences used in GraphParser. Our representation relies on syntactic parsing and syntactic rules to process sentences, which is faster than the CCG parser, so there is potential to increase the acquisition by using a larger document collection. Note that GraphParser's default and baseline configuration have 2.8 points of difference, which represents a 7.6% of relative improvement. Again, this highlights the importance of the grounding.

	Prec	Rec	F1
GraphParser	41,9	37,0	39,3
GraphParser - MWG	39,4	34,0	36,5
GPS expanded-(1)	38,22	33,12	35,52
GPS expanded	38,08	32,99	35,38
GPS	35,74	30,95	33,14

Table 6.14: Comparison between GraphParser and the regular and expanded GPS, plus our best system which is the NNV and VNN ablation of the expanded system (1). MWG refers to GraphParser's baseline configuration where there is no training.

6.7 Conclusions

Question Answering systems in the state of the art are evaluated as a monolithic system that involves acquisition, learning and querying without measuring the contribution of each component. However, such a system level comparison does not provide any insight into the real contribution of each component, and, in particular, the effect of the amount of knowledge digested in the final result.

We show here that the main component is the lexicon itself (the GPS in our case), so we need better ways of creating and evaluating this resource before addressing the learning and querying steps in deeper and more sophisticated settings.

We have presented both the methodology to generate a GPS linked to a particular knowledge base, and a study evaluating the effect in QA performance that different natural language structures produce when they are considered to build the GPS.

For this reason, we evaluated the construction and grounding of the lexicon (GPS in our case) *per se*, without additional training or wiring to the SPARQL queries generation. This additional evaluation, such as done in (Kwiatkowski et al., 2013; Berant et al., 2013; Reddy et al., 2014), is out of the scope of this work. Different methods for training and querying must be evaluated once the GPS is fixed, so we can learn about the effect of different techniques.

We summarize our conclusions to the following research questions:

- **Research Question 4.1:** *What are the methodological steps to build a GPS?*

  We have developed a method to map propositions into Freebase properties using distant supervision that helps in solving both lexical and structural gaps by finding multiple grammatical structures and lexical realizations of the same query.

  Our method is divided in three steps: (1) Build a proposition store. To do so, we select relevant sentences, transform them into graphs from which we extract propositions. (2) Ground each proposition by pairing them with KB properties considering linked entities, and (3) Compute the global weights of each pairing.

  We consider an optional step, (4) Perform a lexical expansion by creating new pairs and re-evaluate the weights of the lexicon entries.

- **Research Question 4.2:** *What is the impact of the GPS when used to feed a semantic parser for question answering?*

  Building and grounding the proposition stores is key to the final performance of the semantic parser. A system with an empty alignment lexicon achieves a 7.80% of F1-measure. In baseline systems with lexicon but without training, our experiments show that the lexicon contributes with near 80% of the results, and training only accounts for 7.6% of relative improvement.

- **Research Question 4.3:** *What linguistic phenomena (syntactic-semantic relations) should be considered in the knowledge acquisition step?*

  We have analysed different linguistic structures that can be included in the GPS and what is the contribution on the final result. For this setting, NVN, NVPN and VNPN patterns have a significant effect in the performance. Our results suggest that extensive coverage of every possible syntactic pattern is not as useful as it may be intuitive. Conversely, systems can dispense with some patterns and reduce the computational cost of building the alignment.

- **Research Question 4.4:** *Are external linguistic resources useful for enriching the GPS?*

  We have shown how to enrich the lexicon using linguistic information from external resources, helping to bridge the lexical gap between utterances and database queries. Enrichment consistently pushes the results in every case, in a range from 0.8% to 2.50% of absolute improvement.

7

Conclusions and Future Work

In this chapter we state the main conclusions obtained and we discuss some interesting lines of research.

I see my path, but I don't know where it leads. Not knowing where I'm going is what inspires me to travel it.

Rosalía de Castro

Contents

7.1 Research Hypotheses

In this dissertation we have studied the fundamental problem of knowledge acquisition in the form of propositions and its uses for language interpretation. Typically, NLP tasks are addressed as independent problems and tackled with specific purpose systems. We argue that, since NLP tasks are specific realizations of our human ability to comprehend and use language, in the long term we will need to provide this behaviour to machines through the definition of methods to acquire and use general purpose background knowledge.

This thesis explores a new paradigm where propositional knowledge is used as a solution to bring closer meaning representations and knowledge bases. We have structured our exploration around the following research hypotheses:

Research Hypothesis 1: *Propositional knowledge can be used to represent and store background knowledge.*
Propositions encode predicate-argument structures that can capture open-domain knowledge. Propositions are manipulable by computers, and enable general purpose inferences. Gather a large number of propositions through the representation of large collections of text yields a background knowledge base that mimics common sense.

In literature we can see that recently open information extraction and specifically propositions have arose as a promising method to capture open domain knowledge. Although there is some work that uses propositional knowledge for specific interpretation tasks, there is a lack of work connecting propositional knowledge acquisition with generic natural language understanding, particularly, as a source of background knowledge.

In this dissertation we have presented a method for automatic propositional knowledge acquisition, and also we have built several proposition stores, both from specific and general domain corpora. These resources have been used for two different NLU reliant tasks, syntactic and semantic parsing. Specifically, we have used propositional knowledge to provide evidence to categorize entities and named entity types on semantic classes and to provide a notion of plausibility in predicate-argument relations.

Research Hypothesis 2: *A graph-based representation is useful to encode documents for proposition extraction.*
A prerequisite to acquire propositional knowledge is to encode text in a meaning representation. Even if an intrinsically evaluation of the knowledge base is infeasible, we define some properties for the meaning representation that reasonably can improve the proposition acquisition.

We discussed that graphs convey a set of properties that are desirable to represent meaning, such as the ability to relate context that may be distant on text. Although there are plenty of meaning representations based on graphs, to the best of our knowledge this is the first attempt to use one to automatically extract propositional knowledge.

We have defined a methodology to automatically build a graph-based representation from text that aims to express the semantics of a sentence in a condensed, simplified way. We have built a large repository of documents represented as text, and used this resource to perform automatic extraction of propositions through syntactic and semantic patterns. Besides, the representation has been used for feature extraction in a relation extraction task.

The graph based representation conveys additional benefits, such as being easily readable by people. For instance, it is easy for a human to detect semantic inconsistencies, such as incorrect dependencies. As a matter of example, the task of correction of appositive dependencies arose from the manual inspection of a graph.

A limitation of our method is that it does not encode meaning across consecutive sentences as in discourse representation theory.

Research Hypothesis 3: *Propositional knowledge can serve as basis to perform textual inference.*
A proposition store can be used as a background knowledge base that enable textual inferences related to language interpretation tasks. Previously we have discussed that the utility of a knowledge base is dependent on the task where it is applied. In this case, we aim to adapt a general purpose knowledge base to two different tasks.

Natural Language Processing tasks are often modelled as statistical data driven tasks that may yield satisfactory results, but still do not solve the task in the sense that these systems do not comprehend and resolve the problem as a human would do, and therefore a human judgement would ever be preferred.

We argue that there is a need to provide a mechanism to the computers to reason based on knowledge, as opposite to raw text statistics. In this line, we believe that proposition are a promising method to represent knowledge in a manner that it can be used perform textual inference, because propositions suppose a compromise solution between raw statistics and strict logical semantics.

Our proposition stores were conceived without specific purposes, still they have been used for multiple language interpretation tasks, including semantic and syntactic parsing, named entity disambiguation, building signature models and interpretation of eventive propositions.

In each of these tasks we have built a different model of knowledge from the propositions. Crucially, propositions are flexible in two senses: Firstly, propositions encode knowledge in a meaning representation very close to language, and therefore their expressivity is higher than other knowledge representations that are more structured, such as ontologies. Secondly, tasks may require a subset of the total knowledge, and propositions allow to select only the relevant information. For instance, we discarded the predicate-argument propositions to improve syntactic parsing and only used semantic class propositions. For grounding, we have selected propositions whose arguments are named entities, specifically, those that can be linked to the knowledge base. This capability is useful to reduce the search space.

7.2 Conclusions

The focus of this thesis is to study the acquisition of propositional knowledge by automatic means, and to show how it can be used as a source of background knowledge in two knowledge intensive tasks, syntactic and semantic parsing.

We began this dissertation dividing the problem of using knowledge for language interpretation in three main challenges: meaning representation, knowledge acquisition itself and textual inference. We address these challenges in four objectives: the first objective is related to meaning representation, the second objective is related to knowledge acquisition and the last two objectives are related to tasks that involve textual inference. In the following lines we summarize our objectives and the outcomes achieved.

Objective 1: Build a conceptual model of the meaning of a sentence though deep processing of text. Represent documents in a meaning representation that expresses the information contained on plain text. In particular, explore a graph-based representation that is capable of representing documents at document level and includes syntactic, semantic and temporal relations.

The first problem for language interpretation is to choose a meaning representation for encoding knowledge from text and define a way to transform text into such representation. Semantic graphs are a promising structure to encode semantics since they allow that elements of the semantic representation participate as arguments of an unbounded number of parents, without taking into consideration how far these elements may be on the text. We have developed a method to transform plain text documents into a semantically enriched morphosyntactic graphs by combining off-the-shelve tools for deep linguistic processing and our own method for semantic enrichment. We have evaluated the graph-based representation in both regular and temporal slot filling tasks.

- ***Objective 1.1:*** Design and implement a methodology to represent documents as semantically enriched morphosyntactic graphs.

  Our methodology transforms documents into graphs in two steps that produce two representations, the initial and the enriched representation. In the first step, we combine off-the-shelve text processing tools to build a dependency tree annotated with extra information such as POS tags, lemmas, named entities types, coreferences, temporal expressions, event recognition and temporal links. The second step uses a manually defined set of rules to collapse discourse referents related by coreference, refine the resulting graphs to ensure consistency, add extra semantic information and normalize syntactic relations. The result is a enriched representation where information is simplified and condensed for knowledge acquisition purposes.

- ***Objective 1.2:*** Build a collection of document represented as graphs as a resource for feature extraction and proposition extraction.

  We have applied our graph-based representation to seven corpora with different characteristics, both in size and in domain. In total, we have produced more than 11 million enriched graphs that were used for proposition extraction. Specifically we have used The TAC 2012 KBP Source Corpus for feature extraction for the regular and the temporal slot filling tasks.

- ***Objective 1.3:*** Study the generation of new features.

  Compared with regular dependency parsers, path features are modified because of the graph representation. The naïve role labelling and the normalization aim to simplify the paths extracted, thus learning should be easier. We have used a set of unique features that correspond to the new semantic relations extracted, and specifically for temporal slot filling, we have used the features related to verb tense, aspect and polarity.

- ***Objective 1.4:*** Extrinsic evaluation in relation extraction.

  We have participated in the 2011 Slot Filling task of the Knowledge Base Population track. Results show slightly better results using the enriched graphs, so we conclude that the balance between the gain of collapsing the graphs and the loss of the added errors introduced by the process is positive.

Objective 2: Automatically acquire knowledge from the graph-based representation. Extract knowledge into propositions, which are predicate-argument structures that express open relations.

The second problem for language interpretation is to automatically distil knowledge to be used as background knowledge. This problem requires to define a knowledge acquisition method that, given a large corpus mapped into a meaning representation, produces a target knowledge representation. In order to be used as source of background knowledge, this knowledge has to be massive, and include general domain information.

- ***Objective 2.1:*** Design and implement a methodology to automatically extract propositions from the graph-based representation.

  Our method uses large collections of documents represented as graphs to automatically extract propositions using syntactic and semantic patterns. The graph-based representation fulfils two goals. First, it is appropriate for being used to structure large amounts of documents, and second, it affects to the extraction of the propositions by facilitating the acquisition of long-distance dependencies and adding extra information such as named entity types and semantic relations. Structuring knowledge into propositions is desirable because, like human language, propositions express semantics through predicate-argument structures, and at the same time, propositions enable textual inferences.

- ***Objective 2.2:*** Build a proposition store from the collection of graphs.

  We have used six corpora represented with the enriched graph-based representation to acquire propositions from more than 11 million documents into three kinds of proposition stores, regular propositions, semantic class propositions and typed propositions. In total, we have acquired more than 1,970 million propositions.

- ***Objective 2.3:*** Demonstrate the application to named entity disambiguation.

  The One Sense Per Collocation hypothesis states that a word acts with the same sense when occurring in the same collocate, whether the collocate is positional or syntactic. Although this hypothesis has been empirically tested for word sense disambiguation, it was not clear if it holds on the more restricted task of named entity disambiguation. We have used our method for propositional knowledge acquisition to create a corpus of propositions with disambiguated named entities that represent collocations of ambiguous entities. This corpus has been used to demonstrate that One sense per proposition also holds for named entity disambiguation.

Objective 3: Improve apposition dependency detection using knowledge about semantic classes. Measure the semantic compatibility between the two sides of an apposition in order to decide the most suitable governor.

Appositives are syntactic dependencies that are often used to describe a class-instance relation between two noun phrases. Appositive detection is not trivial because dependencies can relate terms that are far on the text, and parsers fail to choose when there are several grammatically correct candidates if they lack of the semantic information needed. From a NLU perspective, solving the ambiguity is crucial to perform a correct interpretation of the sentence, and we show how to correct the output of a standard dependency parser by using propositional knowledge as a source of background knowledge.

- ***Objective 3.1:*** Define the problem of apposition correction and classify the errors that parsers commit.

  In order to perform apposition correction, we consider the output of a standard dependency parser where an apposition is detected and the sentence fulfil two premises: First, the first noun phrase has more than one common noun, and second, the nucleus of the second noun phrase is a named entity. Every noun in the first phrase is considered a candidate to govern the apposition, and the task is to decide which candidate is better. We have estimated that 78.6% of errors on apposition parsing are due of the lack of semantic information, and have categorized them on three classes: appositions with one valid candidate (70.6%), with several valid candidates (7.3%) or with undecidable candidates (0.6%). We have focus on solving these three cases.

- ***Objective 3.2:*** Build a gold standard with conflicting cases of apposition detection and manually annotate the correct relation.

  We have built a gold standard with 300 sentences that were annotated with one apposition by a dependency parser and had at least two nouns candidates to govern the apposition. We have discarded 64 sentences since there was no apposition, and the remaining 236 sentences were manually annotated with the correct governor.

- ***Objective 3.3:*** Design and implement a method for apposition correction based on background knowledge.

  Since appositives are often used to denote class-instance relation, we measure the semantic compatibility between candidates and named entities to decide which candidate is better. Semantic class propositions provides two different kinds of evidence, the compatibility between semantic classes and entity names, and the compatibility between semantic classes and entity types. We combine the evidence with three different measures (normalized pointwise mutual information, conditional probabilities and smoothed conditional probabilities) to define 8 configurations that evaluate the compatibility considering an evidence and a measure.

- ***Objective 3.4:*** Evaluate the apposition correction method in the gold standard and compare to parsers in the state of the art.

  Our method yields the best results when combining both sources of evidence (relation between semantic classes and entity names, and relation between semantic classes and entity types) with smoothed conditioned probabilities. We reach a 91.4% accuracy which is a 12.9% of relative improvement with respect to the best baseline (80.9%), which corresponds to the Stanford Parser.

Objective 4: Analyse the impact of the mapping from utterances into semantic relations for semantic parsing over linked data. Align propositions to linked data properties to feed a semantic parser.

Semantic parsing can be seen as a particular flavour of language interpretation, in the sense that natural language has to be mapped into a formal meaning representation. Mapping propositions into a structured knowledge base such as linked data databases can be seen as a way to bridge the gap between Open Information Extraction and structured ontologies. We study how this mapping can be used in Question Answering systems and what is the relative importance of this step compared with the rest of the system.

- ***Objective 4.1:*** Design and implement a methodology to map from propositions into linked data properties.

  We have developed a method to map propositions into Freebase properties using distant supervision that helps in solving both lexical and structural gaps by finding multiple grammatical structures and lexical realizations of the same query.

  Our method is divided in three steps: (1) Build a proposition store. To do so, we select relevant sentences, transform them into graphs from which we extract propositions. (2) Ground each proposition by pairing them with KB properties considering linked entities, and (3) Compute the global weights of each pairing. We consider an optional step: (4) Perform a lexical expansion by creating new pairs and re-evaluate the weights of the lexicon entries.

- ***Objective 4.2:*** Study the contribution of knowledge acquisition on closing the gap between natural language and linked data relations. Isolate the evaluation of the knowledge acquisition from other typical steps in semantic parsing such as training and querying. Evaluate which syntactic structures should be aligned and what is the impact of each one.

  Building and grounding the proposition stores is key to the final performance of the semantic parser. A system with an empty alignment lexicon achieves a 7.80% of F1-measure. In baseline systems with lexicon but without training, our experiments show that the lexicon contributes with near 80% of the results, and training only accounts for 7.6% of relative improvement.

We study the contribution of thirteen extraction patterns to the final results by performing and ablation test. The result show that most patterns are useful, and that removing the harmful patterns can provide a small boost on the results while reducing the cost the acquisition step.

- ***Objective 4.3:*** Study whether external linguistic resources are useful to reduce the lexical gap in the context of mapping propositions to linked data relations.

  We have shown how to enrich the lexicon using linguistic information from an external resource (i.e. WordNet), helping to bridge the lexical gap between utterances and database queries. Enrichment consistently pushes the results in every case, in a range from 0.8% to 2.50% of absolute improvement.

7.3 Looking Forward

The use of propositional knowledge for textual inferences is still an open research field, with many research opportunities in all the areas involved. In this thesis we have shown how to acquire and use propositions in syntactic and in semantic parsing, and develop completely unrelated methods for each case. In the following sections we point to some specific future avenues divided by the area of study.

7.3.1 Meaning Representation

The graph-based representation presents many opportunities for improvement. First, we can take advantage of the information that is hidden on the text, but that people recover effortlessly (Peñas and Ovchinnikova, 2012). Authors do not include information that they assume that their readers know, because its inclusion would mean an extra importance. The problem is that automatic systems cannot recover this information because they lack of the background knowledge necessary and the inference capabilities to use it.

One way to do it is to use type coercion to substitute general syntactic edges of structures that tend to contain this information, such as genitives or nominal compounds, for other edges more specific that point to the nature of the relation between the components could help to improve the information extraction systems.

Other opportunity of improvement is to include a system of event coreference (Humphreys et al., 1997; Hasler et al., 2006), and perform a collapsing to the one of the nominal compounds. This technique aims to improve the cohesion of the information with the goal of providing a better base for proposition extraction.

As we explained in the Chapter 5, we have used semantics to solve structural ambiguities after using syntactic dependencies to build the semantic representation. This is a recursive problem: the more we improve the syntactic correction, the better semantic representation. Once we get the new apposition dependencies for a large corpus, we can repeat the process of knowledge acquisition, creating a bootstrap method that iteratively improves the dependence analysis and semantic class acquisition. This process could be further refined if the process of correction is extended to other relations that depend on semantic compatibility, such as coreference resolution.

In this line, we could aim to correct dependencies in standard repositories of syntactic trees. For example, we could rebank the Penn Treebank corpus (Marcus et al., 1993), focusing on long distant relations, and establish a new standard for syntactic parsing evaluation.

7.3.2 Knowledge Acquisition

Regarding our method for knowledge acquisition, we initially propose an acquisition based on our graph representation. Another approach would be to swap this representation for another recent meaning representation. Specifically, it would be interesting to see what kind of propositions can be gathered using automatically generated AMR graphs. Theoretically, AMR would produce cleaner propositions, however since the AMR parsers are still under development, if the AMR generation is noisy, those mistakes would be carried to the propositions.

Another line of research is related to the estimation of the frequencies of the propositions. So far, we have explored the uses of a simple aggregation of occurrences, but it is interesting to study whether a more sophisticated probabilistic model can be useful for language interpretation or at least for specific NLP tasks.

7.3.3 Textual Inference

Address all textual inferences is a work out of the scope of this dissertation. However, we think Proposition Stores could be the base for general methods able to support many different textual inferences at the same time such as coreference, metonymy and type coercion. Possibly, the most interesting research line is to find a method that unifies the textual inference step for several tasks. For instance, a method whose output can be used to improve syntactic and semantic parsing at the same time. In this line, the most promising methods are related to end-to-end deep learning methods such as neural networks (Collobert et al., 2011; Zhang and LeCun, 2015).

In the case of our method for appositive correction could benefit from adding extra semantic information. For example, we could characterize entities with gender or age, to get more accurate measures of semantic compatibility. For example:

David meets a friend of his wife, ***Peter****.*

Knowing that *wife* is a class used with females and *Peter* is usually a male name leads us to consider the link between *friend* and *Peter* more appropriate.

Although this work focuses on appositive dependencies, it would be interesting to study whether this technique could be extrapolated to solve other dependency types, such as abbreviations and copulative verbs, or even other relations such as coreferences.

Finally, we could aim to include this method as a feature of a dependency parser. The main issue to study is how to add the semantic information without hurting the parser efficiency.

We can also aim to scale up the semantic parsing system. One option would be to generalize the GPS using bigger knowledge bases, which now are small because entities are required to be linked to Freebase. The hypothesis of *one sense per proposition* defined in (Barrena et al., 2014) could help to automate entity linking.

We leave also for future work refining the distant supervision process. For example, we could follow a similar approach as Surdeanu et al. (2012), that try to reduce the noise by jointly modelling instances and labels for relation extraction. This would be equivalent to our problem, where we could model utterances and labels at the same time.

Finally, it would be interesting to study how to improve GPS using distributed representations at word level like Word2Vec (Mikolov et al., 2013b) or GloVe (Pennington et al., 2014), at relation level (Lewis and Steedman, 2013; Ji and Eisenstein, 2015), or embeddings for QA (Zhou et al., 2015, 2016). Whereas they have proven to be useful for training (Kumar et al., 2015; Li and Clark, 2015), to the best of our knowledge there is not any effort to enrich the acquisition step.

7.4 Contributions

As a result of this work we have obtained several useful contributions for the research community:

1. **A new paradigm that relates meaning representations and structured knowledge bases trough propositional knowledge:** A new proposal on how to acquire and use propositional knowledge for language interpretation

purposes. This paradigm opens new research lines, some of them are explored in this thesis.

2. **A method to turn plain text into a graph-based representation:** A method that uses free distributed tools to develop two different representations, the initial representation and the enriched representation.
3. **A procedure for automatic proposition extraction from graphs:** A method that finds syntactic and semantic patterns to find and extract propositions in the graph-based representation.
4. **Several collections of documents represented as graphs:** We have transformed into graphs documents from 6 different corpora, variable both in size and in domain. In total, we have built more than 11.5 million graphs. We have implemented two different outputs, DOT and pseudo-JSON.
5. **Several proposition stores extracted from graphs:** Proposition stores with knowledge about predicate argument relations, semantic classes and typed propositions with more than 1,970 million instances.
6. **A gold standard for apposition parsing:** 236 sentences with multiple candidates to govern an apposition and a manual disambiguation.
7. **A study of cases of ambiguous appositive relations:** A classification of sentences with multiple candidates to govern an apposition according to the semantic compatibility of the candidates to governor and the dependent part.
8. **An unsupervised method for the correction of appositive dependencies:** This method finds grammatical ambiguous cases where there are two or more candidates to be the governor of an apposition. Then, it chooses the best candidate according to the semantic compatibility with the dependent part. We have compared several sources of evidence, different semantic compatibility measures and its combinations, concluding which is the best approach.
9. **A distant supervised method for mapping propositions to linked data properties:** This method pairs propositions with KB relations considering linked entities, and can be improved performing a lexical expansion using external linguistic resources.
10. **A study of the relevance of syntactic and semantic structures on the task of building a mapping to linked data properties:** We study which syntactic and semantic structures should be aligned to database relations and what is the impact of each one when building a lexicon for semantic parsing.
11. **A methodology to evaluate the grounding step in semantic parsing:** A method that enables the evaluation of the grounding step in isolation, without any influence of training or querying.

Bibliography

Rodrigo Agerri and Anselmo Peñas. 2010. On the automatic generation of intermediate logic forms for wordnet glosses. In *Computational Linguistics and Intelligent Text Processing*, Springer, pages 26–37.

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*. ACM, pages 85–94.

Eneko Agirre, Timothy Baldwin, and David Martinez. 2008. Improving Parsing and {PP} Attachment Performance with Sense Information. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, Columbus, Ohio, pages 317–325.

Eneko Agirre, Kepa Bengoetxea, Koldo Gojenola, and Joakim Nivre. 2011. Improving dependency parsing with semantic classes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '11, pages 699–703.

Eneko Agirre and David Martinez. 2001. Learning class-to-class selectional preferences. In *Proceedings of the 2001 workshop on Computational Natural Language Learning-Volume 7*. Association for Computational Linguistics, page 3.

Henry Anaya-Sánchez, Juan del Rosal, and Anselmo Penas. 2015. Unsupervised learning of meaningful semantic classes for entity aggregates. *IWCS 2015* page 88.

Henry Anaya-Sánchez and Anselmo Peñas. 2015. Unsupervised induction of meaningful semantic classes through selectional preferences. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, pages 361–371.

Henry Anaya-Sanchez, Anselmo Peñas, and Bernardo Cabaleiro. 2013. UNED-READERS: Filtering Relevant Tweets using Probabilistic Signature Models. In Pamela Forner, Roberto Navigli, and Dan Tufis, editors, *CLEF 2013 Evaluation Labs and Workshop. Online Working Notes.*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. *CoRR* abs/1601.01705.

Javier Artiles, Qi Li, Taylor Cassidy, Suzanne Tamang, and Heng Ji. 2011. Cuny blender tac-kbp2011 temporal slot filling system description. In *Proceedings of the Text Analysis Conference*.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1699–1710.

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 421–432.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. *Dbpedia: A nucleus for a web of open data*. Springer.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 86–90.

James K Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America* 65(S1):S132–S132.

Jason Baldridge and Alex Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL '05, pages 96–103.

Timothy Baldwin, Mark Dras, Julia Hockenmaier, Tracy Holloway King, and Gertjan van Noord. 2007. The impact of deep linguistic processing on parsing technology. In *Proceedings of the 10th International Conference on Parsing Technologies*. Association for Computational Linguistics, pages 36–38.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2012. Abstract meaning representation (amr) 1.0 specification. In *Parsing on Freebase from Question-Answer Pairs." In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle: ACL*. pages 1533–1544.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, Sofia, Bulgaria, pages 178–186.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*. volume 7, pages 2670–2676.

Ander Barrena, Eneko Agirre, Bernardo Cabaleiro, Anselmo Peñas, and Aitor Soroa. 2014. "One Entity per Discourse" and "One Entity per Collocation" Improve Named-Entity Disambiguation. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 2260–2269.

Hannah Bast and Elmar Haussmann. 2013. Open information extraction via contextual sentence decomposition. In *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on*. IEEE, pages 154–159.

Emily M. Bender, Dan Flickinger, Stephan Oepen, and Yi Zhang. 2011. Parser evaluation over local and non-local deep dependencies in a large corpus. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK., pages 397–408.

Samy Bengio and Georg Heigold. 2014. Word embeddings for speech recognition. In *INTERSPEECH*. pages 1053–1057.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*. pages 1533–1544.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of ACL*. volume 7, page 92.

Sundheim Beth. 1995. Proceedings of the sixth message understanding conference. Columbia, MD, MUC-6.

Steven Bethard and James H Martin. 2006. Identification of event mentions and their semantic class. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 146–154.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. " O'Reilly Media, Inc.".

Christian Bizer, Tom Heath, and Tim Berners-Lee. 2009. Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts* pages 205–227.

James Blythe, Jerry R Hobbs, Pedro Domingos, Rohit J Kate, and Raymond J Mooney. 2011. Implementing weighted abduction in markov logic. In *Proceedings of the Ninth International Conference on Computational Semantics*. Association for Computational Linguistics, pages 55–64.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, pages 1247–1250.

Taylor L Booth and Richard A Thompson. 1973. Applying probability measures to abstract languages. *IEEE transactions on Computers* 100(5):442–450.

Dan Brickley and Libby Miller. 2012. Foaf vocabulary specification 0.98. *Namespace document* 9.

Eric Brill, Susan Dumais, and Michele Banko. 2002. An analysis of the AskMSR question-answering system. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, pages 257–264.

Sergey Brin. 1999. Extracting patterns and relations from the world wide web. In *The World Wide Web and Databases*, Springer, pages 172–183.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 149–164.

Cristina Butnariu, Su Nam Kim, Preslav Nakov, Diarmuid Ó Séaghdha, Stan Szpakowicz, and Tony Veale. 2009. Semeval-2010 task 9: The interpretation of noun compounds using paraphrasing verbs and prepositions. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*. Association for Computational Linguistics, pages 100–105.

Bernardo Cabaleiro and Anselmo Peñas. 2012. Representación Gráfica de Documentos para Extracción Automática de Relaciones. *Procesamiento del Lenguaje Natural* 49(0).

Bernardo Cabaleiro and Anselmo Peñas. 2013. Corrección no Supervisada de Dependencias Sintácticas de Aposición mediante Clases Semánticas. *Procesamiento del Lenguaje Natural* 51(0).

Bernardo Cabaleiro and Anselmo Peñas. 2015. On Improving Parsing with Automatically Acquired Semantic Classes. *Knowledge-Based Systems* 89(C):359–365.

Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment* 1(1):538–549.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL (1)*. Citeseer, pages 423–433.

Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. 2009. Clueweb09 data set.

Erik Cambria, Daniel Olsher, and Dheeraj Rajagopal. 2014. Senticnet 3: a common and common-sense knowledge base for cognition-driven sentiment analysis. In *Proceedings of the twenty-eighth AAAI conference on artificial intelligence*. AAAI Press, pages 1515–1521.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

Bob Carpenter. 1997. *Type-logical semantics*. MIT press.

Angel X Chang and Christopher Manning. 2012. SUTime: A library for recognizing and normalizing time expressions. In *LREC*. pages 3735–3740.

Eugene Charniak and Robert P Goldman. 1989. A semantics for probabilistic quantifier-free first-order languages, with particular application to story understanding. In *IJCAI*. Citeseer, volume 89, pages 1074–1079.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 173–180.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*. pages 740–750.

David L Chen and Raymond J Mooney. 2011. Learning to interpret natural language navigation instructions from observations. *San Francisco, CA* pages 859–865.

Desai Chen, Nathan Schneider, Dipanjan Das, and Noah A Smith. 2010. Semafor: Frame argument resolution with log-linear models. In *Proceedings of the 5th international workshop on semantic evaluation*. Association for Computational Linguistics, pages 264–267.

Jinying Chen and Martha S Palmer. 2009. Improving english verb sense disambiguation performance with linguistically motivated features and clear sense distinction boundaries. *Language Resources and Evaluation* 43(2):181–208.

Nancy Chinchor. 1998. Overview of MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. pages 178–185.

Janara Christensen, Stephen Soderland Mausam, Stephen Soderland, and Oren Etzioni. 2013. Towards coherent multi-document summarization. In *HLT-NAACL*. Citeseer, pages 1163–1173.

Kenneth Church and Ramesh Patil. 1982. Coping with syntactic ambiguity or how to put the block in the box on the table. *Comput. Linguist.* 8(3-4):139–149.

Massimiliano Ciaramita. 2007. Dependency Parsing with Second-Order Feature Maps and Annotated Semantic Information. In *Proc. of the 12th International Workshop on Parsing Technologies (IWPT)*.

Kevin Clark and Christopher D Manning. 2015. Entity-centric coreference resolution with model stacking. In *Association of Computational Linguistics (ACL)*.

Peter Clark and Phil Harrison. 2009. Large-scale extraction and use of knowledge from text. In *Proceedings of the fifth international conference on Knowledge capture*. ACM, New York, NY, USA, K-CAP '09, pages 153–160.

Stephen Clark and James R Curran. 2004. Parsing the wsj using ccg and log-linear models. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 103.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational linguistics* 29(4):589–637.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

James R Curran, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, pages 33–36.

Jon Curtis, John Cabral, and David Baxter. 2006. On the application of the cyc ontology to word sense disambiguation. In *FLAIRS Conference*. pages 652–657.

Jon Curtis, Gavin Matthews, and David Baxter. 2005. On the effective use of cyc in a question answering system. In *Proc Workshop on Knowledge and Reasoning for Answering Questions*. pages 61–70.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, Springer, pages 177–190.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, and Others. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*. volume 6, pages 449–454.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. *URL http://nlp. stanford. edu/software/dependencies manual. pdf* .

Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. 2015. Analysis of named entity recognition and linking for tweets. *Information Processing & Management* 51(2):32–49.

Pamela Downing. 1977. On the creation and use of english compound nouns. *Language* pages 810–842.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075* .

Myroslava O Dzikovska, Rodney D Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and Hoa T Dang. 2013. Semeval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge. Technical report, DTIC Document.

Oren Etzioni, Michele Banko, and Michael J Cafarella. 2006. Machine reading. In *AAAI*. volume 6, pages 1517–1519.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th International Conference on World Wide Web*. ACM, New York, NY, USA, WWW '04, pages 100–110.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence* 165(1):91–134.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*. AAAI Press, pages 3–10.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1535–1545.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013a. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1608–1618.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 1156–1165.

Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. 2013b. Paraphrase-driven learning for open question answering. In *ACL (1)*. Citeseer, pages 1608–1618.

James Fan, Aditya Kalyanpur, David C Gondek, and David A Ferrucci. 2012. Automatic knowledge extraction from documents. *IBM Journal of Research and Development* 56(3.4):5–1.

D Fass. 1991. met*: A method for discriminating metonymy and metaphor by computer. *Computational Linguistics* .

Benoit Favre and Dilek Hakkani-Tür. 2009. Phrase and word level strategies for detecting appositions in speech. In *INTERSPEECH*. pages 2711–2714.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, and Others. 2010. Building Watson: An overview of the DeepQA project. *AI magazine* 31(3):59–79.

Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. 1999. Hypertext transfer protocol–http/1.1. Technical report.

Jenny Rose Finkel, Trond Grenager, and Christopher D Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL ’05, pages 363–370.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 1426–1436.

Noah S Friedland, Paul G Allen, Gavin Matthews, Michael Witbrock, David Baxter, Jon Curtis, Blake Shepard, Pierluigi Miraglia, Jurgen Angele, Steffen Staab, et al. 2004. Project halo: Towards a digital aristotle. *AI magazine* 25(4):29.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0) 5.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the Workshop on Speech and Natural Language*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT ’91, pages 233–237.

Guillermo Garrido, Bernardo Cabaleiro, Anselmo Peñas, Ávaro Rodrigo, and Damiano Spina. 2011. Distant supervised learning for the TAC KBP Slot Filling and Temporal Slot Filling Tasks. In *Text Analysis Conference, TAC 2011,Workshop, Notebook Papers*.

Guillermo Garrido, Anselmo Peñas, Bernardo Cabaleiro, and Álvaro Rodrigo. 2012. Temporally Anchored Relation Extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL ’12, pages 107–116.

Guillermo Garrido, Anselmo Penas, and Bernardo Cabaleiro. 2013. UNED Slot Filling and Temporal Slot Filling systems at TAC KBP 2013: System description. In *TAC*.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics* 28(3):245–288.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. Semeval-2007 task 04: Classification of semantic relations between nominals. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. Association for Computational Linguistics, pages 13–18.

Jonathan Gordon and Lenhart K Schubert. 2012. Using textual patterns to learn expected event frequencies. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. Association for Computational Linguistics, pages 122–127.

Jr. Bert F. Green, Alice K. Wolf, Carol Chomsky, and Kenneth Laughery. 1961. Baseball: An automatic question-answerer. In *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*. ACM, New York, NY, USA, IRE-AIEE-ACM '61 (Western), pages 219–224.

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference-6: A Brief History. In *COLING*. volume 96, pages 466–471.

David Gunning, Vinay K Chaudhri, Peter E Clark, Ken Barker, Shaw-Yi Chaw, Mark Greaves, Benjamin Grosof, Alice Leung, David D McDonald, Sunil Mishra, et al. 2010. Project halo update—progress toward digital aristotle. *AI Magazine* 31(3):33–58.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H Martin, and Daniel Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *CoNLL*. pages 110–113.

Jan Hajič, Jan Raab, Miroslav Spousta, et al. 2009. Semi-supervised training for the averaged perceptron pos tagger. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 763–771.

Sherzod Hakimov, Christina Unger, Sebastian Walter, and Philipp Cimiano. 2015. Applying semantic parsing to question answering over linked data: Addressing the lexical gap. In *International Conference on Applications of Natural Language to Information Systems*. Springer, pages 103–109.

Zellig S Harris. 1954. Distributional structure. *Word* 10(2-3):146–162.

Laura Hasler, Constantin Orasan, and Karin Naumann. 2006. Nps for events: Experiments in coreference annotation. In *Proceedings of the 5th edition of the International Conference on Language Resources and Evaluation (LREC2006)*. Citeseer, pages 1167–1172.

Oktie Hassanzadeh and Mariano P Consens. 2009. Linked movie data base. In *LDOW*.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '92, pages 539–545.

Irene Heim. 1983. *File Change Semantics and the Familiarity Theory of Definiteness*, Walter de Gruyter, pages 164–189.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*. Association for Computational Linguistics, pages 94–99.

Iris Hendrickx, Zornitsa Kozareva, Preslav Nakov, Diarmuid O Séaghdha, Stan Szpakowicz, and Tony Veale. 2013. Semeval-2013 task 4: Free paraphrases of noun compounds. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. pages 138–143.

Jerry R Hobbs. 1985. Ontological promiscuity. In *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 60–69.

Jerry R Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards. 1988. Interpretation as abduction. In *Proceedings of the 26th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 95–103.

Jerry R Hobbs, Mark E Stickel, Douglas E Appelt, and Paul Martin. 1993. Interpretation as abduction. *Artificial Intelligence* 63(1-2):69–142.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press, pages 3161–3165.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 782–792.

Eduard Hovy, Ulf Hermjakob, and Deepak Ravichandran. 2002. A question/answer typology with surface text patterns. In *Proceedings of the second international conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., pages 247–251.

Kevin Humphreys, Robert Gaizauskas, and Saliha Azzam. 1997. Event coreference for information extraction. In *Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*. Association for Computational Linguistics, pages 75–81.

Heng Ji, Ralph Grishman, and Hoa Dang. 2011. Overview of the TAC2011 Knowledge Base Population Track. In *TAC 2011 Proceedings Papers*.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association for Computational Linguistics* 3:329–344.

David A Jurgens, Peter D Turney, Saif M Mohammad, and Keith J Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 356–364.

Michael Kaisser and Bonnie Webber. 2007. Question answering based on semantic roles. In *Proceedings of the Workshop on Deep Linguistic Processing*. Association for Computational Linguistics, pages 41–48.

Hans Kamp. 1988. Discourse representation theory. In *IBM Germany Scientific Symposium Series*. Springer, pages 84–111.

Hans Kamp and Uwe Reyle. 1993. From discourse to logic; introduction to the modeltheoretic semantics of natural language .

Ronald M Kaplan and Joan Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representation. *Formal Issues in Lexical-Functional Grammar* pages 29–130.

Lauri Karttunen. 1968. *What do referential indices refer to?*. Rand Corporation: [Paper]. Rand Corp.

Rohit J Kate and Raymond J Mooney. 2009. Rj: Probabilistic abduction using markov logic networks. In *In: IJCAI-09 Workshop on Plan, Activity, and Intent Recognition*. Citeseer.

Jerrold J Katz and Jerry A Fodor. 1963. The structure of a semantic theory. *language* 39(2):170–210.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '03, pages 423–430.

Alexander Koller. 2015. Semantic construction with graph grammars. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS)*. pages 228–238.

Oleksandr Kolomiyets, Parisa Kordjamshidi, Steven Bethard, and Marie-Francine Moens. 2013. Semeval-2013 task 3: Spatial role labeling. In *Second joint conference on lexical and computational semantics (* SEM), Volume 2: Proceedings of the seventh international workshop on semantic evaluation (SemEval 2013)*. ACL, pages 255–266.

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, Columbus, Ohio, pages 1048–1056.

Robert Krovetz. 1998. More than one sense per discourse. *NEC Princeton NJ Labs., Research Memorandum* .

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR* abs/1506.07285.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching .

Joel Lang and Mirella Lapata. 2014. Similarity-driven semantic role induction via graph partitioning. *Computational Linguistics* 40(3):633–669.

Irene Langkilde and Kevin Knight. 1998. Generating word lattices from abstract meaning representation. Technical report, Technical report, Information Science Institute, University of Southern California.

Egoitz Laparra and German Rigau. 2013. Impar: A deterministic algorithm for implicit semantic role labelling. In *ACL (1)*. pages 1180–1189.

Mirella Lapata, Frank Keller, and Christoph Scheepers. 2003. Intra-sentential context effects on the interpretation of logical metonymy. *Cognitive Science* 27(4):649–668.

Mirella Lapata and Alex Lascarides. 2003. A probabilistic account of logical metonymy. *Computational Linguistics* 29(2):261–315.

Ora Lassila and Ralph R Swick. 1999. Resource description framework (rdf) model and syntax specification. Technical report.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Comput. Linguist.* 39(4):885–916.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011a. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL Shared Task '11, pages 28–34.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011b. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL Shared Task '11, pages 28–34.

Douglas Lenat, Michael Witbrock, David Baxter, Eugene Blackstone, Chris Deaton, Dave Schneider, Jerry Scott, and Blake Shepard. 2010. Harnessing cyc to answer clinical researchers' ad hoc queries. *AI Magazine* 31(3):13–32.

Douglas B Lenat. 1995. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11):33–38.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*. volume 46, page 47.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *KR*.

Mike Lewis and Mark Steedman. 2013. Combining distributional and logical semantics. *Transactions of the Association for Computational Linguistics* 1:179–192.

Yang Li and Peter Clark. 2015. Answering elementary science questions by constructing coherent scenes using background knowledge. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2007–2012.

Percy Liang. 2015. Learning semantic parsers for natural language understanding .

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics* 39(2):389–446.

Dekang Lin. 2003. Dependency-based evaluation of MINIPAR. In *Treebanks*, Springer, pages 317–329.

Dekang Lin and Patrick Pantel. 2001a. Dirt@ sbt@ discovery of inference rules from text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 323–328.

Dekang Lin and Patrick Pantel. 2001b. Induction of semantic classes from natural language text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, New York, NY, USA, KDD '01, pages 317–322.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering* 20(02):151–184.

Hugo Liu and Push Singh. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal* 22(4):211–226.

Steven L Lytinen. 1992. Conceptual dependency and its descendants. *Computers & Mathematics with Applications* 23(2-5):51–73.

Bill MacCartney and Christopher D Manning. 2009. An extended model of natural logic. In *Proceedings of the eighth international conference on computational semantics*. Association for Computational Linguistics, pages 140–156.

Inderjeet Mani, Barry Schiffman, and Jianping Zhang. 2003. Inferring temporal ordering of events in news. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL-Short '03, pages 55–57.

Inderjeet Mani and George Wilson. 2000. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 69–76.

Christopher D. Manning. 2016. Understanding human language: Can nlp and deep learning help? In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, pages 1–1.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330.

Erwin Marsi, Emiel Krahmer, and Wauter Bosma. 2007. Dependency-based paraphrasing for recognizing textual entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. Association for Computational Linguistics, pages 83–88.

James H Martin and Daniel Jurafsky. 2000. Speech and language processing. *International Edition* 710.

David Martinez and Eneko Agirre. 2000. One sense per collocation and genre/topic variations. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*. Association for Computational Linguistics, pages 207–215.

André FT Martins and Mariana SC Almeida. 2014. Priberam: A turbo semantic parser with second order features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. pages 471–476.

Larry Masinter, Tim Berners-Lee, and Roy T Fielding. 2005. Uniform resource identifier (uri): Generic syntax .

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. 2013. Learning to parse natural language commands to a robot control system. In *Experimental Robotics*. Springer, pages 403–415.

Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, pages 4074–4077.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP-CoNLL '12, pages 523–534.

Jonathan May. 2016. Semeval-2016 task 8: Meaning representation parsing. *Proceedings of SemEval* pages 1063–1073.

Diana Maynard, Kalina Bontcheva, and Hamish Cunningham. 2003. Towards a semantic extraction of named entities. In *In Recent Advances in Natural Language Processing*.

Michael C McCord, J William Murdock, and Branimir K Boguraev. 2012. Deep parsing in Watson. *IBM Journal of Research and Development* 56(3.4):1–3.

Ryan T McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *EMNLP-CoNLL*. pages 122–131.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the tac 2009 knowledge base population track. In *Text Analysis Conference (TAC)*.

Igor Mel'cuk and Alain Polguère. 1987. A formal lexicon in the meaning-text theory (or how to do lexica with words). *Computational Linguistics* 13(3-4):261–275.

Charles F Meyer. 1992. *Apposition in contemporary English*. Cambridge University Press.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM* 38(11):39–41.

Tom M Mitchell, Justin Betteridge, Andrew Carlson, Estevam Hruschka, and Richard Wang. 2009. Populating the Semantic Web by Macro-reading Internet Text. Springer-Verlag, Berlin, Heidelberg, ISWC '09, pages 998–1002.

Dan Moldovan, Adriana Badulescu, Marta Tatu, Daniel Antohe, and Roxana Girju. 2004. Models for the semantic classification of noun phrases. In *Proceedings of the HLT-NAACL Workshop on Computational Lexical Semantics*. Association for Computational Linguistics, pages 60–67.

Diego Mollá. 2001. Ontologically promiscuous flat logical forms for nlp. In *Proceedings of IWCS-4*. pages 249–265.

Richard Montague. 1970. Universal grammar. *Theoria* 36(3):373–398.

Lawrence S Moss. 2010. Natural logic and semantics. In *Logic, Language and Meaning*, Springer, pages 84–93.

Rutu Mulkar, Jerry R Hobbs, Eduard Hovy, Hans Chalupsky, and Chin-Yew Lin. 2007. Learning by reading: Two experiments. In *Proceedings of IJCAI 2007 workshop on Knowledge and Reasoning for Answering Questions*. pages 287–296.

Preslav Nakov and Marti A. Hearst. 2006. Using verbs to characterize noun-noun relations. In *Artificial Intelligence: Methodology, Systems, and Applications*, Springer, pages 233–244.

Preslav Nakov and Marti A. Hearst. 2008. Solving relational similarity problems using the web as a corpus. In *ACL*. pages 452–460.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 104–111.

Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The conll 2007 shared task on dependency parsing. In *Proceedings of the CoNLL shared task session of EMNLP-CoNLL*. sn, pages 915–932.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. Association for Computational Linguistics, pages 50–57.

Joakim Nivre. 2005. *Inductive Dependency Parsing of Natural Language Text*. Ph.D. thesis, School of Mathematics and Systems Engineering, Växjö University.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. pages 1659–1666.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*. volume 6, pages 2216–2219.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. pages 63–72.

Tom O'Hara and Janyce Wiebe. 2009. Exploiting semantic role resources for preposition disambiguation. *Computational Linguistics* 35(2):151–184.

Marian Olteanu and Dan Moldovan. 2005. Pp-attachment disambiguation using large context. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '05, pages 273–280.

Ekaterina Ovchinnikova, Ross Israel, Suzanne Wertheim, Vladimir Zaytsev, Niloofar Montazeri, and Jerry Hobbs. 2014a. Abductive inference for interpretation of metaphors. In *Proceedings of the Second Workshop on Metaphor in NLP*. pages 33–41.

Ekaterina Ovchinnikova, Niloofar Montazeri, Theodore Alexandrov, Jerry R Hobbs, Michael C McCord, and Rutu Mulkar-Mehta. 2014b. Abductive reasoning with a large knowledge base for discourse processing. In *Computing Meaning*, Springer, pages 107–127.

Martha Palmer. 2009. Semlink: Linking propbank, verbnet and framenet. In *Proceedings of the generative lexicon conference*. pages 9–15.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics* 31(1):71–106.

Martha Palmer, Daniel Gildea, and Nianwen Xue. 2010. Semantic role labeling. *Synthesis Lectures on Human Language Technologies* 3(1):1–103.

Martha S Palmer, Deborah A Dahl, Rebecca J Schiffman, Lynette Hirschman, Marcia Linebarger, and John Dowding. 1986. Recovering implicit information. In *Proceedings of the 24th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 10–19.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition, june. *Linguistic Data Consortium, LDC2011T07* .

Anselmo Peñas, Bernardo Cabaleiro, and Mirella Lapata. 2014. Unsupervised Interpretation of Eventive Propositions. In *Proceedings of the 15th International Conference on Computational Linguistics and Intelligent Text Processing - Volume 8403*. Springer-Verlag New York, Inc., New York, NY, USA, CICLing 2014, pages 379–390.

Anselmo Peñas and Eduard Hovy. 2010. Filling knowledge gaps in text for machine reading. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, pages 979–987.

Anselmo Peñas and Ekaterina Ovchinnikova. 2012. Unsupervised acquisition of axioms to paraphrase noun compounds and genitives. In *(Ed.): CICLing 2012, Part I, LNCS 7181*. Springer-Verlag, pages 388–401.

Anselmo Peñas and Eduard Hovy. 2010. Semantic enrichment of text with background knowledge. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*. Association for Computational Linguistics, pages 15–23.

Anselmo Peñas, Eduard H Hovy, Pamela Forner, Álvaro Rodrigo, Richard FE Sutcliffe, Corina Forascu, and Caroline Sporleder. 2011. Overview of qa4mre at clef 2011: Question answering for machine reading evaluation. In *CLEF (Notebook Papers/Labs/Workshop)*. Citeseer, pages 1–20.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for amr parsing. *CoNLL 2015* page 32.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Carl Pollard. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Simone Paolo Ponzetto and Michael Strube. 2007. Deriving a large scale taxonomy from wikipedia. In *AAAI*. volume 7, pages 1440–1445.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *ACL (1)*. pages 933–943.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing english into abstract meaning representation using syntax-based machine translation. *Training* 10:218–021.

James Pustejovsky. 1991. The generative lexicon. *Computational linguistics* 17(4):409–441.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Others. 2003. The timebank corpus. In *Corpus linguistics*. volume 2003, page 40.

James Pustejovsky, Anna Rumshisky, Alex Plotnick, Elisabetta Jezek, Olga Batiukova, and Valeria Quochi. 2010. Semeval-2010 task 7: Argument selection and coercion. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 27–32.

Will Radford and James R. Curran. 2013. Joint apposition extraction with syntactic and semantic constraints. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 671–677.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher D. Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 492–501.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 147–155.

Marta Recasens. 2010. Coreference: Theory, annotation, resolution and evaluation.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics* 2:377–392.

Philip Resnik. 1993. Semantic classes and syntactic ambiguity. In *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '93, pages 278–283.

Philip Resnik. 1997. Selectional preference and sense disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How*. Washington, DC, pages 52–57.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 193–203.

Kirk Roberts and Sanda M Harabagiu. 2011. Unsupervised learning of selectional restrictions and detection of argument coercions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 980–990.

Dan Roth and Mark Sammons. 2007. Semantic and logical inference model for textual entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. Association for Computational Linguistics, pages 107–112.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. 2010. Semeval-2010 task 10: Linking events and their participants in discourse. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 45–50.

Roser Saurí, Robert Knippen, Marc Verhagen, and James Pustejovsky. 2005. Evita: a robust event recognizer for QA systems. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 700–707.

Roger C. Schank. 1972. Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology* 3(4):552–631.

Roger C. Schank and Robert P. Abelson. 1977. Scripts, plans, goals, and understanding: An inquiry into human knowledge structures (artificial intelligence series) .

Lenhart Schubert. 2002. Can we derive general world knowledge from texts? In *Proceedings of the second international conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., pages 94–97.

Lenhart Schubert. 2015. Semantic representation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'15, pages 4132–4138.

Karin Kipper Schuler. 2005. Verbnet: A broad-coverage, comprehensive verb lexicon .

Sebastian Schuster and Christopher D Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Satoshi Sekine and Chikashi Nobata. 2004. Definition, Dictionaries and Tagger for Extended Named Entity Hierarchy. In *LREC*.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *EMNLP-CoNLL*. pages 12–21.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *ACL*. volume 7, pages 760–767.

Parag Singla and Raymond J Mooney. 2011. Abductive markov logic for plan recognition. In *AAAI*. pages 1069–1075.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *ACL (1)*. pages 455–465.

Janara Christensen Stephen Soderland and Gagan Bansal Mausam. 2014. Hierarchical summarization: Scaling up multi-document summarization. In *Proceedings of the 52nd Annual Meeting of the Association for Computlational Linguistics*. pages 902–912.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics* 27(4):521–544.

David G Stork. 1999. The open mind initiative. *IEEE Expert Systems and Their Applications* .

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*. ACM, pages 697–706.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. Weakly supervised memory networks. *CoRR* abs/1503.08895.

Mihai Surdeanu, Sonal Gupta, John Bauer, David McClosky, Angel X. Chang, Valentin I. Spitkovsky, and Christopher D. Manning. 2011. Stanford's distantly-supervised slot-filling system. In *Proceedings of the Fourth Text Analysis Conference (TAC 2011)*. Gaithersburg, Maryland, USA.

Mihai Surdeanu and Christopher D Manning. 2010. Ensemble models for dependency parsing: cheap and good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 649–652.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP-CoNLL '12, pages 455–465.

Robert S Swier and Suzanne Stevenson. 2004. Unsupervised semantic role labelling. In *Proceedings of EMNLP*. volume 95, page 102.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 173–180.

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '00, pages 63–70.

Stephen Tratz and Eduard Hovy. 2011. A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1257–1268.

Alan M Turing. 1950. Computing machinery and intelligence. *Mind* 59(236):433–460.

Masao Utiyama, Masaki Murata, and Hitoshi Isahara. 2000. A statistical approach to the processing of metonymy. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pages 885–891.

Lucy Vanderwende, Arul Menezes, and Chris Quirk. 2015. An amr parser for english, french, german, spanish and japanese and a new amr-annotated corpus. In *Proceedings of NAACL-HLT*. pages 26–30.

Marc Verhagen, Inderjeet Mani, Roser Sauri, Robert Knippen, Seok Bae Jang, Jessica Littman, Anna Rumshisky, John Phillips, and James Pustejovsky. 2005. Automating temporal annotation with TARSQI. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*. Association for Computational Linguistics, pages 81–84.

Marc Verhagen and James Pustejovsky. 2008. Temporal processing with the tarsqi toolkit. In *22nd International Conference on on Computational Linguistics: Demonstration Papers*. Association for Computational Linguistics, pages 189–192.

Atro Voutilainen. 2003. Part-of-speech tagging. *The Oxford handbook of computational linguistics* pages 219–232.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM* 57(10):78–85.

Sebastian Walter, Christina Unger, Philipp Cimiano, and Daniel Bär. 2012. Evaluation of a layered approach to question answering over linked data. In *The Semantic Web–ISWC 2012*, Springer, pages 362–374.

Chuan Wang, Nianwen Xue, Sameer Pradhan, and Sameer Pradhan. 2015. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 366–375.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR* abs/1502.05698.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR* abs/1410.3916.

Mark Wick and Bernard Vatant. 2012. The geonames geographical database. *Available from World Wide Web: http://geonames. org* .

Yuk Wah Wong and Raymond J Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Annual Meeting-Association for computational Linguistics*. volume 45, page 960.

Fei Wu and Daniel S Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. ACM, pages 41–50.

Fei Wu and Daniel S Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 118–127.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, pages 481–492.

David Yarowsky. 1993. One sense per collocation. In *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, pages 266–271.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. Association for Computational Linguistics, pages 25–26.

Alexander Yates and Oren Etzioni. 2007. Unsupervised Resolution of Objects and Relations on the Web. In *HLT-NAACL*. pages 121–130.

Benat Zapirain, Eneko Agirre, Lluis Marquez, and Mihai Surdeanu. 2013. Selectional preferences for semantic role classification. *Computational Linguistics* 39(3):631–663.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*. pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *In Proceedings of the 21st Conference on Uncertainty in AI*. pages 658–666.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710* .

Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. 2015. Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of ACL*. pages 250–259.

Guangyou Zhou, Yin Zhou, Tingting He, and Wensheng Wu. 2016. Learning semantic representation with neural networks for community question answering retrieval. *Knowledge-Based Systems* 93:75–83.

A

Publications of the Author

The following papers correspond to partial advances in the resolution of this thesis. Papers are presented chronologically.

Journals:

I Bernardo Cabaleiro and Anselmo Peñas. 2015. On Improving Parsing with Automatically Acquired Semantic Classes. *Knowledge-Based Systems* 89(C):359–365. JCR Q1.

II Bernardo Cabaleiro, Anselmo Peñas and Suresh Manandhar. Grounding Proposition Stores for Question Answering over Linked Data. Submitted to *Knowledge-Based Systems*. JCR Q1.

Peer-reviewed Conferences:

III Bernardo Cabaleiro and Anselmo Peñas. 2012. Representación Gráfica de Documentos para Extracción Automática de Relaciones. *Procesamiento del Lenguaje Natural* 49(0).

IV Guillermo Garrido, Anselmo Peñas, Bernardo Cabaleiro, and Álvaro Rodrigo. 2012. Temporally Anchored Relation Extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '12, pages 107–116. Core A*.

V Bernardo Cabaleiro and Anselmo Peñas. 2013. Corrección no Supervisada de Dependencias Sintácticas de Aposición mediante Clases Semánticas. *Procesamiento del Lenguaje Natural* 51(0).

VI Ander Barrena, Eneko Agirre, Bernardo Cabaleiro, Anselmo Peñas, and Aitor Soroa. 2014. "One Entity per Discourse" and "One Entity per Collocation" Improve Named-Entity Disambiguation. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 2260–2269. Core A.

Workshop:

VII Guillermo Garrido, Bernardo Cabaleiro, Anselmo Peñas, Ávaro Rodrigo, and Damiano Spina. 2011. Distant supervised learning for the TAC KBP Slot Filling and Temporal Slot Filling Tasks. In *Text Analysis Conference, TAC 2011,Workshop, Notebook Papers.*

Papers I, II, III and V are mainly the work of the author. The main work of papers IV, VI and VII corresponds to G. Garrido and A. Barrena, where the author contributed with data processing, data analysis and discussion of results.

In addition, this work has been used to support the research on the following publications:

- Guillermo Garrido, Anselmo Penas, and Bernardo Cabaleiro. 2013. UNED Slot Filling and Temporal Slot Filling systems at TAC KBP 2013: System description. In *TAC*.
- Henry Anaya-Sanchez, Anselmo Peñas, and Bernardo Cabaleiro. 2013. UNED-READERS: Filtering Relevant Tweets using Probabilistic Signature Models. In Pamela Forner, Roberto Navigli, and Dan Tufis, editors, *CLEF 2013 Evaluation Labs and Workshop. Online Working Notes.*.
- Anselmo Peñas, Bernardo Cabaleiro, and Mirella Lapata. 2014. Unsupervised Interpretation of Eventive Propositions. In *Proceedings of the 15th International Conference on Computational Linguistics and Intelligent Text Processing - Volume 8403*. Springer-Verlag New York, Inc., New York, NY, USA, CICLing 2014, pages 379–390.

B

Graph-based Representation Example

In Chapter 3 we have presented our methodology to transform plain text documents into a graph based meaning representation. As a matter of example, Figure B.1 shows the initial representation graph of the document "URGENT Denmark to withdraw all troops from Iraq in August. Denmark will withdraw all of its 460 troops stationed in Iraq in August, Danish Prime Minister Anders Fogh Rasmussen announced on Wednesday.". Black nodes are regular nodes, green nodes correspond to entities, blue nodes correspond to events and red nodes correspond to temporal expressions.

Figure B.2 shows the resulting graph after the collapsing and enrichment. Note that the coreference relations no longer exists, and some nodes are grouped. The result is a more compact representation.

Finally, Listing B.1 shows the pseudo-JSON file. The file is structured in two parts, nodes and edges. A node is composed by an identifier and a set of properties, and an edge is composed by a pair of identifiers of nodes and a label.

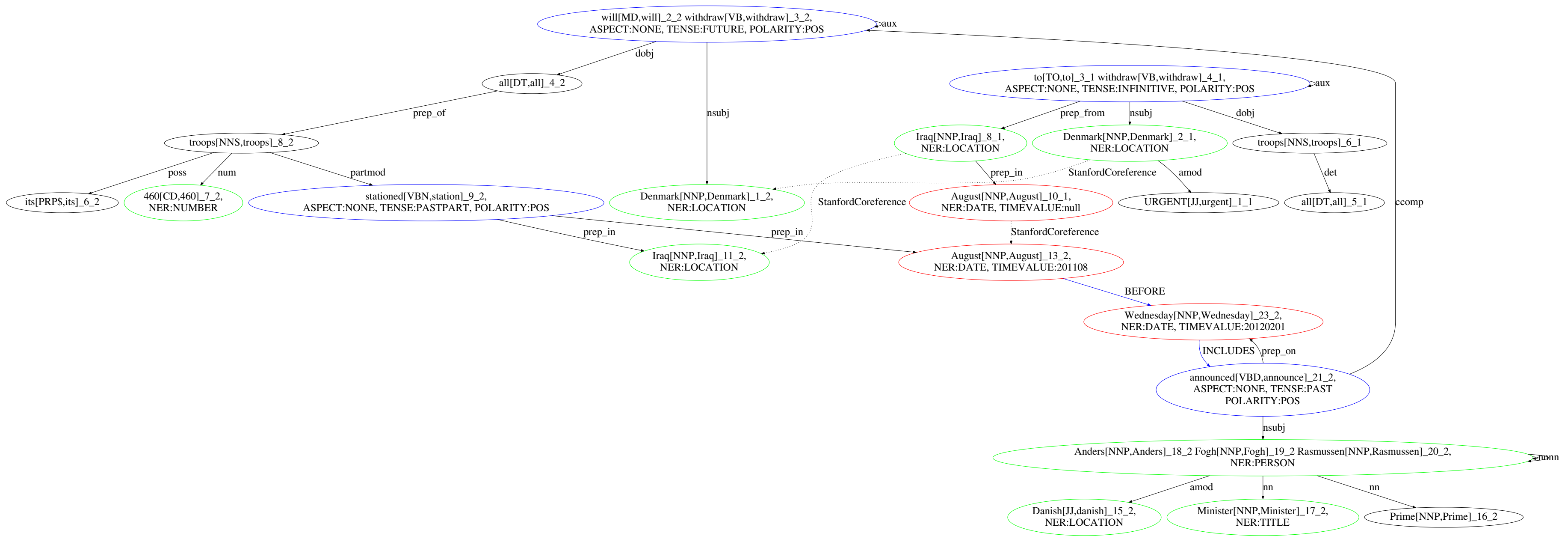

will[MD,will]_2_2 withdraw[VB,withdraw]_3_2, ASPECT:NONE, TENSE:FUTURE, POLARITY:POS
aux
dobj
all[DT,all]_4_2
nsubj
to[TO,to]_3_1 withdraw[VB,withdraw]_4_1, ASPECT:NONE, TENSE:INFINITIVE, POLARITY:POS
aux
prep_of
prep_from
nsubj
dobj
troops[NNS,troops]_8_2
Iraq[NNP,Iraq]_8_1, NER:LOCATION
Denmark[NNP,Denmark]_2_1, NER:LOCATION
troops[NNS,troops]_6_1
poss
num
partmod
prep_in
StanfordCoreference
amod
det
its[PRP$,its]_6_2
460[CD,460]_7_2, NER:NUMBER
stationed[VBN,station]_9_2, ASPECT:NONE, TENSE:PASTPART, POLARITY:POS
Denmark[NNP,Denmark]_1_2, NER:LOCATION
StanfordCoreference
August[NNP,August]_10_1, NER:DATE, TIMEVALUE:null
URGENT[JJ,urgent]_1_1
all[DT,all]_5_1
ccomp
prep_in
prep_in
StanfordCoreference
Iraq[NNP,Iraq]_11_2, NER:LOCATION
August[NNP,August]_13_2, NER:DATE, TIMEVALUE:201108
BEFORE
Wednesday[NNP,Wednesday]_23_2, NER:DATE, TIMEVALUE:20120201
INCLUDES
prep_on
announced[VBD,announce]_21_2, ASPECT:NONE, TENSE:PAST POLARITY:POS
nsubj
Anders[NNP,Anders]_18_2 Fogh[NNP,Fogh]_19_2 Rasmussen[NNP,Rasmussen]_20_2, NER:PERSON
nnnn
amod
nn
nn
Danish[JJ,danish]_15_2, NER:LOCATION
Minister[NNP,Minister]_17_2, NER:TITLE
Prime[NNP,Prime]_16_2

Figure B.1: Initial representation of the document "URGENT Denmark to withdraw all troops from Iraq in August. Denmark will withdraw all of its 460 troops stationed in Iraq in August, Danish Prime Minister Anders Fogh Rasmussen announced on Wednesday." Output formatted as a dot file. Printed with Graphviz.

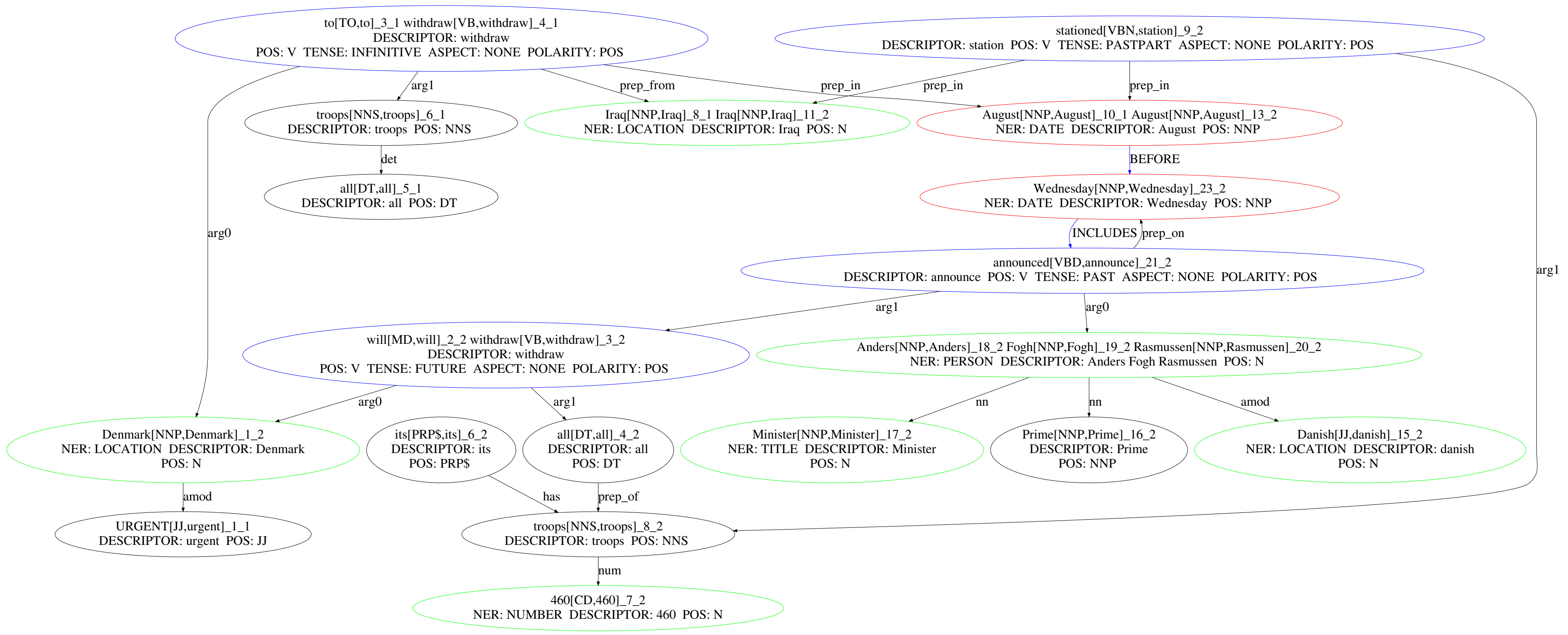

to[TO,to]_3_1 withdraw[VB,withdraw]_4_1
DESCRIPTOR: withdraw
POS: V TENSE: INFINITIVE ASPECT: NONE POLARITY: POS
stationed[VBN,station]_9_2
DESCRIPTOR: station POS: V TENSE: PASTPART ASPECT: NONE POLARITY: POS
arg1
prep_from
prep_in
prep_in
prep_in
troops[NNS,troops]_6_1
DESCRIPTOR: troops POS: NNS
Iraq[NNP,Iraq]_8_1 Iraq[NNP,Iraq]_11_2
NER: LOCATION DESCRIPTOR: Iraq POS: N
August[NNP,August]_10_1 August[NNP,August]_13_2
NER: DATE DESCRIPTOR: August POS: NNP
det
BEFORE
all[DT,all]_5_1
DESCRIPTOR: all POS: DT
Wednesday[NNP,Wednesday]_23_2
NER: DATE DESCRIPTOR: Wednesday POS: NNP
arg0
INCLUDES
prep_on
announced[VBD,announce]_21_2
DESCRIPTOR: announce POS: V TENSE: PAST ASPECT: NONE POLARITY: POS
arg1
arg1
arg0
will[MD,will]_2_2 withdraw[VB,withdraw]_3_2
DESCRIPTOR: withdraw
POS: V TENSE: FUTURE ASPECT: NONE POLARITY: POS
Anders[NNP,Anders]_18_2 Fogh[NNP,Fogh]_19_2 Rasmussen[NNP,Rasmussen]_20_2
NER: PERSON DESCRIPTOR: Anders Fogh Rasmussen POS: N
arg0
arg1
nn
nn
amod
Denmark[NNP,Denmark]_1_2
NER: LOCATION DESCRIPTOR: Denmark
POS: N
its[PRP$,its]_6_2
DESCRIPTOR: its
POS: PRP$
all[DT,all]_4_2
DESCRIPTOR: all
POS: DT
Minister[NNP,Minister]_17_2
NER: TITLE DESCRIPTOR: Minister
POS: N
Prime[NNP,Prime]_16_2
DESCRIPTOR: Prime
POS: NNP
Danish[JJ,danish]_15_2
NER: LOCATION DESCRIPTOR: danish
POS: N
amod
has
prep_of
URGENT[JJ,urgent]_1_1
DESCRIPTOR: urgent POS: JJ
troops[NNS,troops]_8_2
DESCRIPTOR: troops POS: NNS
num
460[CD,460]_7_2
NER: NUMBER DESCRIPTOR: 460 POS: N

Figure B.2: Enriched representation of the document "URGENT Denmark to withdraw all troops from Iraq in August. Denmark will withdraw all of its 460 troops stationed in Iraq in August, Danish Prime Minister Anders Fogh Rasmussen announced on Wednesday." Output formatted as a dot file. Printed with Graphviz.

```
// Nodes
34 {"DESCRIPTOR": "Denmark", "NER": "LOCATION", "POS": "N"
   }
35 {"DESCRIPTOR": "Iraq", "NER": "LOCATION", "POS": "N"}
36 {"DESCRIPTOR": "August", "NER": "DATE", "POS": "NNP"}
38 {"DESCRIPTOR": "460", "NER": "NUMBER", "POS": "N"}
41 {"DESCRIPTOR": "danish", "NER": "LOCATION", "POS": "N"}
42 {"DESCRIPTOR": "Minister", "NER": "TITLE", "POS": "N"}
43 {"DESCRIPTOR": "Anders Fogh Rasmussen", "NER": "PERSON"
   , "POS": "N"}
44 {"DESCRIPTOR": "Wednesday", "NER": "DATE", "POS": "NNP"
   }
45 {"DESCRIPTOR": "urgent", "POS": "JJ"}
46 {"DESCRIPTOR": "withdraw", "POS": "V", "ASPECT": "NONE"
   , "TENSE": "INFINITIVE", "POLARITY": "POS"}
48 {"DESCRIPTOR": "all", "POS": "DT"}
49 {"DESCRIPTOR": "troops", "POS": "NNS"}
50 {"DESCRIPTOR": "from", "POS": "IN"}
51 {"DESCRIPTOR": "in", "POS": "IN"}
52 {"DESCRIPTOR": ".", "POS": "."}
53 {"DESCRIPTOR": "withdraw", "POS": "V", "ASPECT": "NONE"
   , "TENSE": "FUTURE", "POLARITY": "POS"}
55 {"DESCRIPTOR": "all", "POS": "DT"}
56 {"DESCRIPTOR": "of", "POS": "IN"}
57 {"DESCRIPTOR": "its", "POS": "PRP$"}
58 {"DESCRIPTOR": "troops", "POS": "NNS"}
59 {"DESCRIPTOR": "station", "POS": "V", "ASPECT": "NONE",
    "TENSE": "PASTPART", "POLARITY": "POS"}
60 {"DESCRIPTOR": "in", "POS": "IN"}
61 {"DESCRIPTOR": "in", "POS": "IN"}
62 {"DESCRIPTOR": ",", "POS": ","}
63 {"DESCRIPTOR": "Prime", "POS": "NNP"}
64 {"DESCRIPTOR": "announce", "POS": "V", "ASPECT": "NONE"
   , "TENSE": "PAST", "POLARITY": "POS"}
65 {"DESCRIPTOR": "on", "POS": "IN"}
66 {"DESCRIPTOR": ".", "POS": "."}

// Edges
46 49 {"type":"arg1"}
35 35 {"type":"coreference"}
36 36 {"type":"coreference"}
64 44 {"type":"prep_on"}
```

```
34 45 {"type":"amod"}
59 58 {"type":"arg1"}
34 34 {"type":"coreference"}
53 55 {"type":"arg1"}
46 34 {"type":"arg0"}
57 58 {"type":"has"}
43 41 {"type":"amod"}
59 36 {"type":"prep_in"}
49 48 {"type":"det"}
36 44 {"type":"BEFORE"}
64 53 {"type":"arg1"}
53 34 {"type":"arg0"}
43 42 {"type":"nn"}
44 64 {"type":"INCLUDES"}
43 63 {"type":"nn"}
46 35 {"type":"prep_from"}
59 35 {"type":"prep_in"}
46 36 {"type":"prep_in"}
55 58 {"type":"prep_of"}
64 43 {"type":"arg0"}
58 38 {"type":"num"}
```

Listing B.1: Enriched representation of the document "URGENT Denmark to withdraw all troops from Iraq in August. Denmark will withdraw all of its 460 troops stationed in Iraq in August, Danish Prime Minister Anders Fogh Rasmussen announced on Wednesday." Output formatted as a pseudo-JSON file.

C

Proposition Store Examples

In Chapter 4 we have presented our methodology to acquire propositional knowledge. This appendix is devoted to show examples of the propositions acquired.

Consider again the example 1.1 presented in the introduction: *The city council refused the demonstrators a permit because they feared violence.* We can search for the propositions that involve the terms *councils* and *demonstrators* as arguments. Table C.1 shows the relevant propositions obtained. Although it is not trivial, it may be possible to infer that the most plausible interpretation is *city councils fear violence*, or at least, that it is plausible that *demonstrators advocate violence.*

NVN					
Frequency	relation	arg0	arg0 POS	arg1	arg1 POS
279	adopt	council	NN	resolution	NN
272	hold	council	NN	meeting	NN
184	issue	council	NN	statement	NN
184	have	council	NN	member	NNS
168	have	council	NN	power	NN
767	chant	demonstrator	NNS	slogan	NNS
694	throw	demonstrator	NNS	stone	NNS
483	wave	demonstrator	NNS	flag	NNS
476	shout	demonstrator	NNS	slogan	NNS
372	carry	demonstrator	NNS	banner	NNS

Table C.1: Most frequent propositions considering *council* and *demonstrator* as arg0.

In addition, we can also search for the most popular arguments that fear violence, as in Table C.2. Again, arguments such as *authority* or *government* are closer to *city councils* than to *demonstrators.*

NVN					
Frequency	relation	arg0	arg0 POS	arg1	arg1 POS
24	fear	police	NNS	violence	NN
22	fear	observer	NNS	violence	NN
20	fear	official	NNS	violence	NN
16	fear	authority	NNS	violence	NN
10	fear	government	NN	violence	NN

Table C.2: Most frequent propositions considering the relation *fear* and *violence* as arg1.

Next, we present the most frequent propositions considering the three kinds of propositions: predicate-argument propositions, semantic class propositions and typed propositions.

The following tables show some of the most frequent propositions extracted considering patterns NV (Table C.3), NVN (Table C.4), NVPN (Table C.5), VNPN (Table C.6), NVNPN (Table C.7), NPrepN (Table C.8) and NhasN (Table C.9). Propositions are extracted from the Gigaword Corpus, and reflect the journalistic language and conventions used in the documents, such as the prevalence of terms associated with news topics such as business and politics, i.e. *company*, *index*, *official* or *troops*.

NV			
Frequency	relation	arg0	arg0 POS
855169	say	official	NNS
351247	say	spokesman	NN
285682	say	report	NN
269774	say	statement	NN
224334	say	source	NNS

Table C.3: Most frequent propositions extracted considering the NV pattern.

Table C.10 shows the most frequent entities, plus the most frequent entity-semantic class pairs. Unsurprisingly, the most frequent entities are numbers and ordinals, with the notable exception of the acronym *U.S.* as a location. On the contrary, when considering association with semantic classes, locations and organizations are the most frequent entities.

In the following tables we show some of the most frequent typed propositions extracted considering patterns CV (Table C.11), CVC (Table C.12), CVPC (Table C.13), VCPC (Table C.14), CVCPC (Table C.15), CPrepC (Table C.16) and ChasC (Table C.17).

NVN					
Frequency	relation	arg0	arg0 POS	arg1	arg1 POS
21153	rise	index	NN	point	NNS
16928	fall	index	NN	point	NNS
16745	include	city	NNS	municipality	NNS
14029	gain	index	NN	point	NNS
13379	claim	group	NN	responsibility	NN

Table C.4: Most frequent propositions extracted considering the NVN pattern.

NVPN					
Frequency	relation	arg0	arg0 POS	arg2	arg2 POS
20274	close::at	index	NN	point	NNS
18603	contribute::to	writer	NNS	report	NN
18368	speak::on	official	NN	condition	NN
12826	say::in	company	NN	statement	NN
12334	say::on	official	NN	condition	NN

Table C.5: Most frequent propositions extracted considering the NVPN pattern.

VNPN					
Frequency	relation	arg1	arg1 POS	arg2	arg2 POS
12393	tell::after	reporter	NNS	meeting	NN
9956	claim::for	responsibility	NN	attack	NN
9680	speak::on	official	NN	condition	NN
8655	measure::on	earthquake	NN	scale	NN
7198	send::to	troops	NNS	Iraq	NNP

Table C.6: Most frequent propositions extracted considering the VNPN pattern.

NVNPN							
Frequency	relation	arg0	arg0 POS	arg1	arg1 POS	arg2	arg2 POS
6002	change::on	total	NN	hand	NNS	turnover	NN
5995	get::by	non-subscriber	NNS	information	NN	call	VBG
4316	make::by	non-client	NNS	purchase	NNS	call	VBG
3347	include::from	report	NN	feature	NNS	Boston Globe	NNP
2841	decide::in	bank	NNS	rate	NNS	yuan	NN

Table C.7: Most frequent propositions extracted considering the NVNPN pattern.

NPrepN					
Frequency	relation	arg0	arg0 POS	arg1	arg1 POS
157440	of	condition	NN	anonymity	NN
68186	of	weapon	NNS	destruction	NN
67163	of	thousand	NNS	people	NNS
61007	of	number	NN	people	NNS
60471	in	years	NNS	prison	NN

Table C.8: Most frequent propositions extracted considering the NPrepN pattern.

NHasN					
Frequency	relation	arg0	arg0 POS	arg1	arg1 POS
380789	has	U.S.	NNP	dollar	NNS
102760	has	U.S.	NNP	official	NNS
80972	has	Bush	NNP	administration	NN
72046	has	U.S.	NNP	troops	NNS
55250	has	U.S.	NNP	government	NN

Table C.9: Most frequent propositions extracted considering the NHasN pattern.

Semantic Class Propositions			
Frequency	entity	entity type	class
4711056	one	NUMBER	-
3296823	U.S.	LOCATION	-
2605854	two	NUMBER	-
2575269	first	ORDINAL	-
1865796	Tuesday	DATE	-
32012	Yasser Arafat	PERSON	leader
9089	Abuja	LOCATION	capital
9035	Kabul	LOCATION	capital
8614	Mahmud Abbas	PERSON	president
7826	Hamas	ORGANIZATION	group

Table C.10: Most popular entities and most popular entities with an associated semantic class.

CV			
Frequency	relation	arg0 class	arg0 NER
1017530	say	spokesman	C.PERSON
661099	say	director	C.PERSON
555246	say	president	C.PERSON
309868	say	chairman	C.PERSON
272474	say	head	C.PERSON

Table C.11: Most frequent typed propositions extracted considering the CV pattern.

CVC					
Frequency	relation	arg0 class	arg0 NER	arg1 class	arg1 NER
35090	tell	spokesman	C.PERSON	reporter	NNS
31710	tell	spokesman	C.PERSON	ORGANIZATION	NNP
15342	tell	spokesman	C.PERSON	ORGANIZATION	NN
9348	tell	chief	C.PERSON	ORGANIZATION	NNP
9182	tell	leader	C.PERSON	reporter	NNS

Table C.12: Most frequent typed propositions extracted considering the CVC pattern.

CVPC					
Frequency	relation	arg0 class	arg0 NER	arg1 class	arg1 NER
14911	contribute::to	writer	C.PERSON	report	NN
13943	say::in	spokesman	C.PERSON	LOCATION	NNP
5269	tell::in	spokesman	C.PERSON	LOCATION	NNP
3875	contribute::to	correspondent	C.PERSON	report	NN
3747	hold::with	PERSON	NNP	counterpart	C.PERSON

Table C.13: Most frequent typed propositions extracted considering the CVPC pattern.

VCPC					
Frequency	relation	arg1 class	arg1 NER	arg2 class	arg2 NER
3099	tell::in	reporter	NNS	capital	C.LOCATION
2634	base::in	company	C.ORGANIZATION	LOCATION	NNP
1865	base::in	group	C.ORGANIZATION	LOCATION	NNP
1201	tell::in	conference	NN	capital	C.LOCATION
1194	base::in	firm	C.ORGANIZATION	LOCATION	NNP

Table C.14: Most frequent typed propositions extracted considering the VCPC pattern.

CVCPC							
Frequency	relation	arg0 class	arg0 NER	arg1 class	arg1 NER	arg2 class	arg2 NER
2994	tell::in	spokesman	C.PERSON	reporter	NNS	LOCATION	NNP
1451	tell::by	spokesman	C.PERSON	ORGANIZATION	NN	telephone	NN
1440	tell::in	PERSON	NNP	reporter	NNS	capital	C.LOCATION
872	tell::in	leader	C.PERSON	reporter	NNS	LOCATION	NNP
791	tell::in	chief	C.PERSON	reporter	NNS	LOCATION	NNP

Table C.15: Most frequent typed propositions extracted considering the CVCPC pattern.

CPrepC					
Frequency	relation	arg0 class	arg0 NER	arg1 class	arg1 NER
16442	on	contact	C.PERSON	NUMBER	NE
2771	on	contact	C.ORGANIZATION	NUMBER	NE
1846	at	contact	C.PERSON	ORGANIZATION	NE
1666	on	call	C.PERSON	NUMBER	NE
771	on	telephone	C.PERSON	NUMBER	NE

Table C.16: Most frequent typed propositions extracted considering the CPrepC pattern.

CHasC					
Frequency	relation	arg0 class	arg0 NER	arg1 class	arg1 NER
70186	has	ORGANIZATION	NNP	director	C.PERSON
66569	has	ORGANIZATION	NNP	president	C.PERSON
50323	has	ORGANIZATION	NNP	chairman	C.PERSON
32856	has	LOCATION	NNP	capital	C.LOCATION
28930	has	ORGANIZATION	NNP	head	C.PERSON

Table C.17: Most frequent typed propositions extracted considering the CHasC pattern.

D

List of Acronyms

List of acronyms used in this thesis:

- **ACE** Automatic Content Extraction
- **AMR** Abstract Meaning Representation
- **CDT** Conceptual Dependency Theory
- **DRT** Discourse Representation Theory
- **FOL** First Order Logic
- **GPS** Grounded Proposition Store
- **HTTP** HyperText Transfer Protocol
- **IE** Information Extraction
- **KB** Knowledge Base
- **KBP** Knowledge Base Population
- **KBS** Knowledge Base System
- **LD** Linked Data
- **MTT** Meaning-Text Theory
- **MWG** Maximum Weighted Graph
- **NED** Named Entity Disambiguation
- **NER** Named Entity Recognition
- **NLP** Natural Language Processing
- **NLU** Natural Language Understanding
- **OIE** Open Information Extraction
- **OSPC** One Sense Per Collocation

- **OSPD** One Sense Per Discourse
- **POS** Part of Speech
- **QA** Question Answering
- **RDF** Resource Description Framework
- **RSF** Regular Slot Filling
- **SRL** Semantic Role Labeling
- **TAC** Text Analysis Conference
- **TSF** Temporal Slot Filling
- **URI** Uniform Resource Identifiers