Methods for Open Information Extraction and Sense Disambiguation on Natural Language Text

Luciano Del Corro

Dissertation zur Erlangung des Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

Saarbrücken 2015

Dean	Prof. Dr. Markus Bläser
Colloquim	11.01.2016
	Saarbrücken

Examination Board

Supervisor and Reviewer	Prof. Dr. Rainer Gemulla
Reviewer	Prof. Dr. Gerhard Weikum
Reviewer	Prof. Dr. Simone Paolo Ponzetto
Chairman	Prof. Dr. Reinhard Wilhelm
Research Assistant	Dr. Jannik Strötgen

Abstract

Natural language text has been the main and most comprehensive way of expressing and storing knowledge. A long standing goal in computer science is to develop systems that automatically understand textual data, making this knowledge accessible to computers and humans alike. We conceive automatic text understanding as a bottom-up approach, in which a series of interleaved tasks build upon each other. Each task achieves more understanding over the text than the previous one. In this regard, we present three methods that aim to contribute to the primary stages of this setting.

Our first contribution, ClausIE, is an open information extraction method intended to recognize textual expressions of potential facts in text (e.g. "Dante wrote the Divine Comedy") and represent them with an amenable structure for computers [("Dante", "wrote", "the Divine Comedy")]. Unlike previous approaches, ClausIE separates the recognition of the information from its representation, a process that understands the former as universal (i.e., domain-independent) and the later as application-dependent. ClausIE is a principled method that relies on properties of the English language and thereby avoids the use of manually or automatically generated training data.

Once the information in text has been correctly identified, probably the most important element in a structured fact is the relation which links its arguments, a relation whose main component is usually a verbal phrase. Our second contribution, Werdy, is a word entry recognition and disambiguation method. It aims to recognize words or multi-word expressions (e.g., "Divine Comedy" is a multi-word expression) in a fact and disambiguate verbs (e.g., what does "write" mean?). Werdy is also an unsupervised approach, mainly relying on the syntactic and semantic relation established between a verb sense and its arguments.

The other key components in a structured fact are the named entities (e.g., "Dante") that often appear in the arguments. FINET, our last contribution, is a named entity typing method. It aims to understand the types or classes of those names entities (e.g., "Dante" refers to a *writer*). FINET is focused on typing named entities in short inputs (like facts). Unlike previous systems, it is designed to find the types that match the entity mention context (e.g., the fact in which it appears). It uses the most comprehensive type system of any entity typing method to date with more than 16k classes for persons, organizations and locations.

These contributions are intended to constitute constructive building blocks for deeper understanding tasks in a bottom-up automatic text understanding setting.

Kurzfassung

Das Schreiben von Texten ist die wichtigste und reichhaltigste Art und Weise, Wissen auszudrücken und zu speichern. Schon lange verfolgt die Informatik das Ziel, Systeme zu entwickeln, die Texte automatisch verstehen, um dieses Wissen sowohl Maschinen als auch Menschen zugänglich zu machen. In dieser Arbeit verstehen wir das Automatische Textverstehen als bottom-up Aufgabe, in der eine Reihe ineinandergreifender Bausteine aufeinander aufbauen. Jeder Baustein erlangt dabei ein tieferes Textverständnis als der vorhergehende. In diesem Sinne präsentieren wir drei Methoden, die alle zu den fundamentalen Stufen dieses Prozesses beizutragen.

Unser erster Beitrag, ClausIE, ist eine Methode der Offenen Informationsextraktion, die textuelle Ausdrücke von Faktekandidaten (z.B. “Dante schrieb die Göttliche Kommödie”) erkennt, und diese in einer maschinenlesbaren Struktur repräsentiert [(“Dante”, “schrieb”, “die Göttliche Kommödie”)]. Im Gegensatz zu vorherigen Ansätzen trennt ClausIE die Erkennung der faktischen Information von der Repräsentation, in einem Prozess der ersters als universell (d.h. domänenunabhängig), letzteres als streng anwendungsabhängig versteht. ClausIE löst diese Aufgabe in einer grundsätzlichen, auf den Prinzipien der englischen Sprache aufbauenden Weise und vermeidet damit den Gebrauch manueller oder automatisch generierter Trainingsdaten.

Wurde diese Art der Information korrekt identifiziert, ist das wahrscheinlich wichtigste Element eines strukturierten Fakts die Relation, welche die verschiedenen Argumente miteinander verbindet. Hauptbestandteil einer solchen Relation ist üblicherweise eine Verbalphrase. Unser zweiter Beitrag, Werdy, ist eine Worteintrag-Erkennungs und -Disambiguierungsmethode. Es erkennt Wörter oder Mehrwortausdrücke (z.B. ist die “Göttliche Kommdödie” ein Mehrwortausdruck) in einem Fakt und disambiguiert Verben (z.B. was “schreiben” bedeutet). Werdy ist auch ein nichtüberwachtes Verfahren, das hauptsächlich auf der semantischen Beziehung zwischen einer Verbbedeutung und dessen Argumenten beruht.

Die anderen Schlüsselkomponenten eines strukturierten Fakts sind Eigennamen (z.B. “Dante”), die häufig als Argument auftreten. FINET, unser letzer Beitrag, ist eine Methode zur Typisierung von Eigennamen. Sie versteht die Typen oder Klassen solcher Eigennamen (z.B. ist “Dante” ein “Schriftsteller”). FINETs Fokus ist die Typisierung von Eigennamen in kurzen Eingaben, beispielsweise Fakten. Im Gegensatz zu vorherigen Systemen ist es so konzipiert, dass es Typen findet, die dem Kontext der Eigennamen entspricht (z.B. dem Fakt in dem er

auftritt). FINET verwendet mit mehr als 16.000 Typen für Personen, Organisationen und Orten das reichhaltigste Typsystem aller bisherigen Typisierungsmethoden.

Alle Beiträge stellen Bausteine für das tiefere Verständnis in einem bottom-up Verfahren zum automatischen Textverstehen dar.

To Leticia, the love of my life, and Dante, the product of it.

Acknowledgements

First of all I would like to thank my supervisor Rainer Gemulla for his invaluable trust, guidance and support. I enjoyed working with him very much and I am especially grateful for the freedom he gave me during this time and his constant willingness to teach. I would also like to thank Gerhard Weikum for his unbounded support and vital insight. The D5 group at the Max Planck for Informatics provided me with an excellent working environment especially affable and cooperative. I am thankful to my co-authors Abdullah Abujabal, Fabio Petroni and Kaustubh Beedkar, and to Christina Teflioudi, Faraz Makari, Mohamed Yahya and Alejandro Pironti who in one way or the other contributed to my research. Virgilio Tedín Uriburu, Daniel Heymann and Emiliano Chamorro provided me invaluable inspiration and selfless support in my career. I would like to thank the Ambinauts Johannes Hoffart (who also translated the abstract), Dragan Milchevski and Daniel Bär for sharing with me the next adventure. I am grateful to my friends and family, especially to Leticia, without her everything I have achieved would have been truly impossible. Last but not least, I would like to thank all those idealists that encourage me to progress towards an unachievable and always evolving ideal. At least I try.

Contents

Nomenclature

DER Derivationally related form

DP Dependency parse

KB Knowledge-base

NED Named entity disambiguation

NER Named entity recognition

NET Named entity typing

NLP Natural language processing

OIE Open information extraction

POS Part-of-speech

VOS repository Verb-object-sense repository

WERD Word entry recognition and disambiguation

WSD Word sense disambiguation

Chapter 1

Introduction

1.1 Automatic Text Understanding: Goals and Challenges

The great majority of the knowledge that mankind has produced and still produces is available only in the form of natural language text, including books, news articles, scientific papers, and web pages. In much of human history, the most effective and comprehensive way of storing knowledge has been plain text. Our work contributes with methods to represent and understand this knowledge such that it is amenable to automated processing by computers.

Automatic text understanding is not a trivial task. Natural language is written for humans; it can be noisy, ambiguous, opinionated, or difficult to interpret without context or the appropriate background knowledge. Machine Reading (Etzioni et al., 2006) defines "understanding text" as the formation of a coherent set of beliefs based on a textual corpus and a background theory. The success of this paradigm hinges on the capacity to construct a semantic representation of the knowledge in text that can be understood and reasoned about by computers. It requires to identify this "set of beliefs" in its natural language expression, an automatic way to unveil its meaning, and a representation formalism suited for computer processing. These requirements involve the combination of techniques from the machine learning, logic, linguistics, and data management communities, among others. An ideal – or perhaps idealized – system would be sufficiently powerful to capture the entire set of information represented in a given text collection, regardless of its domain (e.g., biology, history, economics).

Conceptually, the goal of automatic text understanding is to develop a system that *replicates human text understanding capabilities*; a system able to generate a computer readable semantic representation of the information embedded in natural language text. According to Woods (1975), this representation (logical adequacy) should "precisely, formally and unambiguously" represent any particular interpretation of the text.

An automatic text understanding system must then overcome multiple challenges. Regardless of its representation formalism, the system must be common to every domain, powerful enough to unveil every possible meaning of the text, capture any piece of information inside it, and be able to infer the non-explicitly stated knowledge. Even more, it must be open in the sense that it must not depend on a bounded set of words, entities, relations or any other element but must able to learn new concepts as they appear. This last characteristic implies that automatic text understanding cannot be exclusively a supervised task as it must not solely rely on existing knowledge. Additionally, as text can be noisy, ambiguous or opinionated, the outcome of the system may well be expressed probabilistically or through certain degree of confidence.

Achieving such an ideal text understanding setting implies the capacity to construct a computer-based ontology which stores in a computer readable format the knowledge processed by the system. This ontology should be able to characterize every object or entity, physical or abstract, and every relation between them. An ontology ultimately constitutes the knowledge available to the machine, which in addition to certain reasoning capabilities, shape applications to serve the most varied purposes. Accordingly, automatic text understanding can also be seen as the challenge to solve the necessary steps to represent natural language written information in a computer based ontology, a task that has been commonly referred as Information Extraction.

In this work, we propose a set of methods to contribute to the initial stages of an automatic text understanding system, a system which, in a near or distant future, can replicate human text understanding capabilities. This, perhaps unachievable, full artificial intelligence does not have to be seen as an all or nothing bet but as a hill climbing approach: any step towards it implies new risks and challenges, but also opens up the set of possible applications that in a major or minor extent have the potential to improve the welfare of humanity.

1.2 Contributions

In a bottom-up approach, an automatic text understanding system can be thought of as a set of interleaved tasks whose aim is to construct a fully fledged ontology or knowledge-base (KB) from natural language text. The KB is conceived as a collection of computer readable facts, statements or beliefs, ideally containing all the information in the original text. Each task in the pipeline achieves more semantic understanding with respect to the previous one. Fig. 1.1 displays a possible example pipeline for this bottom-up approach, in which our contributions are marked in blue.

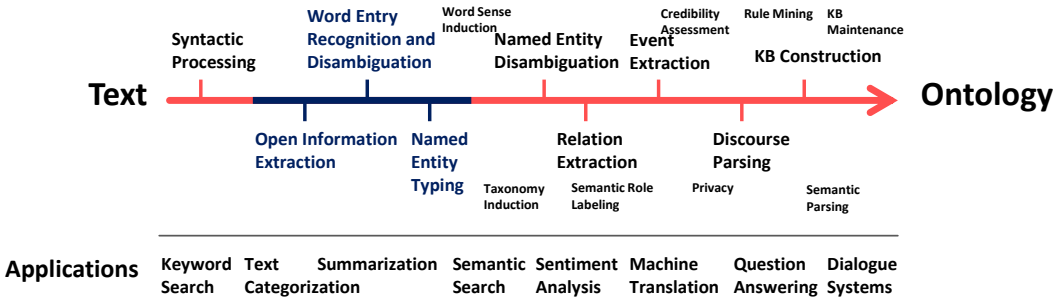


Figure 1.1 Text to ontology: related work

This work proposes three methods that aim to solve the very initial stages on this text understanding (or information extraction) bottom-up approach. They target the most basic processing, aiming to provide strong foundations for semantically deeper tasks. In line with the automatic text understanding setting described in the previous section, our methods are mostly unsupervised, domain independent and avoid filtering information.

The driving idea in this work is that natural language text as the primary form of storing knowledge already entails clear principles to express information. Any method that translates natural language text into a computer readable representation should be able to exploit those principles. Even though linguistic rules may be at times vague or ambiguous, they provide a valuable ground to construct that representation. In this regard, our methods have also the common characteristic of being linguistically-based. Even though they make use a multiple computational techniques, they intensively rely on linguistic knowledge.

As it was stated above, text understanding refers to the formation of a coherent set of beliefs based on a textual corpus and a background theory. At a high level, this first requires to extract or identify the "set of beliefs" (or facts) in text and second to unveil their meaning. Our first method, ClausIE (Del Corro and Gemulla, 2013), is an open information extraction (OIE) (Banko et al., 2007) method which deals with that first phase. It extracts an unbounded set of propositions (i.e., a textual representation of a potential fact[1]) from natural language text with a clear structure in the form of triples or n-ary propositions amenable for computer processing. The second and third methods are focused on the second part: they attempt to discover the elemental semantics of the verbs and named entities, the most important elements in a proposition. The second method, Werdy (Del Corro et al., 2014), a word recognition and

[1]We refer simply as a "fact" to a KB fact, a canonicalized (or disambiguated) representation of a fact [e.g., "Dante passed away in Ravenna" → *diedIn(Dante Alighieri, Ravenna)*], and to a "proposition" as a non-canonicalized representation of a potential fact [e.g., ("Dante, passed away, in Ravenna")]. None of the elements inside a proposition are disambiguated. We mark non-disambiguated pieces of text in quotation marks and disambiguated ones in italics.

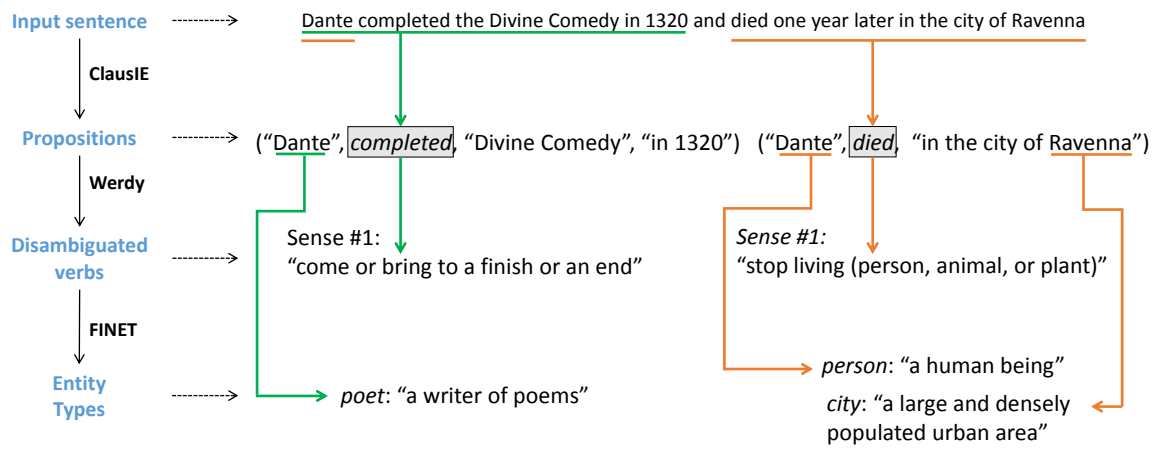


Figure 1.2 Contributions: example

disambiguation (WERD) system, identifies words and multi-word expressions in text and disambiguates verbs. The third one, FINET (Del Corro et al., 2015), is a named entity typing (NET) system which classifies named entities such as persons, organizations or locations with very specific types such as *scientist*, *company*, *city*, etc. Fig. 1.2 presents an overview of our contributions. In the following, we give a more detailed description of each method.

ClausIE. OIE attempts to extract propositions from natural language text. A proposition is a textual (or non-disambiguated) representation of a potential fact. It consist of a relation and a set of arguments. For instance, ("Dante", "passed away in" "Ravenna") is a proposition with a relation "passed away in" linking the arguments "Dante" and "Ravenna"

An OIE system should be scalable, domain independent and not filter out any piece of information. It must be unbounded in the sense that the extracted propositions must not be constrained to any particular set of entities or relations. ClausIE is an OIE system which structures information in text solely based on syntactic properties of the English language. ClausIE overcomes two of the main problems by previous OIE systems.

First, while virtually all of existing OIE methods make use of hand-crafted extraction heuristics or automatically constructed training data to learn extractors (and/or estimate the confidence of propositions) we consider that the detection of the information can be addressed accurately and in a principled and unsupervised way by exploiting well established linguistic principles. Our hypothesis to approach OIE is that language, as the primary representation of knowledge, already provides a systematic way of structuring information, although this structure is often oblivious to computers. Second, unlike previous OIE systems, ClausIE manages to separate the recognition of the information from its materialization. ClausIE is

built on the idea that the recognition of the information should be universal, and therefore domain or application independent, while its materialization should be strictly application dependent.

ClausIE, translates the information in text in "computer language" by exploiting the grammatical structure of sentences. It makes use of the fact that propositions are often expressed in terms of clauses. A clause is essentially a simple sentence that consists of a set of grammatical units, some obligatory (e.g., subject, verb) and some optional (e.g., adverbials). Not all combinations of these constituents appear in the English language. In fact, it is well known that there is exactly seven different clause types containing only obligatory constituents (Quirk et al., 1985). The type of the clause determines the structure of the information and given a clause, we can (in principle) determine its type by exploiting the interaction between the constituents and the verb. Our detection of clauses is based on deep syntactic analysis, a type of analysis which reveals the entire syntactic structure of the sentence (e.g. subjects, direct objects, adverbials, etc).

Consider for example the sentence in Fig. 1.2 "Dante completed the Divine Comedy in 1320 and died one year later in the city of Ravenna". This sentence contains two clauses: "Dante completed the Divine Comedy in 1320" and "Dante died one year later in the city of Ravenna". The first clause is of subject-verb-object (SVO) type with a subject “Dante”, a verb “completed”, an object “the Divine Comedy” and an optional adverbial “in 1320”, which can be eventually omitted. The second clause is of type Subject-Verb (SV) with two optional adverbials: “in the city of Ravenna” and “one year later”. Note that an element is obligatory when it cannot be discarded without changing the meaning of the clause.

After recognizing the clauses and their types ClausIE forms so-called propositions from their grammatical units. In this context, a proposition is ultimately a structured representation of a clause. The generation of propositions can be eventually customized to the underlying application with no effect in the information recognition process. In our example, the propositions can be, for instance, expressed as (“Dante”, “completed”, “the Divine Comedy”, “in 1320”) and (“Dante”, “died”, “in Ravenna”). Propositions are easier to process by computers because they are simple, have a clear representation, and provide information about the structure in terms of subjects, relations, and arguments.

Werdy. The structure of a proposition does not, however, provide a complete picture of the information in a clause. For example, in the proposition (“Dante”, “completed”, “the Divine Comedy”), we would like to understand that "Dante" refers to the famous Italian poet *Dante Alighieri*, that "Divine Comedy" refers to a literary work, or that "completed"

means "bring to an end or finish". The disambiguation of the verb or verbal phrase, the main element linking the constituents of a proposition has received little attention in the literature.

Our Werdy system addresses this gap by automatically disambiguating the verb. Werdy is a method to (i) automatically recognize in natural language text both single words and multi-word phrases that match entries in a lexical KB like WordNet (Fellbaum, 1998), and (ii) disambiguate these words or phrases by identifying their senses in the KB. WordNet is a comprehensive lexical resource for Word sense disambiguation (WSD), covering nouns, verbs, adjectives, adverbs, and many multi-word expressions.

A key challenge for recognizing KB entries in natural language text is that entries often consist of multiple words. In WordNet-3.0 more than 40% of the entries are multi-word. Such entries are challenging to recognize accurately for two main reasons: First, the components of multi-word entries in the KB (such as *fiscal year*) often consist of components that are themselves KB entries (*fiscal* and *year*). Second, multi-word entries (such as *take a breath*) may not appear consecutively in a sentence (“He takes a deep breath.”). Unlike other systems which are bounded to continuous fragments of text of a given maximum length, Werdy addresses this problem in a principled way by (conceptually) matching the dependency syntactic structure of the KB entries to the dependency syntactic structure of the input sentence. This allows the system to discard modifiers that break the continuity of the KB entries in the text. Once Werdy identifies all possible entries in a sentence it passes them to the disambiguation step.

Regarding the disambiguation step, previous work has achieved relative success in the disambiguation of nouns, adjectives and adverbs. However, the disambiguation of verbs and verb-phrases has received less attention. Verb-sense disambiguation is regarded as more difficult task in artificial intelligence because verbs tend to have many different meanings. For instance, the verb "complete" has 5 different meanings in WordNet; other common verbs such as "take" can have more than 40. Werdy is an unsupervised linguistic based system; to determine the correct sense, it exploits the observation that each verb sense occurs in only a limited set of clause types and only with a limited set of arguments. For example, the sense of "complete" that refers to bring to an end requires an SVO clause and an object or semantic argument type that can serve as "piece of work" (e.g., *Divine Comedy*). Given a verb or verbal phrases Werdy prunes its possible senses based on the idea that a verb selects the categories of its arguments both syntactically (c-selection) and semantically (s-selection). By systematically leveraging this knowledge, Werdy is able to determine the sense of each verb with high precision.

FINET. After the verb, the main constituents of a proposition are probably the named entities that may appear in it. In our example proposition (“Dante”, “completed”, “the Divine Comedy”), once we know the meaning of the verb, we may want to understand what "Dante" or "Divine Comedy" refer to. One way of characterizing named entities is through their types or classes. For instance, we could infer that “Dante” is a *poet* or *writer* and “Divine Comedy” is a *poem*, a *literary work*, or a *book*. NET is a key task in automatic text understanding since it allows a new level of semantic understanding.

In this work we describe FINET, a NET system which efficiently types named entity mentions in short inputs —such as propositions, sentences or tweets— with respect to WordNet’s super fine-grained type system. Unlike previous systems FINET aims to extract the most explicit type, the one that best fits the context. For instance, for the input “Obama wrote a book” the best type according to the context for the named entity “Obama” would be *author* or *writer*. However, given the supervised or semi-supervised nature of existing systems the most likely type would be *president*. Supervised or semi-supervised systems cannot detect types that were not present in the training set and have difficulties in mapping types to context when the training data is not adequate.

FINET is different from previous approaches in the sense that it generates explicit candidate types in an unsupervised way. It uses a sequence of multiple extractors, ranging from explicitly mentioned types to implicit types, and subsequently selects the most appropriate one using ideas from WSD. FINET combats the data scarcity and noise problems that plague existing systems for named entity typing: it does not rely on supervision in many of its extractors and it generates training data for type selection directly from WordNet and other resources.

Our system makes use of explicit type extractions whenever possible. FINET consists of four extractors. The first one is pattern based for cases in which the explicit type is mentioned in the sentence (e.g., “President Barack Obama gave a speech”). The second one attempts to detect the type from the type mention (e.g., “New York City”). The third one exploits the verb-argument concordance reasoning over the verb (e.g., “Messi plays soccer”). Finally, if none of the previous steps fire, our corpus-based extractor, leverage a large unlabeled corpus to propagate types from similar named entities occurring in the same context. This last extractor makes use of the distributional hypothesis: entities appearing in the same context tend to be of the same type (e.g., “Barack Obama met Dilma Rousseff in Brasilia”).

FINET supports the most fine-grained type system so far, including types for which no training data is provided. FINET supports the entire WordNet hierarchy with more than 16k types for locations, organizations and persons, the previous most fine grained system supports only frequent 505 types.

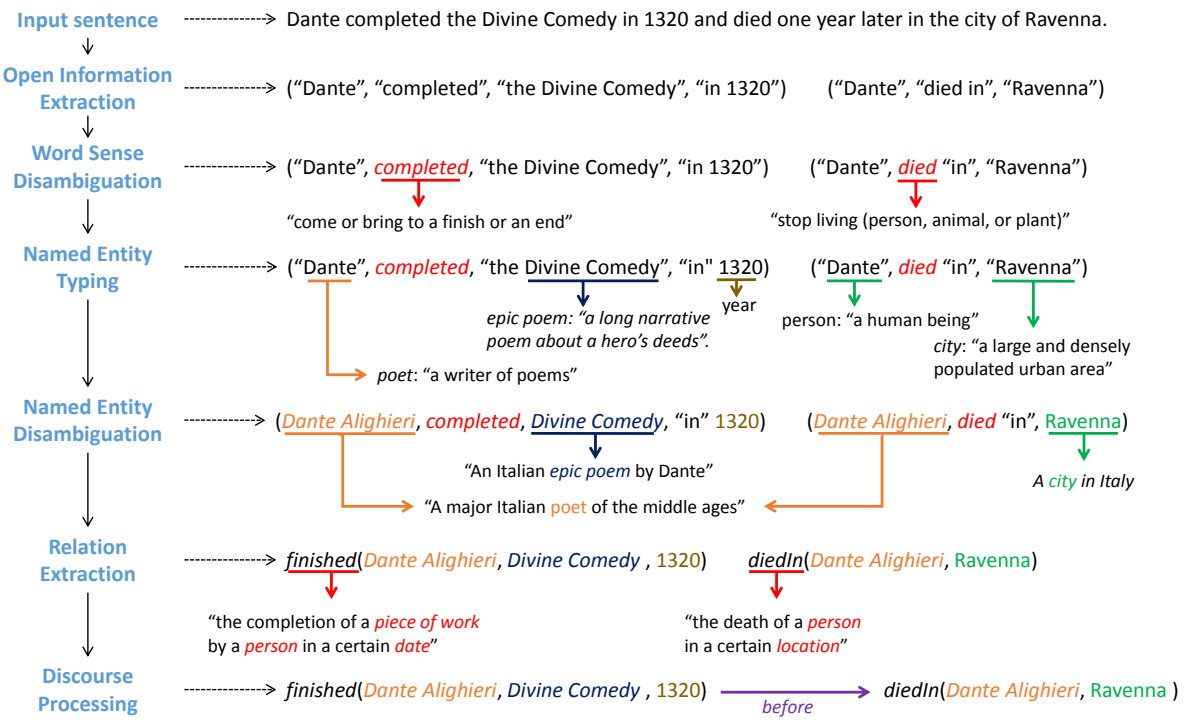


Figure 1.3 Text to ontology: example.

1.3 Applications

In a bottom-up approach to automatic text understanding (as displayed in Fig. 1.1) shallower tasks can be seen as input for deeper ones. In this perspective, going from plain text to a full text understanding setting configures a multiple-layer build up, each of them carrying additional semantic information (or increasing the "understanding degree") with respect to the previous one until a "final" full (computer-based) semantic layer is achieved. In this section, we describe how our methods may serve as input for deeper text understanding tasks. Fig. 1.3 displays an example of this semantic build-up.

Named Entity Disambiguation. NED (Ferragina and Scaiella, 2010; Hoffart et al., 2011; Usbeck et al., 2014; Moro et al., 2014). is the task of linking a named entity mention in text to an entity in a KB. For instance, the goal is to understand that in the clause "Dante completed the Divine Comedy", "Dante" refers to the famous Italian poet *Dante Alighieri* and not to the soccer player *Dante Bonfim Costa Santos*. NED is not a trivial task, named entity mentions can be highly ambiguous with hundreds of named entity candidates. NET systems like FINET can help to prune that candidate space. For instance, if we know in advance that the text fragment "Dante" must refer to a *poet* or a *writer* we can immediately

discard entities like *soccer players*. NED can also benefit from WERD. First, given a named entity repository, the recognition of a named entity in text is equivalent to the recognition of words as presented in Werdy. This is specially important in the case of certain named entities whose mentions may occur in discontinuous pieces of text (e.g., "Antony 'Tony' Montana"). Second, it has already been shown that disambiguating word senses may help NED and vice-versa (Moro et al., 2014). The idea is that some word senses are more related to certain named entities. For instance, *soccer players* like *Thomas Müller* tend to occur more often with the verb *play* which refers to "participate in sports" and *musicians* such as *Wolfang Müller* are more related to the sense of *play* which denotes "play on an instrument".

Relation Extraction. Commonly, relation extraction (Surdeanu and Ciaramita, 2007; Mintz et al., 2009; Nickel et al., 2012; Drumond et al., 2012; Min et al., 2013; Riedel et al., 2013; Petroni et al., 2015) refers to the extraction of facts from natural language text. Relation Extraction, takes as input (or subsumes) all the shallower tasks to its left in Fig. 1.1. Relation extraction requires to understand the form of the fact, a task tackled by ClausIE. It also requires disambiguated named entities and to understand the relation between them. In open relation extraction, where relations are not constrained to any subset (a more appropriate setting for automatic text understanding), it has been shown that if the named entities of the fact are previously disambiguated (Riedel et al., 2013) or if their types are provided as context (Petroni et al., 2015) the performance of the extractor increases significantly. Relation extraction is also close to verb sense disambiguation. In some cases, when the object of the relation corresponds to a grammatical object (as in ("Dante", "completed", "the Divine Comedy")), the relation simply corresponds to the verb sense.

Event Extraction. An event is something that happens in a given point of time and place. Event extraction (Ling and Weld, 2010; Kuzey and Weikum, 2014) is the task of recognizing and classifying events in text. It is a very related task to entity typing, verb-sense disambiguation and relation extraction. Examples of event types are *elections*, *tournaments* or *volcanic eruptions* and the occurrences or instances of these event types are known as named events. Extracting an event requires first to recognize the proposition referring to an event (e.g., ("Dante", "died", "in Ravenna", "in 1320")), an OIE task, and later tag it with the appropriate type (e.g., *death*). Types from events are very related to verb-senses. A verb describes what is happening in a clause, and therefore, the type of the event through an appropriate nominalization (i.e., the transformation of a verb into a noun *die* $\rightarrow$ *death*). In this regard, FINET describes a method to extract types from verbs.

Discourse Parsing. Discourse Parsing (Stede, 2012; Hernault et al., 2010; Feng and Hirst, 2012) is the task of discovering the semantic relation between different text units (clauses, sentences and other groupings). As we show in ClausIE, propositions can be directly extracted from text clauses. Thus, discourse parsing, is a very suitable framework to understand the relations between facts or propositions. Discourse Parsing is a relatively newly explored field which recently has acquired high prominence. It is probably one of the deeper tasks in the text understanding pipeline and one could consider it as the task of discovering relations between facts. For instance, one could explicitly think of a KB not only as a collection of facts, but as a collection of linked facts. Given our example sentences above, we could include in our ontology two connected facts as *completed(Dante Alighieri, Divine Comedy)* $\xrightarrow[before]{}$ *diedIn(Dante Alighieri, Ravenna)*.

Ontology Construction. In principle, a KB is a collection of entities and relations between them. Ontology or KB construction traditionally consists of gathering facts from unstructured or semi-structured sources and store them in a KB. The unstructured source is usually plain text, while semi-structured sources corresponds to tables or data arrangements without a well defined schema (e.g., Wikipedia infoboxes). Most of the well established KBs, extract information from semi-structured sources (Hoffart et al., 2013; Lehmann et al., 2014) or rely on manual efforts (Foundation, 2015; Bollacker et al., 2008). However, there is increasing amount of work in the construction of KBs from plain text (Suchanek et al., 2009; Carlson et al., 2010; Zouaq, 2011; Wu et al., 2012; Dong et al., 2014).

KB construction from text takes as input or subsumes the tasks described to its left in the pipeline above. For instance, it needs to structure the information in text (as OIE would do) and it needs to disambiguate the components of the propositions like the entities and the relation. In principle it basically requires to go from a proposition or clause to the fact that will be eventually stored. Taking advantage of an ontology in a text understanding setting also requires a formalism that allows to reason about the information in the KB; an aim of semantic parsing (Krishnamurthy and Mitchell, 2014; Grefenstette et al., 2014) which has received considerable attention recently. Different methods related to KB construction have been developed to address user privacy (Biega et al., 2014), the credibility of the information (Qazvinian et al., 2011; Mocanu et al., 2014) or the sources (Dong et al., 2015), the extraction of metaphorical (Strzalkowski et al., 2013; Schulder and Hovy, 2014) or common sense knowledge (Tandon et al., 2014), the extension of a given KB (Gupta et al., 2014; West et al., 2014; Carlson et al., 2010), the discovery of non explicitly stated knowledge (Carlson et al., 2010; Galárraga et al., 2013), among others.

KB construction is a titanic effort which needs to cover many important aspects of the automatic text understanding framework. A KB ultimately constitutes the knowledge accessible to the computer and it is reasonable to assume that the more accurate and comprehensive this knowledge, the more complex and useful the applications that can be developed. In this regard, it is important that each task in the pipeline is solved appropriately to guarantee an acceptable output.

1.4 Why Automatic Text Understanding?

The ultimate goal of research should be to improve human life quality in its various aspects. In computer science this has a direct materialization though the range of applications that reach end-users. In the previous section we have described different semantic layers according to the degree of understanding that each task accomplishes. In this section we discuss a few end-user applications that can be derived from the pipeline above.

The set of layers in the process of automatic text understanding are mostly built upon each other in the sense that more semantic layers already carry the information embedded in the shallower ones. Each subsequent layer increases the understanding capabilities over the information, and therefore the knowledge accessible to the computer, but also requires deeper and more complex reasoning, and therefore more computing resources. This makes important to determine what is the optimal level necessary to acquire the required knowledge for the underlying application.

Each level of understanding allows the development of a new range of more complex applications. From basic tasks such as keyword or structured search to more complex ones such as question answering or semantic search that require deeper understanding. The potentiality of the given application depends ultimately on the system capacity to replicate human understanding capabilities.

At the the deeper understanding level we can name applications such as Semantic Search, Question Answering or Dialogue Systems. Semantic Search (Hoffart et al., 2014) refers to the capacity of the search engine to fully abstract itself from the lexical form of the concepts (i.e., it does not search for strings but for concepts or meaning). This allows the engine to better interpret the intentions of the user and handle more general and complex queries. If the search engine first understands in which pieces of text *Dante Alighieri* is mentioned regardless of how he is "mentioned" (e.g, “Dante Alighieri”, “Dante”, “the author of the Divine Comedy”, etc) the user can directly search for the person *Dante Alighieri* and all the documents where he is mentioned will be retrieved no matter how he is "textually" mentioned. Even more, if types of the entities are known the user can directly search for *writers* so

that all documents were *Dante Alighieri* or other *writers* are mentioned will be retrieved regardless if the term "writer" is mentioned in them. A range of semantic search engines have recently appeared mostly as prototypes (e.g., Hoffart et al. (2014)) and even major search engines have become more semantically based recently.

Question Answering (Yahya et al., 2013; Ravichandran and Hovy, 2002) is also a task requiring deep text understanding capabilities. It aims to automatically answer questions posed in natural language, or more generally to provide the user with the piece of information it requested and not a document which may contain it. Given our example in Fig. 1.3 we can ask questions like "Where did Dante died?" and the answer will be *Ravenna*. Again in this case, the knowledge over the entities can be used to answer questions not directly stated in text such as "In which continent did Dante died?". Question Answering is, in principle, a more complex task than semantic search since it additionally requires understanding and structuring the information in a form that can be later mapped to questions and answers. It requires the difficult task to interpret the question in terms of KB queries. At a high level the question must be structured in terms of a proposition, as OIE would do, understand its meaning, and unknown variables in terms of a KB fact. Recently, the IBM system Watson was able to beat the champions of Jeopardy, a popular question answering TV program in the United States. Of course one can think that there is still a long way to go since questions in a TV program tend to be rather predictable; they tend to be structurally simple, direct and limited in the number of topics. However, it still constitutes a landmark achievement showing not only the importance of generating computer manageable knowledge but also that the potentiality to generate text understanding based applications for everyday life is not anymore an unachievable ideal.

Finally, as an ultimate application one can think of a system which engages in full conversational interaction with the user. A system able to not only answer questions but to better interpret user intention, ask the user for feedback in order to fulfill specific requirements or adapt to particular circumstances. The field dealing with such a system has been called dialogue systems (Milward and Beveridge, 2003; Sonntag et al., 2010). A dialogue system should ideally be able to interact with a person as it was an ordinary human being but with faster reasoning capabilities. Some applications of dialogue systems, although still far away from this idealized version, have been introduced in every day life (e.g. Apple Siri, Microsoft Cortana, Google Now), industry (e.g. call-centers) and even the medical domain (Sonntag and Schulz, 2014).

According to Alan Turing an intelligent system is one in which a machine behaves in a indistinguishable way with a man while interacting. The range of applications that a system like that may trigger is enormous and beyond today's imagination, imposing risks

and challenges that will need to be addressed, but also huge benefits to our everyday life. We believe that the possibility to achieve such a goal highly depends on our ability to maximize the automatic text understanding capabilities and therefore the knowledge that the machine is able to handle. We hope that the methods presented here constitute strong foundations that contribute to that goal.

1.5 Publications

This work includes material published in peer-reviewed papers and in the reports of the Max-Planck scientific advisory board and curatorship board. These publications in chronological order are the following:

- Del Corro, L., and Gemulla, R. (2013). ClausIE: Clause-Based Open Information Extraction. In Proceedings of WWW, pages 355-366.

- Del Corro, L., and Gemulla, R. (2013). Clause-Based Open Information Extraction. MPI for Informatics, Max Planck Society, editor, Eleventh Biennial Report : May 2011 – March 2013, pages 570-572.

- Del Corro, L., Gemulla, R., and Weikum, G. (2014). Werdy: Recognition and disambiguation of verbs and verb phrases with syntactic and semantic pruning. In Proceedings of EMNLP, pages 374–385.

- Del Corro, L., and Gemulla, R. (2015). Clause-Based Open Information Extraction. MPI for Informatics, Max Planck Society, editor, Twelfth Biennial Report : April 2013 – March 2015, pages 678-680.

- Del Corro, L., Gemulla, R., and Weikum, G. (2015). Werdy: Recognition and Disambiguation of Verbs and Verb Phrases with Syntactic and Semantic Pruning. MPI for Informatics, Max Planck Society, editor, Twelfth Biennial Report : April 2013 – March 2015, pages 680-682.

- Del Corro, L., Gemulla, R. (2015). Open Information Extraction. MPI for Informatics, Max Planck Society, editor, Report 2015, page 75.

- Del Corro, L., Abujabal, A., Gemulla, R., and Weikum, G. (2015). Finet: Context-aware fine-grained named entity typing. In Proceedings of EMNLP, pages 868–878.

1.6 Outlook

The reminder of this thesis is organized as follows. Chapter 2 describes ClausIE, our OIE system. Chapter 3 presents Werdy a WERD system which recognizes words and multi-word expressions and disambiguates verbs. Chapter 4 introduces FINET our NET system. Each chapter contains a detailed description of the method, a comprehensive discussion over related work and an extensive experimental evaluation. The source code with the implementation of each method and the datasets used in the experimental evaluations are openly available in all cases. Finally, Chapter 5 provides the conclusion of this work and future directions for improvements.

Chapter 2

ClausIE: Clause-Based Open Information Extraction

2.1 Introduction

As described in the previous chapter, open information extraction (OIE) (Banko et al., 2007) is the natural first step to any automatic text understanding approach. It aims to obtain a structured machine-readable representation of the information in text in the form of triples or n-ary propositions. The propositions itself may also be used in end-user applications such as structured search or may serve as input to deeper text understanding tasks.

OIE requires a general method capable of working on text regardless its domain (e.g. biology, history, economics, etc) that captures the entire set of information inside it. It aims to structure large amounts of natural-language text with a clear representation in the form of triples or n-ary propositions. The key goals of OIE are (1) domain independence, (2) unsupervised extraction, and (3) scalability to large amounts of text. OIE methods do not require any background knowledge or manually labeled training data and are not limited to a set of pre-specified relations or entities. In this context, we developed an OIE method called ClausIE (Del Corro and Gemulla, 2013). ClausIE is completely unsupervised, solely based on linguistic principles of the English language.

Consider for example the sentence "A. Einstein, who was born in Ulm, has won the Nobel Prize." OIE systems aim to extract triples ("A. Einstein", "has won", "Nobel Prize") and ("A. Einstein", "was born in", "Ulm") from this sentence, in which no entity resolution or disambiguation of the verbal phrase is performed. We call each extraction a *proposition*. A proposition consists of a *subject* ("A. Einstein"), a relational phrase or simply *relation* ("has won"), and zero, one, or more *arguments* ("the Nobel Prize"). OIE is perhaps simplest

form of text analysis (i.e., we know the structure of the information but not the meaning of it). The extracted propositions can be used directly for applications such as shallow semantic querying (Who has won the Nobel Prize?) or structured search (e.g., retrieve all propositions with "Albert Einstein" as subject), and, as discussed in the previous chapter, it may serve as input for deeper text understanding tasks such as semantic role labeling, relation extraction, KB construction. Consider for example the task of extending a given ontology about persons and their prizes. Entity disambiguation techniques may identify and link both "Albert Einstein" and the "Nobel Prize" in the above sentence, OIE methods establish the connection between these entities [(*Albert Einstein*, "has won", *Nobel Prize*)], and relation extraction techniques try to obtain the fully disambiguated fact (Petroni et al., 2015) [*won(Albert Einstein, Nobel Prize)*].

Virtually all existing OIE methods make use of hand-crafted extraction heuristics or automatically constructed training data to learn extractors (and/or estimate the confidence of propositions). Some approaches—such as TextRunner (Banko et al., 2007), WOE^{pos} (Wu and Weld, 2010), Reverb (Fader et al., 2011), and R2A2 (Etzioni et al., 2011)—focus on efficiency by restricting syntactic analysis to part-of-speech (POS) tagging and chunking. These fast extractors usually obtain high precision for high-confidence propositions, i.e., at low points of recall, but the restriction to shallow syntactic analysis limits maximum recall and/or may lead to a significant drop of precision at higher points of recall. Other approaches—such as Wanderlust (Akbik and Broß, 2009), WOE^{parse} (Wu and Weld, 2010), KrakeN (Akbik and Löser, 2012), OLLIE (Mausam et al., 2012), Gamallo et al. (2012), Bast and Haussmann (2013)—additionally use dependency parsing or parse trees. These extractors are more expensive than the extractors above; they trade efficiency for improved precision and recall. Each of these approaches makes use of various heuristics to obtain propositions from the dependency parses.

Our approach to OIE called ClausIE (for clause-based open information extraction) falls into the second category. ClausIE fundamentally differs from previous approaches in that it separates (i) the *detection* of the information expressed in a sentence from (ii) its *representation* in terms of one or more propositions. The output of this first phase is the detection of the full structure of the clause, which can be used in a second phase to materialize propositions according to the underlying application. The key idea is that the first step should be "universal" (i.e., domain and application independent), whereas the second step will be determined by the requirements of the underlying application or the domain. The first step identifies the information and the second expresses it.

The main reasoning behind this separation is that (i) can be addressed accurately and in a principled way by exploiting properties of the English language. In ClausIE, we establish the

connection between, linguistic clauses, clause types and propositions. We identify the set of "clauses" of each sentence and, for each clause, the corresponding clause type according to the grammatical function of its constituent (e.g., subject-verb-object, SVO). Our detection of clauses is based on the grammatical structure of the sentence; to detect clause types, we additionally use a small set of domain-independent lexica (e.g., of copular verbs). In contrast to many previous approaches, ClausIE does not make use of any training data, whether labeled or automatically constructed, and does not require global post-processing (e.g., to filter out low-precision extractions), i.e., document processing in ClausIE is embarrassingly parallel. These properties allow ClausIE to process both individual sentences as well as large document collections automatically and in a scalable way. Since ClausIE is a principled technique, its accuracy greatly depends on the ability of the syntactic analyzer, used at the background, to correctly detect the grammatical structure of the sentence.

In the second phase, we generate one or more propositions for each clause based on the type of the clause; the generation of propositions can be customized to the underlying application. Once we have identified the type of clause, we can determine the optional and obligatory constituents of the proposition (to be generated) to which the clause refers to. It is, we can identify the essential and optional pieces of information inside each clause and give the representation required by the specific application (e.g. triples, n-ary propositions, with optional arguments, etc) and since each clause expresses a proposition, we can generate at least one proposition per clause. For example, from the clause "Anna passed the exam with ease," we may want to generate one or more of the following propositions: ("Anna", "passed the exam with", "ease"), ("Anna", "passed", "the exam with ease"), ("Anna", "passed", "the exam"), or 4-tuple ("Anna", "passed", "the exam", "with ease"?) where the last argument is marked as optional. The form that the relation can take, can be also potentially customized in the context of the particular application. For example, in the sentence "Messi from Argentina plays in Bacelona" either ("Messi from Argentina", "plays", "in Barcelona") or ("Messi from Argentina", "plays in", "Barcelona") or alternatively ("Messi", "plays in", "Barcelona") are different materialization of the same original information.

In contrast to previous approaches, ClausIE does not make use of any training data, whether labeled or automatically constructed. Moreover, ClausIE, unlike other OIE systems, can (optionally) extract propositions in which the subject or one or more of the arguments does not constitute a noun phrase.

ClausIE also generates extractions from non-verbal relations. For instance, from the sentence "Albert Einstein, the German scientist, won the Nobel Prize", ClausIE can generate the proposition ("Albert Einstein", "is", "German scientist"). The coverage of non verbal

relations is at the moment limited to appositions or participial modifiers but it can be extended by including the appropriate syntactic-based rules over the clause structure.

Compared to existing methods ClausIE achieves higher precision and recall of the extracted propositions. We conducted an experimental study on multiple real-world datasets of varying quality in order to compare ClausIE to alternative approaches. We found that ClausIE obtains significantly more propositions than most previous approaches (3.8–4.6 times more correct propositions) at similar or higher precision.

2.2 The Seven Clauses

A *clause* is a part of a sentence that expresses some coherent piece of information; it consists of one *subject* (S), one *verb* (V), and optionally of an *indirect object* (O), a *direct object* (O), a *complement* (C), and one or more *adverbials* (A). Not all combinations of these constituents appear in the English language. In fact, when clauses are classified according to the grammatical function of their constituents, we obtain only seven different clause types (Quirk et al., 1985).[1] For example, the sentence "AE has won the Nobel Prize" is of type SVO; here "AE" is the subject, "has won" the verb, and "the Nobel Prize" the object. A complete list of all seven clause types is given in the upper part of Tab. 2.1.

Assume for the moment that the input sentence consists of only a single clause. ClausIE is based on the observation that the clause type conveys the minimal unit of coherent information in the clause. Intuitively, this means that if we remove a constituent of a clause that is also part of its type, the resulting clause does not carry semantically meaningful information (or the sense of the verb changes). For example, the sentence "AE remained in Princeton" consists of a subject, a verb, and an adverbial. The clause is of type SVA, i.e., the clause "AE remained" obtained by ignoring the adverbial is incoherent (and indeed semantically meaningless). In contrast, clause "AE died in Princeton"—which also consists of a subject, a verb, and an adverbial—is of type SV. Since here the adverbial does not appear in the clause type, the derived clause "AE died" is coherent. In what follows, we call constituents of a clause that are also part of the clause type *essential* (here "AE" and "died"); all other constituents are called *optional* ("in Princeton"). Note that subjects, verbs, (direct and indirect) objects, and complements are always essential; adverbials, however, may or may not be essential.

Coherence plays an important role in OIE. For example, Reverb (Fader et al., 2011) employs heuristic rules in order to avoid (some) incoherent extractions. ClausIE ultimately aims to generate propositions from the constituents of the clause. Coherency tells us which constituents must be included into a proposition and which may be omitted. One option to

[1] There is also an existential clause (such as this one), which we treat similarly to SV.

ensure coherent extractions is to always construct propositions that include all constituents of a clause. Such an approach addresses coherency, but—as argued by Fader et al. (2011)—may in turn lead to over-specified extractions. Consider, for example, sentence "AE was awarded the NP in Sweden in 1921" and suppose we limit attention to noun-phrase arguments; such an approach is followed by most OIE systems. We can then extract coherent propositions

P_1=("AE", "was awarded", "the NP"),
P_2=("AE", "was awarded the NP in", "Sweden"),
P_3=("AE", "was awarded the NP in", "1921"),
P_4=("AE", "was awarded the NP in Sweden in", "1921").

Here P_4 (and perhaps P_2 and P_3) is over-specified in that phrase "was awarded the Nobel Prize in Sweden in" is probably not a good relational phrase. Since ClausIE detects essential and optional constituents of a clause, we can customize proposition generation as desired; coherency is always guaranteed. One potential customization—which we also used in our experimental study—is to extract all coherent propositions in combination with zero or one optional adverbial. With this approach, we extract P_1, P_2, and P_3, but not P_4 from the sentence above.[2] Heuristic approaches such as Reverb do not allow for such flexibility (in our example, Reverb extracts P_2 only). As a final note, over-specificity can also arise in subjects, objects, and complements; here the dependency parse (DP) can be exploited to address over-specificity in a natural way. For instance in the sentence "The great AE from Germany was awarded the NP in Sweden in 1921" a proposition like P_1 above can be easily generated by ignoring the prepositional phrase and the adjective in the subject which modify "AE". The hyerarchical structure of the DP naturally tells us the relative importance of the information inside each argument (i.e. "AE" is the parent of both the adjective and the prepositional phrase). Currently, ClausIE does not explicitly adress over-specificity of the arguments, a natural direction for future work.

Given a clause, we can (in principle) determine its type. First observe that each occurrence of a verb in an English sentence is of exactly one of the following types: intransitive, (extended) copular, monotransitive, ditransitive, or complex transitive. A verb is *intransitive* if it does not take an object argument, *monotransitive* if it takes a direct object, and *ditransitive* if it takes both a direct and an indirect object. *Copular* verbs link the subject with a complement or predicative, while *extended-copular* verbs express a relation between the subject and an obligatory adverbial (Quirk et al., 1985). As it can be seen in Tab. 2.1, which also gives an example sentence for each verb type, the verb type along with the presence of a direct object, indirect object, or complement *uniquely* identifies the type of a clause. Vice versa,

[2] ClausIE may also be customized to extract n-tuple ("AE", "was awarded", "the NP", "in Sweden"?, "in 1921"?), where "?" indicates optional arguments.

the verb type is uniquely determined by the (type of the) constituents and the type of the clause. We exploit this observation directly in ClausIE, i.e., we exploit information about the clause obtained from the DP, and information about verb types from a small set of domain independent lexica. In many cases, this combined approach allows us to accurately determine the clause type; see Sec. 2.3.

If a sentence contains multiple (potentially nested) clauses, ClausIE considers each clause separately. Consider, for example, sentence "AE was awarded the NP before Schrödinger devised his famous thought experiment". The sentence contains two clauses (one spanning the entire sentence, and one starting at "Schrödinger"); coherent propositions include ("AE", "was awarded", "the NP") and ("Schrödinger", "devised", "his famous thought experiment"). OIE does not aim to capture the "context" of each clause; this simplification allows for effective extraction but may also lead to non-factual extractions (Mausam et al., 2012). For example, the proposition ("the only real valuable thing", "is", "intuition") obtained from the second clause of sentence "AE said the only real valuable thing is intuition" is non-factual. We do not specifically avoid non-factual propositions in ClausIE; see Mausam et al. (2012) for techniques that can detect such propositions.

Table 2.1 Patterns and clause types (based on Quirk et al. (1985)).

	Pattern	Clause type	Example	Derived clauses
			Basic patterns	
S_1:	SV_i	SV	AE died.	(AE, died)
S_2:	SV_eA	SVA	AE remained in Princeton.	(AE, remained, in Princeton)
S_3:	SV_cC	SVC	AE is smart.	(AE, is, smart)
S_4:	$SV_{mt}O$	SVO	AE has won the Nobel Prize.	(AE, has won, the Nobel Prize)
S_5:	$SV_{dt}O_iO$	SVOO	RSAS gave AE the Nobel Prize.	(RSAS, gave, AE, the Nobel Prize)
S_6:	$SV_{ct}OA$	SVOA	The doorman showed AE to his office.	(The doorman, showed, AE, to his office)
S_7:	$SV_{ct}OC$	SVOC	AE declared the meeting open.	(AE, declared, the meeting, open)
			Some extended patterns	
S_8:	SV_iAA	SV	AE died in Princeton in 1955.	(AE, died) (AE, died, in Princeton) (AE, died, in 1955) (AE, died, in Princeton, in 1955)
S_9:	SV_eAA	SVA	AE remained in Princeton until his death.	(AE, remained, in Princeton) (AE, remained, in Princeton, until his death)
S_{10}:	SV_cCA	SVC	AE is a scientist of the 20th century.	(AE, is, a scientist) (AE, is, a scientist, of the 20th century)
S_{11}:	$SV_{mt}OA$	SVO	AE has won the Nobel Prize in 1921.	(AE, has won, the Nobel Prize) (AE, has won, the Nobel Prize, in 1921)
S_{12}:	$ASV_{mt}O$	SVO	In 1921, AE has won the Nobel Prize.	(AE, has won, the Nobel Prize) (AE, has won, the Nobel Prize, in 1921)

S: Subject, V: Verb, C: Complement, O: Direct object, O_i: Indirect object, A: Adverbial, V_i: Intransitive verb, V_c: Copular verb, V_c: Extended-copular verb, V_{mt}: Monotransitive verb, V_{dt}: Ditransitive verb, V_{ct}: Complex-transitive verb

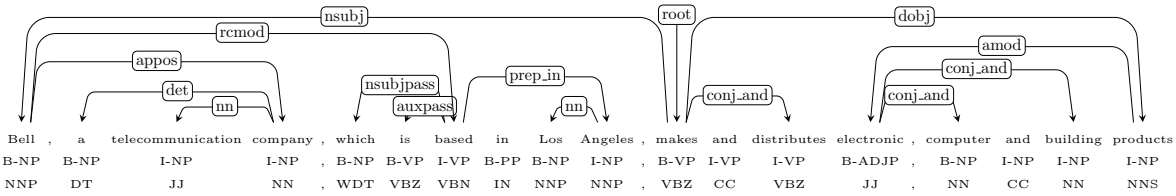


Figure 2.1 An example sentence with dependency parse, chunks, and POS tags (chunks by Apache OpenNLP)

2.3 ClausIE

We now describe how we obtain and subsequently exploit clauses and clause types in ClausIE. For each input sentence, ClausIE conducts the following steps:

1. Compute the DP of the sentence (Sec. 2.3.1).
2. Determine the set of clauses using the DP (Sec. 2.3.2)
3. For each clause, determine the set of coherent derived clauses based on the DP and small, domain-independent lexica (Sec. 2.3.3).
4. Generate propositions from (a subset of) the coherent clauses (Sec. 2.3.4).

The overall runtime of ClausIE is dominated by dependency parsing in step 1; steps 2–4 are inexpensive. Since ClausIE is a principled method, its accuracy will greatly depend on the ability of the parser in step 1 to correctly determine the grammatical structure of the sentence.

2.3.1 Step 1: Dependency Parsing

ClausIE makes use of the unlexicalized Stanford dependency parser (Klein and Manning, 2003) to discover the syntactical structure of an input sentence. The DP consists of a set of directed syntactic relations between the words in the sentence. The root of the DP is either a non-copular verb or the subject complement of a copular verb. For instance, in sentence "*Messi plays football*", word "*plays*" forms the root of DP; it is connected to "*Messi*" via a subject relation (`nsubj`) and to "*football*" via the direct-object relation (`dobj`). A more complex example is shown in Fig. 2.1; a complete list of relations can be found in de Marnee and Manning (2012).

2.3.2 Step 2: From Dependencies to Clauses

We first identify the clauses in the input sentence, i.e., we aim to obtain the head word of all the constituents of each clause. For example, we obtain (S: "Bell", V: "makes",

O: “products”) for the main clause of the sentence shown in Fig. 2.1. We use a simple mapping of dependency relations to clause constituents. First, we construct a clause for every subject dependency in the DP (e.g., `nsubj`); the dependant constitutes the subject (S) and the governor the verb (V).[3] All other constituents of the clause are dependants of the verb: objects (O) and complements (C) via `dobj`, `iobj`, `xcomp`, or `ccomp`; [4] and adverbials (A) via dependency relations such as `advmod`, `advcl`, or `prep`.

To cope with non-verb-mediated relations, and to improve recall and informativeness of extractions, ClausIE additionally creates a number of “synthetic clauses”, i.e., clauses that do not directly appear in the sentence. In subsequent steps, these synthetic clauses are treated in the same way as the actual clauses of the sentence. As discussed below, the constituents of a synthetic clause either refer to a word in the DP or correspond to an artificially created verb. In more detail, we replace the relative pronoun (e.g., “who” or “which”) of a relative clause by its antecedent, which is obtained via `rcmod` dependency to the governor of the relative pronoun. The replacement of relative pronouns aims to increase the informativeness of extractions; e.g., we obtain (S: “Bell”, V: “based”, A: “Angeles”) instead of (S: “which”, V: “based”, A: “Angeles”) in Fig. 2.1. ClausIE also handles non-verb-mediated extractions to a limited extent: We create synthetic clauses for appositions (`appos`) and possessives (`pos` or via the pronoun “whose”; see above). The so-obtained clauses use an artificial verb such as ’is’ (typed as copula) or ’has’ (typed as monotransitive), respectively. In our example, we obtain clause (S: “Bell”, V_c: ’is’, C: “company”) in this way, where words within single quotation marks refer to an artificial verb and without with quotation marks refer to a word in the (DP of the) original sentence. Finally, we generate a synthetic clause from participial modifiers (`partmod`), which indicate reduced relative clauses. The dependant of a participial modifier relation is a participial verb form, which we combine with an artificial verb such as “are” to obtain the “verb” of the synthetic clause (typed SVA). For example, we obtain from sentence “Truffles picked during the spring are tasty” the synthetic clause (S: “Truffles”, V: “are picked”, A: “[during the] spring”).

In summary, we identify the following clauses for the sentence of Fig. 2.1:

(S: “Bell”, V: “makes”, O: “products”),
(S: “Bell”, V: “based”, A: “Angeles”),
(S: “Bell”, V_c: “is”, C: “company”).

[3] Except for the SVC clause type. Here the governor of the subject dependency is the complement (C), and both verb and adverbials are dependants of the complement.

[4] For SVOC clauses, complements may appear as dependants of the direct object.

2.3.3 Step 3: Identifying Clause Types

Once clauses have been obtained, ClausIE tries to identify the type of each clause (recall Tab. 2.1). As argued in Sec. 2.2, we can combine knowledge of properties of verbs with knowledge about the structure of the input clause. Our approach to clause-type detection can be viewed as a decision tree, i.e., we ask a number of question whose answers ultimately determine the clause type. The decision tree is shown as Fig. 2.2; here questions Q_1–Q_3 and Q_7–Q_9 refer to the clause structure; questions Q_4, Q_5, and Q_{10} to verb properties, and questions Q_6 and Q_{11} deal with ambiguous cases. We describe each of these questions in detail below, and also discuss some techniques that help dealing with potential errors in the DP. After clause types have been identified, we mark all optional adverbials. In our example of Fig. 2.1, we obtain

(S: “Bell”, V: “makes”, O: “products”),
(S: “Bell”, V: “based”, A!: “Angeles”),
(S: “Bell”, V: “is”, A!: “company”),

where “A!” indicates essential adverbials and “A?” indicates optional adverbials.

Clause types SVC, SVOO, and SVOC are identified solely by the structure of the clause; all adverbials are optional for these types. For example, if a clause does not contain an object (Q_1) but does contain a complement (Q_2), it must be of type SVC. For example, we identify S_{10} of Tab. 2.1 as SVC so that its adverbial “of the 20th century” is optional.

If the sentence contains neither object nor complement, we are left with distinguishing clause types SV (intransitive verb) and SVA (extended-copular verb), a more difficult task. In many cases, the distinction can be performed accurately. We say that an adverbial is a *candidate adverbial* (for an essential adverbial) if it (1) is a dependant of the verb and (2) appears to the right of the verb. If the clause does not contain a candidate adverbial (Q_3), it is of type SV; e.g.,“The year after, AE succeeded”. Otherwise, ClausIE makes use of two lexica of verb types: a lexicon of verbs that are known to be non-extended-copular (Q_4, implies SV) and a lexicon of verbs known to be extended-copular (Q_5, implies SVA).[5] E.g., the adverbial in “AE remained in Princeton” is identified as essential since “remain” is a copular verb. If both dictionaries fail, we cannot determine the clause type accurately. In its default configuration, ClausIE then proceeds conservatively (Q_6), i.e., it assumes SVA to avoid marking an essential adverbial as optional.

[5]Note that these lexica can be learned automatically by observing which verbs appear (sufficiently frequently) without a candidate adverbial in the text collection or by using resources such as WordNet frames. We did not yet employ such techniques; our current implementation makes use of only a small hand-crafted dictionary of 31 extended-copular verbs (e.g., “be” or “remain”) and two non-extended copular verbs (“die” and “walk”). The dictionary is not hard-wired into ClausIE though, and can be eventually extended.

We proceed to distinguishing SVO and SVOA for clauses that neither contain a complement nor both a direct and an indirect object (SVOO). If the clause does not have a candidate adverbial (Q_9), we mark it as SVO. Similarly, if the clause has an indirect object (but not a direct object, Q_9), it cannot be of type SVOA (Quirk et al., 1985) so that we also mark it SVO; e.g., as in "He taught his students passionately". Otherwise, the clause contains both a direct object (but no indirect object and no complement) and a candidate adverbial. The distinction between SVO and SVOA is difficult in this (quite common) case; e.g., S_{11} (SVO) and S_6 (SVOA) in Tab. 2.1. Here we proceed heuristically. First, ClausIE accepts a lexicon of verbs that are potentially complex-transitive (Q_{10}) and outputs SVOA if the verb appears in the lexicon.[6] Otherwise, in ClausIE's default configuration, we proceed greedily (Q_{11}) and choose SVO, i.e., we mark the adverbial as optional.

Dependency tree ClausIE performs a number of additional steps in order to deal with design choices and errors of the Stanford parser.

We first discuss how we deal with clauses that have constituents of clausal form. The Stanford parser outputs dependency relations such as `xcomp` or `ccomp` for the object and the complement of a clause if they have a clausal form. We treat these dependencies as complements if the verb appears in our lexicon of copular verbs, and treat them as objects (or object complements) otherwise. If the clause additionally contains an indirect object, the parser outputs `dobj` instead of `iobj`. In this case, we cannot distinguish between SVOO and SVOC. Since we are ultimately interested in optional adverbials, and since all adverbials are optional for both SVOO and SVOC, we still obtain correct extractions. The Stanford parser sometimes places the object complement within the direct object. If this happens, we may determine clause types SVO or SVOA instead of SVOC. In both cases, extractions are coherent; if we detect SVOA, however, an optional adverbial is incorrectly flagged as essential. Finally, the parser outputs relation `dep` when it is unable to identify the type of a relation. ClausIE avoids processing the dependant of `dep` in verbal phrases to reduce potential extraction errors.

[6] The lexicon currently contains 15 verbs (e.g., "put" and "get").

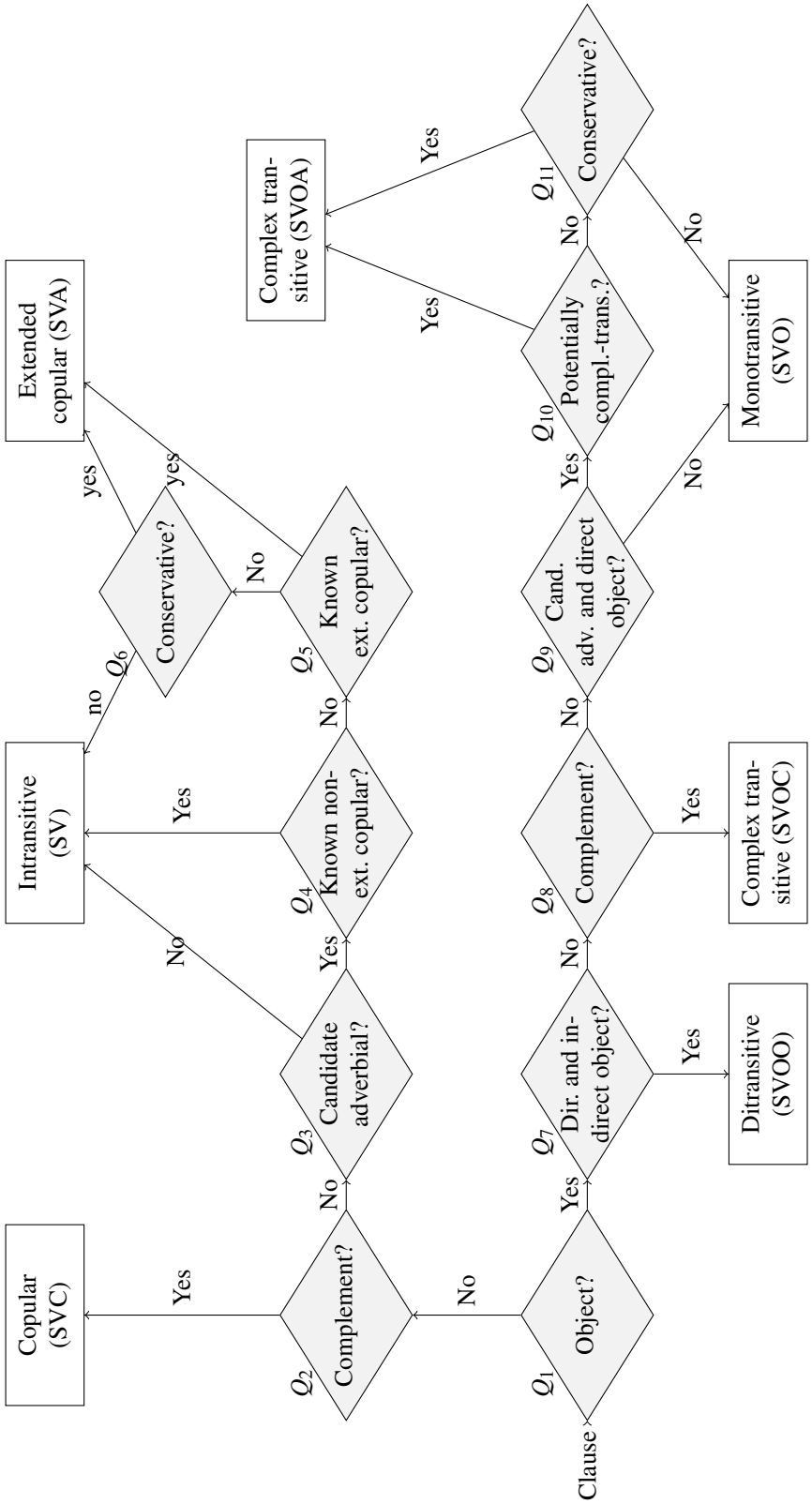


Figure 2.2 Flow chart for verb-type and clause-type detection

2.3.4 Step 4: From Clauses to Propositions

As a consequence of ClausIE's separation of clause and clause-type detection from proposition generation, the latter is flexible and can be customized to the application. In the current version, there are two basic steps involved in proposition generation. The first step is to decide which (combinations of) constituents form a proposition; the second step then generates the proposition from the constituents.

Constituent selection. Recall that a proposition consists of a subject, a relation, and zero, one, or more arguments. A natural choice is to generate n-ary propositions that consist of all the constituents of the clause, potentially with some arguments being marked optional. ClausIE supports generation of such n-ary propositions, but in addition allows to generate triple propositions, i.e., propositions that consist of a subject, a relation, and a (potentially empty) argument.[7] In fact, the concept of a triple (or binary relation) is fundamental to the semantic web, most ontological knowledge bases, and most OIE systems. A key question is which constituents should be included into the generated triple. For the setting we use here, we take a pragmatic approach: We do not only generate a single triple from each clause, but allow for the generation of multiple triples, each exposing different pieces of information. Consider for example the clause (S: "AE", V: "died", A?: "[in] Princeton", A?: "[in] 1955") obtained from S_8 in Tab. 2.1. Since both adverbials are marked optional, we can select four coherent derived clauses:

(S: "AE", V: "died"),
(S: "AE", V: "died", A: ["in"] "Princeton"),
(S: "AE", V: "died", A: ["in"] "1955"),
(S: "AE", V: "died", A: ["in"] "Princeton", A: "[in] 1955").

In general, if there are n optional adverbials, there are 2^n coherent derived clauses. To avoid over-specified triples, our default choice in ClausIE—which we also used in our experiments—is to select at most one optional adverbial (and all essential constituents). ClausIE also makes use of a lexicon consisting of small set of adverbials to be always omitted (e.g., "so") or included (e.g., "hardly") when optional.

Coordinated conjunctions (CC). A *coordinated conjunction* is a conjunction that connects two or more parts of the sentence—called *conjoints*—via a coordinator such as "and"

[7] In OIE, the argument component of a triple is often called "object"; e.g., "1921" in ("AE", "has won the NP in", "1921"). Here we avoid the term object for the argument of a triple to avoid confusion with the grammatical object of the clause.

or "or". CCs are detected by the Stanford parser and indicated by dependency relations such as `conj`. If a CC is present in a constituent of a clause, ClausIE optionally *processes* the CC, i.e., replaces the CC by each of its conjoints to avoid over-specified extractions. Consider the example sentence shown in Fig. 2.1. There is a CC in the verb constituent ("makes and distributes") and in the object constituent ("electronic, computer, and building products") of the main clause. By replacing CCs by conjoints, we obtain the following clauses:

(S: "Bell", V: "makes", O: "[electronic] products"),
(S: "Bell", V: "makes", O: "[computer] products"),
(S: "Bell", V: "makes", O: "[building] products"),
(S: "Bell", V: "distributes", O: "[electronic] products"),
(S: "Bell", V: "distributes", O: "[computer] products"),
(S: "Bell", V: "distributes", O: "[building] products").

The processing of CCs is closely related to text simplification (Evans, 2011); we can view the resulting clauses as simpler versions of the original clauses.

Note that in noun phrases, the replacement of a CC by one of its conjoints may lead to incorrect extractions when the CC is *combinatory* (as opposed to *segregatory*). For example, the CC in "Anna and Bob married each other" is combinatory; thus an extraction such as "Anna married each other" is incoherent. If the CC has an ampersand as coordinator, ClausIE treats it as combinatory and thus does not process it (e.g., "Standard & Poor's"). Similarly, CCs headed by words such as "between" are not processed (e.g., "between Norway and Finland"). In all other cases, the CC is treated as segregatory and thus processed. Combinatory CCs are rare in some domains (Evans, 2011), but may occur frequently in others. Since combinatory CCs are hard to detect (in some cases even for humans), ClausIE exposes an option to disable processing of CCs.

Finally, ClausIE treats CCs with preconjuncts (`preconj` dependency; e.g., "both [red and blue]") and (pre)determiners ((`pre`)`det`; e.g., "both [the boys and the girls]") specially. In particular, we omit all preconjuncts and some (pre)determiners (like "between" or "both") when processing a CC. For example, we extract from "Anna likes both red and blue" the propositions ("Anna", "likes", "red") and ("Anna", "likes", "blue").

Proposition generation. ClausIE generates one proposition for each selected subset of constituents. To generate a proposition, ClausIE needs to decide which part of each constituent to place into the subject, the relation, and the arguments. The perhaps simplest option is to first generate a textual representation of each constituent in its entirety and then use these representations to construct the proposition. We map the subject (verb) of each clause

to the subject (relation) of the proposition. When *n*-ary propositions are extracted, we create an argument for each of the remaining constituents (first all constituents following the verb, then all constituents preceding the verb, in the order in which they appear). To extract triples, we concatenate all arguments. From the sentence of Fig. 2.1, ClausIE extracts the following triples:

(“Bell”, ’is’, “a telecommunication company”),
(“Bell”, “is based”, “in Los Angeles”),
(“Bell”, “makes”, “electronic products”),
(“Bell”, “makes”, “computer products”),
(“Bell”, “makes”, “building products”),
(“Bell”, “distributes”, “electronic products”),
(“Bell”, “distributes”, “computer products”),
(“Bell”, “distributes”, “building products”).

As it was mentioned before, since the materialization of the information is not constrained by the detection phase, the system is easily customizable. For example, we can eventually obtain extractions similar to Reverb (Fader et al., 2011) by appending all but the final argument into the relation; if the final argument is a prepositional phrase, we also include the preposition into the relation. Another natural direction is to analyze the composition of each constituent in order to generate alternative textual representations. For instance in the sentence “Albert Einstein from Ulm won the Nobel Prize”, one could generate propositions such as (“AE from Ulm”, “won”, “the Nobel Prize”) or probably more appropriate (“AE”, “won”, “Nobel Prize”). A natural, extension of ClausIE is to wok on the proposition generation so that applications can customize each constituent in the proposition according to their needs. For instance, some applications may want to avoid generating large arguments or the use of unnecessary modifiers (e.g. adjectives). This future work could be tackled in a principled way by taking into account the hierarchies of the dependency trees which defines a degree of importance between the words.

2.4 Experiments

We conducted an experimental study to compare ClausIE to a number of alternative approaches. We found that ClausIE achieved significantly higher recall than the OIE extractors we compared to. Moreover, ClausIE consistently provided higher precision than alternative extractors over all levels of recall.

2.4.1 Experimental Setup

We first describe the datasets and the methodology used in our experiments.[8] We compared ClausIE to TextRunner (Banko et al., 2007), Reverb (Fader et al., 2011), WOE (Wu and Weld, 2010) (using DP), OLLIE (Mausam et al., 2012) and KrakeN (Akbik and Löser, 2012); neither extractions nor source code of any other extractor were available to us. Since most of OIE methods make use of machine-learning techniques, which require sensibly-chosen training data, or may need tweaking to provide good extractions, we did not compare ClausIE to these other OIE extractors. In all our experiments, we used the unlexicalized version of the Stanford DP (version 2.0.4). We configured ClausIE to generate triple propositions and ran it both with and without processing of coordinated conjunctions in subjects and arguments (denoted "ClausIE" and "ClausIE w/o CCs," respectively); coordinated conjunctions in verbal phrases were processed in both configurations.

We used three different datasets in our experiments. First, the Reverb dataset[9] consists of 500 sentences with manually-labeled extractions for TextRunner, TextRunner trained using Reverb, Reverb, OLLIE, and WOE. The sentences have been obtained via the random-link service of Yahoo and are generally very noisy. Second, we extracted 200 random sentences from Wikipedia pages. These sentences are shorter, simpler, and less noisy than those of the Reverb dataset. Since some Wikipedia articles are written by non-experts, however, the Wikipedia sentences do contain some incorrect grammatical constructions. Finally, we extracted 200 random sentences from the New York Times collection (NYT (Sandhaus, 2008)); these sentences are generally very clean but tend to be long and complex.

We manually labeled the extractions obtained from all extractors. To maintain consistency among the labels, the entire set of extractions of TextRunner, WOE, and Reverb for the Reverb dataset was relabeled; the precision numbers obtained using our labels closely agreed with those obtained using the original labels. For the Wikipedia and NYT datasets, we compare ClausIE with only Reverb and OLLIE, for which an extractor was publicly available. Each extraction was labeled by two independent labelers; an extraction was treated as correct only if it was labeled as correct by both labelers. Since we are primarily interested in the ability of OIE to capture verb-mediated propositions, labelers were instructed to ignore the *context* of the clause during labeling. For example, in the sentence "But inexpensive point-and-shoot cameras can do the job if they have a telephoto setting or a zoom lens", the proposition ("inexpensive point-and-shoot cameras", "can do", "the job") is treated as a correct extraction. We also asked labelers to be liberal w.r.t. coreference or entity resolution;

[8] All datasets, extractions, labels, as well as ClausIE's source code are available at http://people.mpi-inf.mpg.de/~corrogg/.

[9] http://reverb.cs.washington.edu/

e.g., a proposition such as (“he”, ’has’, “office”), or any unlemmatized version thereof, is treated as correct. Finally, we instructed labelers to label as incorrect relations that were overly specific, i.e., that contained named entities or numbers, or were excessively long (e.g., “has reported 1993 events in Moscow in”). We measured the agreement between labelers in terms of Cohen’s Kappa (Scott’s Pi). The score was 0.57 (0.57) for the Reverb dataset, 0.68 (0.68) for the Wikipedia dataset, 0.63 (0.63) for the New York Times dataset. The lower agreement score for the Reverb data might be attributed to the high amount of noise in the input sentences, which made it hard to judge the correctness of some of the extractions.

We used the absolute number of extractions instead of recall since it is infeasible to obtain the set of “all” correct propositions. For ClausIE, we determined the total number of extractions but also the number of non-redundant extractions (marked “non-red.”), i.e., extractions not “contained” in other extractions. For example, ClausIE extracts from sentence “AE remained in Princeton until his death” propositions (“AE”, “remained”, “in Princeton”) and (“AE”, “remained”, “in Princeton until his death”); the former extraction is marked redundant. We ordered all extractions by decreasing confidence; for ClausIE, we took the confidence of the DP as obtained by the Stanford parser as the confidence of a proposition. For KrakeN, extractions were unavailable to us; we reproduce the information provided in Akbik and Löser (2012) instead.

2.4.2 Example Extractions

We first illustrate the differences between the extractors for some manually-selected example sentences; Tab. 2.4 shows the extractions of each OIE extractor for a sentence of each of the datasets.

On the Reverb sentence, all OIE extractors agree on proposition R_1, which is correct. Reverb obtains a second proposition R_2, which is incorrect; it is[obtained because Reverb restricts subjects to noun phrases without prepositions and thus incorrectly omits “the only other name on”. In contrast, ClausIE identifies the subject correctly and hence extracts a correct proposition (R_{20}); it exploits access to the DP, which is (deliberately) not used by Reverb. WOE and OLLIE also make use of the DP, but still fail to identify the subject of the second clause correctly (R_5 and R_{11}, respectively), perhaps due to their use of automatically learned DP patterns (e.g., OLLIE learns from Reverb). For this reason, OLLIE also produces a number of additional incorrect extractions. Note that propositions R_{18} and R_{20} produced by ClausIE are labeled as redundant. As argued below, redundant extractions may be valuable by themselves due to their simpler structure.

On the Wikipedia dataset, almost all of the extractions are correct; ClausIE extracts the largest number of propositions, followed by OLLIE and Reverb. OLLIE misses the

essential adverbial "in Aberdeen" in proposition W_5, but still produces a correct (although tautological) proposition. ClausIE produces incorrect proposition W_{11} due to an error in the dependency parse (which does not associate "from Tuberculosis" with "death" but with "lived"). Proposition W_{12} was labeled correct (although this is arguable); here "his" refers to "he", and "has" is our synthetic verb for a possessive. Finally, ClausIE produces propositions W_6–W_8 due to its processing of the coordinated conjunctions. In this particular case, the parser identified "two children", "Edna", and "Donald" incorrectly as conjoints; otherwise propositions W_7 and W_8 would not have been generated.

Finally, on the NYT dataset, Reverb produces incorrect proposition N_2 by incorrectly identifying the argument. Reverb is designed to extract at most one prepositional phrase following the verb and thus misses "outside the United States". It also misses "in NATO" due its use of a lexical constraint (i.e., the phrase "includes the biggest standing army in", which is over-specified, does not appear sufficiently frequently in the corpus). ClausIE creates a correct and an incorrect (but coherent) proposition (N_{13} and N_{12}, resp.) from this clause of the sentence; the latter proposition is incorrect due to an error in the DP parse (which does not correctly associate "in NATO outside the United States" with "army"). ClausIE also produces three additional incorrect propositions (N_{15}–N_{17}). Proposition N_{15} has an incorrect subject due to an incorrect DP, propositions N_{16} and N_{17} are non-informative and thus labeled as incorrect (here we labeled conservatively). The sentence also contains a possessive, which is processed correctly by ClausIE to obtain proposition N_{14}. Finally, OLLIE extracts three incorrect propositions with an over-specified relation (N_3–N_5), and incorrect proposition N_6 due to a noisy extraction pattern.

2.4.3 Precision and Number of Extractions

Our results are summarized in Tab. 2.2 and Fig. 2.3. Tab. 2.2 shows the total number of correct extractions as well as the total number of extractions for each method and dataset. Fig. 2.3 plots the precision of each OIE extractor as a function of the number of extractions (ordered by decreasing confidence).

We found that in its default configuration, ClausIE produced 2.5–3.5 times more correct extractions than OLLIE, the best-performing alternative method. This increase in recall is obtained because ClausIE considers all adverbials in a clause (instead of only the one following the verb), extracts non-verb-mediated propositions, detects non-consecutive constituents, processes coordinated conjunctions, and outputs triples with non-noun-phrase arguments. Roughly 27–29% of the extractions of ClausIE were redundant. We believe that redundant extractions can be valuable: Even though a non-redundant proposition expresses more information, the corresponding redundant propositions has a simpler structure and is

	Reverb dataset			Wikipedia dataset			NYT dataset		
	C	I	P	C	I	P	C	I	P
ClausIE	**1706**	2975	57.34	**598**	1001	59.74	**696**	1303	53.42
ClausIE w/o CCs	1466	2344	**62.54**	536	792	**67.67**	594	926	64.15
ClausIE (non-redundant)	1221	2161	56.50	424	727	58.32	508	926	54.86
ClausIE w/o CCs (non-redundant)	1050	1707	61.51	381	569	66.96	444	685	**64.82**
OLLIE	547	1242	44.04	234	565	41.42	211	497	42.45
Reverb	388	727	53.37	165	249	66.27	149	271	54.98
WOE	447	1028	43.48	–	–				
TextRunner (Reverb)	343	837	40.98	–	–				
TextRunner	286	798	35.84	–	–				

C: Correct, I: Incorrect, P: Precision

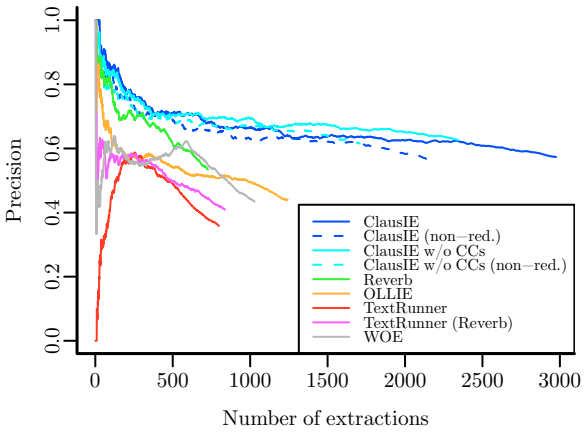
Table 2.2 Number of correct extractions and total number of extractions

easier to deal with. When redundant extractions are removed, ClausIE produces 1.8–2.4 times more correct extractions than OLLIE.

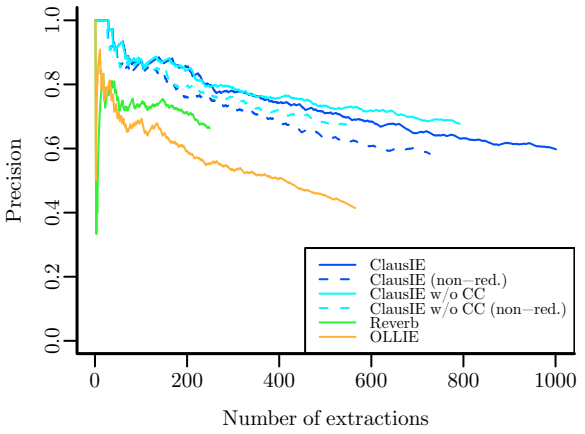
The precision of TextRunner was significantly lower than that of Reverb, WOE, and ClausIE. The latter three extractors obtain high precision on high-confidence propositions; the precision drops as we include more and more low-confidence propositions. In the case of ClausIE, the precision dropped quickly initially but then stabilized at between 53% and 60% (whether or not we include redundant propositions). Except for the Wikipedia dataset, the precision over all extractions obtained by ClausIE was higher than that of any other method, and ClausIE extracted significantly more propositions.

We also ran a configuration of ClausIE in which processing of coordinated conjunctions in subjects and arguments was disabled. This resulted in an increase of precision between 5% and 10.7% (on Wikipedia). Thus ClausIE's processing of CCs is somewhat error-prone, partly due to the presence of combinatory conjunctions and partly due to errors in the dependency parse. Nevertheless, when CCs are not processed, the number of extractions dropped significantly (between 11% and 27%), so that CC processing appears to be beneficial overall.

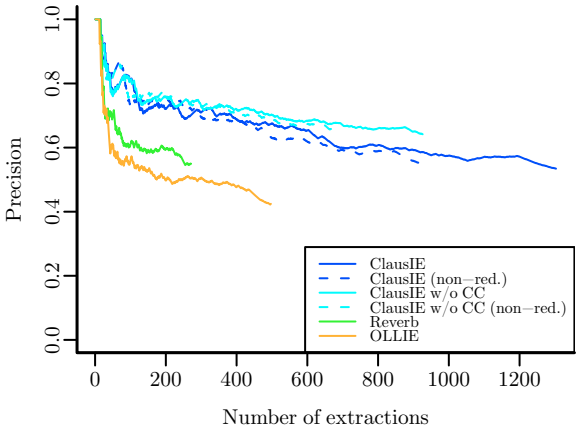
According to Akbik and Löser (2012), KrakeN extracts 572 propositions from the Reverb data; 308 of these propositions were correct and complete, 81 were correct but not complete. Note that KrakeN extracts n-ary propositions, whereas our experiments focus on triples (which cannot be produced by KrakeN for $n > 3$). Note that KrakeN did not extract propositions from dependency parses that contained the `dep` relation (i.e., an unknown dependency); this was true for 155 out of the 500 sentences in the Reverb data. ClausIE handles such cases gracefully, e.g., by extracting propositions from clauses that appear unaffected by the unknown dependency.



(a) Reverb dataset



(b) Wikipedia dataset



(c) New York Times dataset

Figure 2.3 Experimental results

The high recall and consistently good precision of ClausIE observed in our experiments indicates that reasoning over clauses and clause-types is a viable approach to OIE.

2.4.4 Extractions Errors of ClausIE

We did a preliminary analysis of the results obtained by ClausIE. We found that in most of the cases, ClausIE's extraction errors were due to incorrect dependency parses (see Sec. 2.4.2 for examples). In some cases, the incorrect DP resulted from noise in the input sentences, such as bad grammatical forms or spurious words. Our hope is that potential future improvements in dependency parsing will also lead to higher-precision extractions obtained by ClausIE. Another source of imprecision of ClausIE was due to our processing of coordinated conjunctions; see the discussion in Sec. 2.4.3. On the one hand, the Stanford DP parser tended to produce erroneous parses in the presence of CCs. On the other hand, when the coordinated conjunction was combinatory, the extractions obtained by ClausIE were incorrect. ClausIE also misclassified some SVOA clauses as SVO and thus omitted an essential adverbial. As mentioned previously, it is often hard to distinguish SVO from SVOA; an improved dictionary of potentially complex-transitive verbs may help to avoid some of these extraction errors. Moreover, Quirk (Quirk et al., 1985) notes that adverbials in SVA and SVOA clauses are largely restricted to space adjuncts, which may also help in identifying such clauses. Finally, this problem is alleviated to some extent if ClausIE is configured to produce n-ary extractions; then essential adverbials will not be omitted, although they can potentially be flagged as optional.

2.5 Related Work

Open Information Extraction. The task of OIE was introduced by the seminal work of Banko et al. (2007), who also proposed the TextRunner OIE system. A number of methods have been developed to improve on TextRunner. At a high level, all of these approaches make use of a set of *patterns* in order to obtain propositions. Depending on the specific approach, these patterns are either hand-crafted (based on on various heuristics) or learned from automatically generated training data (e.g., in the form of a classifier); the patterns apply to POS tags, chunks, DPs and parse trees. Early work mostly explored the use shallow syntactic analysis as POS tags and chunks, however, more recent work has focused on the use of deeper syntactic analysis.

The degree of syntactic analysis that should be used in information extraction methods has been discussed in the natural language processing (NLP) community (Bunescu and Mooney,

2005; Wilks and Brewster, 2009; Zouaq et al., 2009) and it is reasonable to assume that more syntactic information has a positive impact on extraction accuracy. Our experiments confirm this hypothesis. Moreover, as new syntactic analyzers keep appearing with improvements in both parsing precision and processing speed, as well as the increasing availability of hardware resources for heavy processing, it seems reasonable to work on systems using as input deep syntactic analysis. Deeper syntactic analysis provides a more complete picture of the syntactic structure of the sentence.

Most of the existing approaches aim to extract *triples*, i.e., propositions of form (subject, relation and argument); extraction of higher-arity propositions are handled by Akbik and Löser (2012) and Christensen et al. (2010). ClausIE also makes use of a set of hand-crafted patterns, which are constructed in a principled way, and can generate either triples or higher-arity propositions. Unlike ClausIE, all OIE systems do not separate the detection of the information from its representation. Tab. 2.3 provides a summary of the existing OIE systems.

Systems using shallow syntactic parsing. As it was previously mentioned, there are two major categories of OIE systems: approaches that make use of only shallow syntactic parsing, and approaches that apply heavier NLP technology. TextRunner (Banko et al., 2007) belongs to the former class. It first trains a Bayes classifier based on DPs of 1000s of sentences in an offline phase; the classifier is then applied to efficiently extract propositions in an online phase. The training data is automatically generated; it corresponds to triples based on short paths heuristics. Shortest paths are a popular heuristic to generate triples from raw text (Bunescu and Mooney, 2005). In a shortest path, the relation between two entities is formed by the minimum number of words, containing a verb, that connects them. The words are collected by following the grammatical relations in a DP ignoring their directions. The starting and ending point of the path are the entities that serve as the arguments of the triple. Specifically in the case of TextRunner, the shortest path is relaxed so that it does not only search for the shortest path but for a path of a minimum length: a triple is considered as a positive example if the path connecting them does not cross clauses, does not contain solely a pronoun and is not bigger than a certain length.

WOE^{pos} (Wu and Weld, 2010) also uses a classifier, but the classifier is based on a high-quality training corpus obtained automatically from Wikipedia for improved precision and recall. The training data is formed by triples generated via shortest path. In contrast, although ClausIE also generates extractions from DPs it does not rely on the extensively used shortest path heuristic but on a principled technique based on well founded linguistic theories. This means that ClausIE could be also used to generate training data for OIE classifiers or any other task requiring triples as input.

Reverb (Fader et al., 2011) is the perhaps simplest (and thus very attractive) shallow extractor; it makes use of syntactical and lexical constraints that aim to reduce the amount of uninformative, incoherent, and over-specified extractions (see Sec. 2.2). Reverb defines a simple pattern based on shallow syntax which allows to capture at once, at a general level, some of the relevant clause types that we present in this chapter. However, as it works at a shallow syntactic level it is unable to understand the complex internal structure of each extraction, so it makes mistakes derived from this simplification . This implies, that it is faster because it uses much less complex syntactic analysis but also less precise because it only attempts to target very generic patterns. Recall is lower because it misses patterns that cannot be generalized to this single pattern.

Finally, R2A2 (Etzioni et al., 2011) uses a number of classifiers to identify the arguments of a verbal phrase (based on hand-labeled training data), and is able to extract propositions that contain arguments beyond noun phrases. R2A2 is the best-performing shallow OIE extractor to date. ClausIE uses more complex syntactic information, and therefore it is slower, however, as it produces high-quality extractions it can potentially be used as training for R2A2.

Systems using deep syntactic parsing. The second category of OIE systems makes use of deep syntactic processing like dependency parsing (Akbik and Broß, 2009; Wu and Weld, 2010; Akbik and Löser, 2012; Mausam et al., 2012; Gamallo et al., 2012; Yahya et al., 2014) or parse trees (Bast and Haussmann, 2013). Some systems use either hand-labeled (Wanderlust (Akbik and Broß, 2009)) or automatically generated (WOE$^{\text{parse}}$ (Wu and Weld, 2010) and OLLIE (Mausam et al., 2012)) training data to learn extraction patterns on the dependency tree.

OLLIE is a semi-supervised dependency-based system. It generates dependency patterns via a bootstrapping process over a big corpus. The seeds are generated by extracting high confidence propositions from Reverb, and assuming that each sentence containing the subject and argument from those propositions, imply the extracted relation. OLLIE generates in this way a large corpus of DP-based patterns, which are used to extract triples from new input sentences. ClausIE, on the contrary, does not need to learn patterns and avoids mistakes that may be generated by the learning process or the assumption, common in semi-supervised models, that two co-occurring entities in a sentence express the relation observed in the annotated data. Our system does not need to learn those patterns or rely in any assumption.

Other approaches (KrakeN (Akbik and Löser, 2012), Gamallo et al. (2012) and Zouaq (2011)) use a set of hand-crafted patterns on the DP. In contrast to all existing approaches, ClausIE reasons about the set of clauses (and their types) that appear in an input sentence;

this reasoning is based on the DP and a small set of domain-independent lexica. Most of the patterns of Akbik and Löser (2012). Gamallo et al. (2012) and Zouaq (2011) are naturally captured by ClausIE. Moreover, ClausIE can be customized to output triples or n-ary propositions, and the conceptual separation between recognition and representation implies that it can be eventually adapted to focus on either noun-phrase or more general subjects and arguments, or to flexibly adjust how much information is included in the relational phrase and how much in its arguments.

Finally, Bast and Haussmann (2013) identifies building blocks in the sentence and applies a set of patterns to generate propositions, both verb and non-verb mediated. ClausIE covers in a principled way all verb based patterns generated by Bast and Haussmann (2013) and at the same time it is able to understand the internal structure of each extraction, which makes it more flexible in the representation of the information allowing potential customization if needed (e.g. reduce the size of the arguments by removing unnecessary modifiers).

OIE and Semantic Role Labeling. OIE is the perhaps simplest form of semantic analysis. A closely related and more general problem is semantic role labeling (SRL), which aims to identify arguments of verbs as well as their semantic roles. Christensen et al. (2010) have shown that SRL can be used to increase the precision and recall of OIE; however, existing SRL systems heavily rely on manually labeled training data and are posed to solve a task that, although related, goes semantically beyond OIE with the risk of lowering recall. OIE is a shallower semantic task that can be potentially used in the argument recognition step of SRL systems.

Non verb-based relations. OIE focus mainly on verb-mediated propositions. ClausIE is able to identify non-verb-mediated propositions, although to a limited extent, specialized techniques—such as Venetis et al. (2011) for extracting the "is-a" relationship—go significantly further. Renoun (Yahya et al., 2014), a recent system, intends to cover this gap by systematically extracting noun mediated propositions via a bootstrapping process. This pattern-based semi-supervised method attempt to extract propositions such as "president"(*"Barack Obama"*, *"US"*), where a noun acts as the relation between the entity arguments.

KB construction. A KB can be thought as a collection of facts, and these, in turn as disambiguated propositions. This means that OIE can be seen as the very first step in KB construction. Techniques for automated ontology construction (e.g., (Suchanek et al., 2009; Carlson et al., 2010; Zouaq, 2011; Wu et al., 2012)) require to first identify the proposition in

text before mapping it to an ontological fact. Since many KB construction approaches rely on a pre-specified set of entities or relations, OIE is the natural starting point for systems that go beyond the closed-world assumption. Patty (Nakashole et al., 2012), for example, aims to extract a set of typed lexical patterns that are indicative of a relation. It clusters relational phrases based on the types of its arguments, so that OIE would be the ideal framework to accurately extract those relational phrases.

2.6 Conclusion

OIE is the natural first step to any automatic text understanding task: it recognizes and structures the information in text in a way that can be managed and reasoned about by computers. We presented principled approach to open information extraction called ClausIE which relies on well founded linguistic theories. In contrast to previous approaches, ClausIE separates the detection of clauses and clause types from the actual generation of propositions; the detection of information is general for every text but the materialization of that information ultimately depends on the underlying application. ClausIE obtains more and higher-precision extractions than alternative methods and allows a flexible generation of propositions. ClausIE can be seen as a first step towards clause-based open information extraction. Potential improvements include construction of richer lexica, improved processing of the constituents of each clause to avoid over-specification in subjects and arguments, as well as context analysis to detect relations between clauses and the extraction of non-verb mediated relations.

ClausIE's accuracy mostly relies on a correct syntactic analysis of the sentences, which in our case is provided by a standard dependency parser, and on a set of dictionaries which contain known verbs for each verb type. However, each verb may in fact belong to different verb categories according to the different senses it has. This means that if we know the sense of the verb and its corresponding clause type, we can identify more accurately the clause type of the clause and therefore generate a more accurate structuring of the information. In the next chapter, we present a method to detect the sense of the verbs given a clause.

Paper	Type	Method	Target	Synt. processing	Verbal	Non-verbal	N-ary	Year
TextRunner	sup.	Naive Bayes	short paths	shallow	yes	no	no	2007
Wanderlust	sup.	frequent dep. paths	46 frequent paths	deep	yes	no	no	2009
R2A2	sup.	ILP	SRL based	deep	yes	no	yes	2010
WOE^{pos}	sup.	CRF	shortest path	shallow	yes	no	no	2010
WOE^{parse}	sup.	CRF	shortest path	deep	yes	no	no	2010
ReVerb	uns.	shallow pattern	NP-VP(P?)-NP pattern	shallow	yes	no	no	2011
KrakeN	uns.	dep. pattern	common dep. paths	deep	yes	no	yes	2012
Gamallo et al. (2012)	uns.	dep. pattern	5 common patterns	deep	yes	no	no	2012
OLLIE	semi-sup.	dep. patterns	frequent dep. patterns	deep	yes	yes	no	2012
ClausIE	uns.	dep. pattern	linguistic clause-based	deep	yes	yes	yes	2013
Bast and Haussmann (2013)	uns.	cont. tree. pattern	14 cont. tree patterns	deep	yes	yes	no	2013
Renoun	semi-sup.	dep. patterns	requent dep. patterns	deep	no	yes	no	2014

cont. tree: constituency tree; dep.: dependency; sup.:supervised; synt.: syntactic; semi-sup.:semi-supervised; uns.:unsupervised

Table 2.3 Open Information Extractors in order of appearence

System	#	Proposition	Label
		Reverb dataset	
The principal opposition parties boycotted the polls after accusations of vote-rigging , and the only other name on the ballot was a little-known challenger from a marginal political party.			
Reverb	R_1:	(“*The principal opposition parties*”, “*boycotted*”, “*the polls*”)	Correct
	R_2:	(“*the ballot*”, “*was*”, “*a little-known challenger*”)	Incorrect
TextRunner	R_3:	(“*The principal opposition parties*”, “*boycotted*”, “*the polls*”)	Correct
WOE	R_4:	(“*The principal opposition parties*”, “*boycotted*”, “*the polls*”)	Correct
	R_5:	(“*the only other name*”, “*was*”, “*a little-known challenger*”)	Incorrect
OLLIE	R_6:	(“*The principal opposition parties*”, “*boycotted*”, “*the polls*”)	Correct
	R_7:	(“*The principal opposition parties*”, “*boycotted the polls after*”, “*accusations of vote-rigging*”)	Correct
	R_8:	(“*The principal opposition parties*”, “*was*”, “*a little-known challenger*”)	Incorrect
	R_9:	(“*The principal opposition parties*”, “*was a little-known challenger from*”, “*a marginal political party*”)	Incorrect
	R_{10}:	(“*the polls*”, “*be boycotted after*”, “*accusations of vote-rigging*”)	Correct
	R_{11}:	(“*the only other name*”, “*was*”, “*a little-known challenger*”)	Incorrect
	R_{12}:	(“*the only other name*”, “*was a little-known challenger from*”, “*a marginal political party*”)	Incorrect
	R_{13}:	(“*the only other name*”, “*boycotted*”, “*the polls*”)	Incorrect
	R_{14}:	(“*the only other name*”, “*boycotted the polls after*”, “*accusations of vote-rigging*”)	Incorrect
	R_{15}:	(“*a little-known challenger*”, “*be the only other name on*”, “*the ballot*”)	Correct
	R_{16}:	(“*only*”, “*be other name on*”, “*the ballot*”)	Incorrect
	R_{17}:	(“*other*”, “*be name on*”, “*the ballot*”)	Incorrect
ClausIE	R_{18}:	(“*The principal opposition parties*”, “*boycotted*”, “*the polls*”)	Correct (red.)
	R_{19}:	(“*The principal opposition parties*”, “*boycotted*”, “*the polls after accusations of vote-rigging*”)	Correct
	R_{20}:	(“*the only other name on the ballot*”, “*was*”, “*a little-known challenger*”)	Correct (red.)
	R_{21}:	(“*the only other name on the ballot*”, “*was*”, “*a little-known challenger from a marginal political party*”)	Correct
		Wikipedia dataset	
He fathered two children, Edna and Donald, and lived in Aberdeen until his death from tuberculosis in 1942.			
Reverb	W_1:	(“*He*”, “*fathered*”, “*two children*”)	Correct
	W_2:	(“*two children*”, “*lived in*”, “*Aberdeen*”)	Correct
OLLIE	W_3:	(“*He*”, “*fathered*”, “*two children*”)	Correct
	W_4:	(“*He*”, “*lived in*”, “*Aberdeen*”)	Correct
	W_5:	(“*He*”, “*lived until*”, “*his death*”)	Correct
ClausIE	W_6:	(“*He*”, “*fathered*”, “*two children*”)	Correct
	W_7:	(“*He*”, “*fathered*”, “*Edna*”)	Correct
	W_8:	(“*He*”, “*fathered*”, “*Donald*”)	Correct
	W_9:	(“*He*”, “*lived*”, “*in Aberdeen*”)	Correct (red.)
	W_{10}:	(“*He*”, “*lived*”, “*in Aberdeen until his death*”)	Correct
	W_{11}:	(“*He*”, “*lived*”, “*in Aberdeen from tuberculosis in 1942*”)	Incorrect
	W_{12}:	(“*his*”, “*has*”, “*death*”)	Correct
		New York Times dataset	
Taken for granted it sometimes may be, but this year the Defense Department sought $950 million in assistance from Congress (and secured half that amount) for Ankara’s huge military machine, which includes the biggest standing army in NATO outside the United States.			
Reverb	N_1:	(“*the Defense Department*”, “*sought*”, “*$ 950 million*”)	Correct
	N_2:	(“*Ankara’s huge military machine*”, “*includes*”, “*the biggest standing army*”)	Incorrect
OLLIE	N_3:	(“*the Defense Department*”, “*sought $ 950 million in assistance from Congress half*”, “*(and secured half)*”)	Incorrect
	N_4:	(“*the Defense Department*”, “*sought $ 950 million in assistance from Congress in*”, “*this year*”)	Incorrect
	N_5:	(“*Ankara ’s huge military machine*”, “*includes the biggest standing army in NATO outside*”, “*the United States*”)	Incorrect
	N_6:	(“*the biggest standing army*”, “*be includes by*”, “*Ankara ’s huge military machine*”)	Incorrect
ClausIE	N_7:	(“*the Defense Department*”, “*sought*”, “*$ 950 million*”)	Correct (red.)
	N_8:	(“*the Defense Department*”, “*sought*”, “*$ 950 million in assistance*”)	Correct
	N_9:	(“*the Defense Department*”, “*sought*”, “*$ 950 million this year*”)	Correct
	N_{10}:	(“*the Defense Department*”, “*sought*”, “*$ 950 million for Ankara’s huge military machine*”)	Correct
	N_{11}:	(“*the Defense Department*”, “*sought*”, “*$ 950 million from Congress*”)	Correct
	N_{12}:	(“*Ankara’s huge military machine*”, “*includes*”, “*the biggest standing army in NATO*”)	Incorrect
	N_{13}:	(“*Ankara’s huge military machine*”, “*includes*”, “*the biggest standing army in NATO outside the United States*”)	Correct
	N_{14}:	(“*Ankara*”, “*has*”, “*huge military machine*”)	Correct
	N_{15}:	(“*Taken for*”, “*granted*”, “*it sometimes may be*”)	Incorrect
	N_{16}:	(“*it*”, “*may be*”)	Incorrect
	N_{17}:	(“*it*”, “*may be*”, “*sometimes*”)	Incorrect

Table 2.4 Example extractions from a sentence of each dataset

Chapter 3

Werdy: Recognition and Disambiguation of Verbs and Verb Phrases

3.1 Introduction

Understanding the semantics of words and multi-word expressions in natural language text is an important task for automatic knowledge acquisition. It serves as a fundamental building block in a wide area of applications, including semantic parsing, question answering, paraphrasing, knowledge base construction, etc. In this chapter, we study the task of word-sense recognition and disambiguation (WERD) with a focus on verbs and verbal phrases. Verbs are the central element in a sentence, and the key to understand the relations between sets of entities expressed in a sentence. In some occasions, disambiguating a verb is equivalent to the disambiguation of a relation in a proposition (e.g., ("Messi", "plays", "soccer")). Even more, as we show in the previous chapter, knowing the type of the verb, plays a key role in structuring the information in a sentence.

In this chapter, we present Werdy, a method to (i) automatically recognize in natural language text both single words and multi-word phrases that match entries in a lexical knowledge base (KB) like WordNet (Fellbaum, 1998), and (ii) disambiguate these words or phrases by identifying their senses in the KB. WordNet is a comprehensive lexical resource for word-sense disambiguation (WSD), covering nouns, verbs, adjectives, adverbs, and many multi-word expressions. In the following, the notion of an *entry* refers to a word or phrase in the KB, whereas a *sense* denotes the lexical synset of the entry's meaning in the given sentence.

A key challenge for recognizing KB entries in natural language text is that entries often consist of multiple words. In WordNet-3.0, more than 40% of the entries are multi-word. Such

entries are challenging to recognize accurately for two main reasons: First, the components of multi-word entries in the KB (such as *fiscal year*) often consist of components that are themselves KB entries (*fiscal* and *year*). Second, multi-word entries (such as *take a breath*) may not appear consecutively in a sentence ("He takes a deep breath."). Werdy addresses the latter problem by (conceptually) matching the syntactic structure of the KB entries to the syntactic structure of the input sentence. To address the former problem, Werdy identifies all possible entries in a sentence and passes them to the disambiguation phase (*take*, *breath*, *take a breath*, ...); the disambiguation phase provides more information about which multi-word entries to keep. Thus, our method solves the recognition and the disambiguation tasks jointly.

Once KB entries have been identified, Werdy disambiguates each entry against its possible senses. State-of-the-art methods for WSD (Navigli, 2009) work fairly well for nouns and noun phrases. However, the disambiguation of verbs and verbal phrases has received much less attention in the literature.

WSD methods can be roughly categorized into (i) methods that are based on supervised training over sense-annotated corpora (e.g., Zhong and Ng (2010)), and (ii) methods that harness KB's to assess the semantic relatedness among word senses for mapping entries to senses (e.g., Ponzetto and Navigli (2010)). For these methods, mapping verbs to senses is a difficult task since verbs tend to have more senses than nouns. In WordNet (including monosemous words) there are on average 1.24 senses per noun and 2.17 per verb.

To disambiguate verbs and verbal phrases, Werdy proceeds in multiple steps. First, Werdy obtains the set of candidate senses for each recognized entry from the KB. Second, it reduces the set of candidate entries using novel syntactic and semantic pruning techniques. The key insight behind our syntactic pruning is that each verb sense tends to occur in only a limited number of *syntactic patterns*. For example, the sentence "Albert Einstein remained in Princeton" has a subject ("Albert Einstein"), a verb ("remained") and an adverbial ("in Princeton"), and it follows an SVA (subject-verb-adverbial) clause pattern. We can thus safely prune verb senses that do not match the syntactic structure of the sentence. Moreover, each verb sense is compatible with only a limited number of *semantic argument types* (such as *location*, *river*, *person*, *musician*, etc); this phenomena is called *selectional preference* or *selectional restriction*. Senses that are compatible only with argument types not present in the sentence can be pruned. Our pruning steps are based on the idea that a verb selects the categories of its arguments both syntactically (c-selection) and semantically (s-selection). In the final step, Werdy employs a state-of-the-art general WSD method to select the most suitable sense from the remaining candidates. Since many incorrect senses have already been pruned, this step significantly gains in accuracy and efficiency over standard WSD.

Our semantic pruning technique builds on a newly created resource of pairs of senses for verbs and their object arguments. For example, the WordNet verb sense ⟨*play*-1⟩ (i.e., the 1st sense in WordNet of the verb entry "play") selects as direct object the noun sense ⟨*sport*-1⟩. We refer to this novel resource as the *VO Sense Repository*, or VOS repository for short.[1] It is constructed from the WordNet gloss-tags corpus, the SemCor dataset, and a small set of manually created VO sense pairs.

We evaluated Werdy on the SemEval-2007 coarse-grained WSD task (Navigli et al., 2007), both with and without automatic recognition of entries. We found that our techniques boost state-of-the-art WSD methods and obtain high-quality results. Werdy significantly increases the precision and recall of the best performing baselines.

3.2 Overview of Werdy

Werdy consists of four steps: (i) entry recognition, (ii) syntactic pruning, (iii) semantic pruning, and (iv) word-sense disambiguation. The contribution of this work is in the first three steps, and in the construction of the VO sense repository. Each of these steps operates on the clause level, i.e., we first determine the set of clauses present in the input sentence and then process clauses separately. A *clause* is a part of a sentence that expresses some statement or coherent piece of information. Clauses are thus suitable minimal units for automatic text understanding tasks (see chapter 2); see Sec.3.3 for details.

In the *entry-recognition step* (Sec. 3.3), Werdy obtains for the input sentence a set of potential KB entries along with their part-of-speech tags (POS). The candidate senses of each entry are obtained from WordNet. For instance, in the sentence "He takes a deep and long breath", the set of potential entries includes *take* (verb, 44 candidate senses), *take a breath* (verb, 1 candidate sense), and *breath* (noun, 5 candidate senses). Note that in contrast to Werdy, most existing word-sense disambiguation methods assume that entries have already been (correctly) identified.

In the *syntactic-pruning step* (Sec. 3.4), we eliminate candidate senses that do not agree with the syntactic structure of the clause. It is well-established that the syntactic realization of a clause is intrinsically related with the sense of its verb (Quirk et al., 1985; Levin, 1993; Hanks, 1996; Baker et al., 1998; Palmer et al., 2005). Quirk et al. (1985) identified seven possible clause types in the English language (such as "subject verb adverbial", SVA). We make use of techniques from chapter 2 to identify the clause type of each clause in the sentence. We then match the clause type with the set of WordNet frames (e.g., "somebody

[1] The VOS repository, Werdy's source code, and results of our experimental study are available at http://people.mpi-inf.mpg.de/~corrogg/.

verb something") that WordNet provides for each verb sense, and prune verb senses for which there is no match.

In the *semantic-pruning step* (Sec. 3.5), we further prune the set of candidate senses by taking the semantic types of direct objects into account. Similarly to the syntactic relation mentioned above, a verb sense also imposes a (selectional) restriction on the semantic type of its arguments (Quirk et al., 1985; Levin, 1993; Hanks, 1996; Baker et al., 1998; Palmer et al., 2005). For instance, the verb *play* with sense *participate in games or sports* requires an object argument of type $\langle$*game*-1$\rangle$, $\langle$*game*-3$\rangle$, or $\langle$*sport*-1$\rangle$. Senses that do not match the arguments found in the clause are pruned. This step is based on the newly constructed VOS Repository (Sec. 3.6). Note that when there is no direct object, only the syntactic pruning step applies.

3.3 Entry Recognition

The key challenge in recognizing lexical KB entries in text is that entries are not restricted to single words. In addition to named entities (such as people, places, etc.), KB's contain multi-word expressions. For example, WordNet-3.0 contains entries such as *take place* (verb), *let down* (verb), *take into account* (verb), *be born* (verb), *high school* (noun), *fiscal year* (noun), and *Prime Minister* (noun). Note that each individual word in a multi-word entry is usually also an entry by itself, and can even be part of several multi-word entries. To ensure correct disambiguation, all potential multi-word entries need to be recognized (Finlayson and Kulkarni, 2011), even when they do not appear as consecutive words in a sentence.

Werdy addresses these challenges by exploring the syntactic structure of both the input sentence and the lexical KB entries. The structure of the sentence is captured in a dependency parse (DP). Given a word in a sentence, Werdy conceptually generates all subtrees of the DP starting at that word, and matches them against the KB. This process can be performed efficiently as WordNet entries are short and can be indexed appropriately. To match the individual words of a sentence against the words of a KB entry, we follow the standard approach and perform lemmatization and stemming (Finlayson, 2014). To further handle personal pronouns and possessives, we follow Arranz et al. (2005) and normalize personal pronouns (I, you, my, your, ...) to *one's*, and reflexive pronouns (myself, yourself, ...) to *oneself*.

Consider the example sentence "He takes my hand and a deep breath". We first identify the clauses and their DP's (Fig. 3.1) using the method described in chapter 2, which also processes coordinating conjunctions. We obtain clauses "He takes my hand" and "He takes a deep breath", which we process separately. To obtain possible entries for the first clause, we

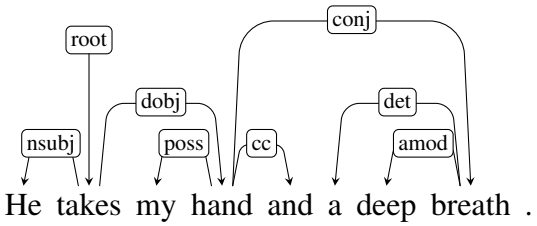


Figure 3.1 An example dependency parse

start with its head word (*take*) and incrementally consider its descendants (*take hand*, *take one's hand*, ...). The exploration is terminated as early as possible; for example, we do not consider *take one's hand* because there is no WordNet entry that contains both *take* and *hand*. For the second clause, we start with *take* (found in WordNet), then expand to *take breath* (not found but can occur together), then *take a breath* (found), then *take a deep breath* (not found, cannot occur together) and so on.

Note that the word "take" in the sentence refers to two different entries and senses: ⟨*take*-4⟩ for the first clause and ⟨*take a breath*-1⟩ for the second clause. In this stage no decisions are made about selecting entries and disambiguating them; these decisions are made in the final WSD stage of Werdy.

We tested Werdy's entry-recognizer on the SemEval-2007 corpus. We detected the correct entries for all but two verbs (out of more than 400). The two missed entries ("take up" and "get rolling") resulted from incorrect dependency parses.

3.4 Syntactic Pruning

Once KB entries have been recognized, Werdy prunes the set of possible senses of each verb entry by considering the syntactic structure of the clause in which the entry occurs. This pruning is based on the observation that each verb sense may occur only in a limited number of clause types, each having specific semantic functions (Quirk et al., 1985). When the clause type of the sentence is incompatible with a candidate sense of an entry, this sense is eliminated.

Werdy first detects in the input sentence the set of clauses and their constituents. Recall from chapter 2 that a clause consists of one *subject* (S), one *verb* (V), and optionally an *indirect object* (O), a *direct object* (O), a *complement* (C) and one or more *adverbials* (A). Not all combinations of clause constituents appear in the English language. When we classify clauses according to the grammatical function of their constituents, we obtain only seven different clause types (Quirk et al., 1985); see Tab. 3.1. For example, the sentence "He takes

Pattern	Clause type	Example	WN frame example [frame number]
SV_i	SV	AE died.	Somebody verb [2]
SV_eA	SVA	AE remained in Princeton.	Somebody verb PP [22]
SV_cC	SVC	AE is smart.	Somebody verb adjective [6]
$SV_{mt}O$	SVO	AE has won the Nobel Prize.	Somebody verb something [8]
$SV_{dt}O_iO$	SVOO	RSAS gave AE the Nobel Prize.	Somebody verb somebody something [14]
$SV_{ct}OA$	SVOA	The doorman showed AE to his office.	Somebody verb somebody PP [20]
$SV_{ct}OC$	SVOC	AE declared the meeting open.	Something verb something adjective/noun [5]

S: Subject, V: Verb, C: Complement, O: Direct object, O_i: Indirect object, A: Adverbial, V_i: Intransitive verb, V_c: Copular verb, V_c: Extended-copular verb, V_{mt}: Monotransitive verb, V_{dt}: Ditransitive verb, V_{ct}: Complex-transitive verb

Table 3.1 Clause types and examples of matching WordNet frames

my hand" is of type SVO; here "He" is the subject, "takes" the verb, and "my hand" the object. The clause type can (in principle) be determined by observing the verb type and its complementation.

For instance, consider the SVA clause "The student remained in Princeton". The verb *remain* has four senses in WN: (1) stay the same; remain in a certain state (e.g., "The dress remained wet"), (2) continue in a place, position, or situation ("He remained dean for another year"), (3) be left; of persons, questions, problems ("There remains the question of who pulled the trigger") or (4) stay behind ("The hostility remained long after they made up"). The first sense of *remain* requires an SVC pattern; the other cases require either SV or SVA. Our example clause is of type SVA so that we can safely prune the first sense.

WordNet provides an important resource for obtaining the set of clause types that are compatible with each sense of a verb. In particular, each verb sense in WordNet is annotated with a set of *frames* (e.g., "somebody verb something") in which they may occur, capturing both syntactic and semantic constraints. There are 35 different frames in total which are displayed in Tab. 3.2. We manually assigned a set of clause types to each frame (e.g., SVO to frame "somebody verb something"). Tab. 3.1 shows an example frame for each of the seven clause types. On average, each WordNet-3.0 verb sense is associated with 1.57 frames; the maximum number of frames per sense is 9. The distribution of frames is highly skewed: More than 61% of the 21,649 frame annotations belong to one of four simple SVO frames (numbers 8, 9, 10 and 11), and 22 out of the 35 frames have less than 100 instances. This skew makes the syntactic pruning step effective for non-SVO clauses, but less effective for SVO clauses.

Werdy directly determines a set of possible frame types for each clause of the input sentence. Our approach is based on the clause-type detection method from chapter 2, but we also consider additional information that is captured in frames but not in clause types. For example, we distinguish different realizations of objects (such as clausal objects from

Frame Number	**Frame**	**Examples**	Rel. Freq.
1	Something verb	The plane turns	0.0842
2	Somebody verb	He runs	0.1255
3	It is verb -ing	It is raining	0.0010
4	Something is verb -ing PP	The plane is going to Paris	0.0260
5	Something verb something Adjective/Noun	It makes it clear	0.0013
6	Something verb Adjective/Noun	It becomes difficult	0.0018
7	Somebody verb Adjective	He seems crazy	0.0016
8	Somebody verb something	He knows physics	0.3128
9	Somebody verb somebody	He believes the judge	0.1212
10	Something verb somebody	The airline helped the passengers	0.0498
11	Something verb something	The airline provided new planes	0.1144
12	Something verb to somebody	The airline reacted to the market	0.0006
13	Somebody verb on something	He agrees on some points	0.0014
14	Somebody verb somebody something	He told him the secret	0.005
15	Somebody verb something to somebody	He read a book to her	0.0091
16	Somebody verb something from somebody	He took a word from her	0.0035
17	Somebody verb somebody with something	He provided her with shelter	0.0033
18	Somebody verb somebody of something	The police accuse him of stealing	0.0014
19	Somebody verb something on somebody	He inflicted pain on him	0.0008
20	Somebody verb somebody PP	She asked him for a loan	0.0111
21	Somebody verb something PP	He used his influence to win	0.0243
22	Somebody verb PP	He comes into the office	0.0605
23	Somebody's (body part) verb	The leg hurts	0.0008
24	Somebody verb somebody to INFINITIVE	He wants him to win	0.0054
25	Somebody verb somebody INFINITIVE	He lets me win	0.0004
26	Somebody verb that CLAUSE	The president says that the situation is difficult	0.0170
27	Somebody verb to somebody	He talks to her	0.0016
28	Somebody verb to INFINITIVE	He wants to believe	0.0055
29	Somebody verb whether INFINITIVE	He will decide whether to come or not	0.0013
30	Somebody verb somebody into V-ing something	He talked her into doing that	0.0010
31	Somebody verb something with something	He replaced the pen with a pencil	0.0024
32	Somebody verb INFINITIVE	He dares to come	0.0002
33	Somebody verb VERB-ing	He enjoys swimming	0.0026
34	It verb that CLAUSE	It requires that you trust her	0.0014
35	Something verb INFINITIVE	This will help to prevent accidents	0.0002

Table 3.2 WordNet frames

non-clausal objects), which are not captured in the clause type. Given the DP of a clause, Werdy identifies the set of WN frames that can potentially match the clause as outlined in the flowchart of Fig. 3.2. Werdy walks through the flowchart; for each question, we check for the presence or absence of a specific constituent of a clause (e.g., a direct object for Q_1) and proceed appropriately until we obtain a set of possible frames. This set is further reduced by considering additional information in the frames (not shown; e.g., that the verb must end on "-ing"). For our example clause "The student remained in Princeton", we first identify possible frames $\{1,2,12,13,22,27\}$ (see 3.2) using the flowchart (Q_1 no, Q_2 no, Q_3 yes); using the additional information in the frames, Werdy then further prunes this set to $\{1,2,22\}$. The corresponding set of remaining candidate sense for *remain* is as given above, i.e., $\{\langle remain\text{-}2\rangle, \langle remain\text{-}3\rangle, \langle remain\text{-}4\rangle\}$.

Our mapping of clause types to WordNet frames is judiciously designed for the way WordNet is organized. For instance, frames containing adverbials generally do not specify whether or not the adverbial is obligatory; here we are conservative in that we do not prune such frames if the input clause does not contain an adverbial. As another example, some frames overlap or subsume each other; e.g, frame "somebody verb something" (8) subsumes "somebody verb that clause" (26). In some word senses annotated with the more general frame, the more specific one can also apply (e.g., $\langle$*point out*-1$\rangle$ is annotated with 8 but not 26; 26 can apply), in others it does not (e.g., $\langle$*play*-1$\rangle$ is also annotated with 8 but not 26; but here 26 cannot apply). To ensure the effectiveness of syntactic pruning, we only consider the frames that are directly specified in WordNet. This procedure often produces the desired results; in a few cases, however, we do prune the correct sense (e.g., frame 26 for clause "He points out that . . . ").

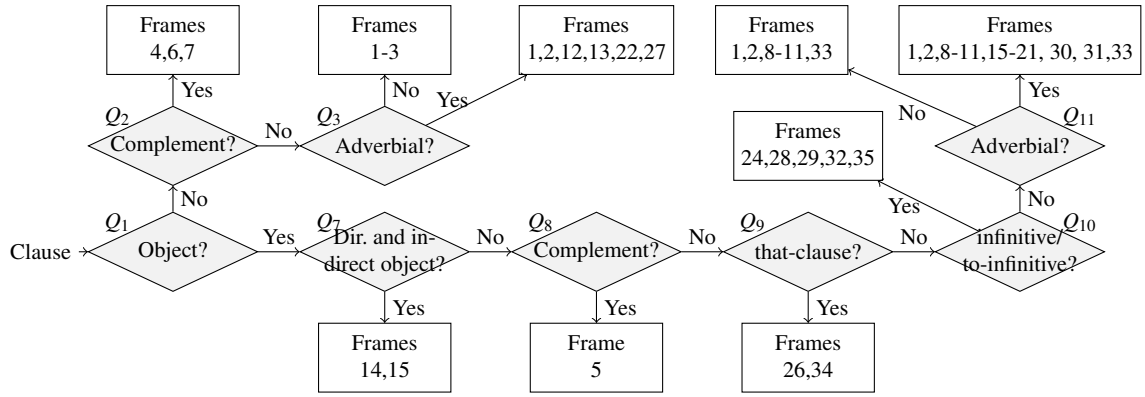


Figure 3.2 Flow chart for frame detection

3.5 Semantic Pruning

A verb sense imposes a restriction on the semantic type of the arguments it may take and vice versa (Quirk et al., 1985; Levin, 1993; Hanks, 1996; Baker et al., 1998; Palmer et al., 2005; Kipper et al., 2008). This allows us to further prune the verb candidate set by discarding verb senses whose semantic argument is not present in the clause.

WordNet frames potentially allow a shallow type pruning based on the semantics provided for the clause constituents. However we could solely distinguish people ("somebody") from things ("something"), which is too crude to obtain substantial pruning effects. Moreover, this distinction is sometimes ambiguous.

Instead, we have developed a more powerful approach to semantic pruning based on our VOS repository. We remove from the verb candidate set those senses whose semantic argument cannot be present in the sentence. For instance, consider the clause "*The man plays football*." Suppose that we know that the verb entry *play* with sense $\langle play\text{-}1 \rangle$ ("participate in sports") takes an object of type $\langle sport\text{-}1 \rangle$; i.e., we have a tuple $\langle play\text{-}1, sport\text{-}1 \rangle$ in our repository. Then, we check whether any of the possible senses of *football*—(i) *sport* or (ii) *ball*—is of type $\langle sport\text{-}1 \rangle$. Here the first sense has the correct type (the second sense does not); thus we retain $\langle play\text{-}1 \rangle$ as a possible sense for the verb entry *play*. Next, suppose that we consider sense $\langle play\text{-}3 \rangle$ ("play on an instrument"), which according to our corpus takes $\langle instrument\text{-}6 \rangle$ as argument (i.e., there is a tuple $\langle play\text{-}3, instrument\text{-}6 \rangle$ in our VOS repository). Since none of the senses of *football* is of type $\langle instrument\text{-}6 \rangle$, we can safely drop $\langle play\text{-}3 \rangle$ from our candidate set. We perform this procedure for every verb sense in the candidate set.

Semantic pruning makes use of both VOS repository and the hypernym structure of the noun senses in WordNet. For each sentence, we obtain the possible senses of the direct-object argument of the verb. We then consider each candidate sense of the verb (e.g., $\langle play\text{-}1 \rangle$), and check whether any of its compatible object-argument senses (from our repository) is a hypernym of any of the possible senses of its actual object argument (in the sentence); e.g., $\langle sport\text{-}1 \rangle$ is a hypernym of $\langle football\text{-}1 \rangle$. If so, we retain the verb's candidate sense. If not, either the candidate sense of the verb is indeed incompatible with the object argument in the sentence, or our repository is incomplete. To handle incompleteness to some extent, we also consider hyponyms of the object-argument senses in our repository; e.g., if we observe object *sport* in a sentence and have verb-sense argument $\langle football\text{-}1 \rangle$ in our corpus, we consider this a match. If the hyponyms lead to a match, we retain the verb's candidate sense; otherwise, we discard it.

3.6 Verb-Object Sense Repository

We use three different methods to construct the repository. In particular, we harness the sense-annotated WordNet glosses[2] as well as the sense-annotated SemCor corpus (Landes et al., 1998).[3]

The major part of the VOS repository was acquired from WordNet's gloss tags, a corpus containing sense annotations for WordNet glosses. According to Atkins and Rundell (2008), noun definitions should be expressed in terms of the class to which they belong, and verb definitions should refer to the types of the subjects or objects related to the action. Based on this rationale, we extracted all noun senses that appear in the gloss of each verb sense; each of these noun senses is treated as a possible sense of the object argument of the corresponding verb sense. For example, the gloss of ⟨*play*-1⟩ is "participate in games or sports;" each noun is annotated with its senses (2 and 3 for "games", 1 for "sports"). We extract tuples ⟨*play*-1, *game*-2⟩, ⟨*play*-1, *game*-3⟩, and ⟨*play*-1, *sport*-1⟩ from this gloss. Note that we only extract direct-object arguments, i.e., we do not consider the type of the subject argument of a verb sense. Since the constituents of the predicate are much more important than the subject to determine or describe a verb sense, lexical resources rarely contain information on the subject (Atkins and Rundell, 2008). Similarly, WordNet glosses typically do not provide any information about adverbials. Overall, we collected arguments for 8,657 verb senses (out of WordNet's 13,767 verb senses) and a total of 13,050 ⟨*verb*-#, *object*-#⟩-pairs.

We leveraged the sense-annotated SemCor corpus to further extend our VOS repository. We parsed each sentence in the corpus to obtain the respective pairs of verb sense and object sense. Since sentences are often more specific than glosses, and thus less helpful for constructing our repository, we generalized the so-found object senses using a heuristic method. In particular, we first obtained all the object senses of each verb sense, and then repeatedly generalized sets of *at least two senses* that share a direct hypernym to this hypernym. The rationale is that we only want to generalize if we have some evidence that a more general sense may apply; we thus require at least two hyponyms before we generalize. For instance, ⟨*play*-1, *soccer*-1⟩ and ⟨*play*-1, *American football*-1⟩ is generalized to ⟨*play*-1, *football*-1⟩, which implies that the pair ⟨*play*-1, *rugby*-1⟩ is also now considered given that ⟨*rugby*-1⟩ is a hyponym of ⟨*football*-1⟩. Using this method, we collected arguments for 1,516 verb senses and a total of 4,131 sense pairs.

Finally, we noticed that the most frequent senses used in the English language are usually so general that their glosses do not contain any relevant semantic argument. For instance, one of the most frequent verbs is ⟨*see*-1⟩, which has gloss "perceive by ⟨*sight*-3⟩". The

[2] http://wordnet.princeton.edu/glosstag.shtml
[3] http://web.eecs.umich.edu/~mihalcea/downloads.html

correct semantic argument ⟨*entity*-1⟩ is so general that it is omitted from the gloss. In fact, our gloss-tag extractor generates tuple ⟨*see*-1, *sight*-3⟩, which is incorrect. We thus manually annotated the 30 most frequent verb senses with their object argument types.

Our final repository contains arguments for 9,335 verb senses and a total of 17,181 pairs. Pairs from SemCor tend to be more specific because they refer to text occurrences. The assumption of taking the nouns of the glosses as arguments seems to be mostly correct, although some errors may be introduced. Consider the pair ⟨*play*-28, *stream*-2⟩ extracted from the gloss "discharge or direct or be discharged or directed as if in a continuous ⟨*stream*-2⟩". Also, in some cases, the glosses may refer to adverbials as in ⟨*play*-14, *location*-1⟩, taken from gloss "perform on a certain ⟨*location*-1⟩". Note that if an argument is missing from our repository, we may prune the correct sense of the verb. If, however, there is an additional, incorrect argument in the repository, the correct verb sense is retained but pruning may be less effective.

3.7 Evaluation

Dataset. We tested Werdy on the SemEval-2007 coarse-grained dataset.[4] It consists of five sense-annotated documents; the sense annotations refer to a coarse-grained version of WordNet. In addition to sense annotations, the corpus also provides the corresponding KB entries (henceforth termed "gold entries") as well as a POS tag. We restrict our evaluation to verbs that act as clause heads. In total, 461 such verbs were recognized by ClausIE (chapter 2) and the Stanford Parser (Klein and Manning, 2003).[5]

WSD Algorithms. For the final step of Werdy, we used the KB-based WSD algorithms of Ponzetto and Navigli (2010) and It-Makes-Sense (Zhong and Ng, 2010), a state-of-the-art supervised system that was the best performer in SemEval-2007. Each method only labels entries for which it is sufficiently confident.

Simplified Extended Lesk (SimpleExtLesk). A version of Lesk (1986). Each entry is assigned the sense with highest term overlap between the entry's context (words in the sentence) and both the sense's gloss (Kilgarriff and Rosenzweig, 2000) as well as the glosses of its neighbors (Banerjee and Pedersen, 2003). A sense is output only if the overlap exceeds some threshold; we used thresholds in the range of 1–20 in our experiments and selected the best performing (see Tab. 3.3). There are many subtleties and details in the

[4]The data is annotated with WordNet 2.1 senses; we converted the annotations to WordNet-3.0 using DKPro-WSD (Miller et al., 2013).

[5]Version 3.3.1, model englishRNN.ser.gz

implementation of SimpleExtLesk so we used two different libraries: a Java implementation of WordNet::Similarity (Pedersen et al., 2004),[6] which we modified to accept a context string, and DKPro-WSD (Miller et al., 2013) version 1.1.0, with lemmatization, removal of stop words, paired overlap enabled and normalization disabled.

Degree Centrality. Proposed by Navigli and Lapata (2010). The method collects all paths connecting each candidate sense of an entry to the set of candidate senses of the words the entry's context. The candidate sense with the highest degree in the resulting subgraph is selected. We implemented this algorithm using the Neo4j library.[7] We used a fixed threshold of 1 and vary the search depth in range 1–20 to search for the best performing (see Tab. 3.3). We used the candidate senses of all nouns and verbs in a sentence as context.

It-Makes-Sense (IMS). A state-of-the-art, publicly available supervised system Zhong and Ng (2010) and a refined version of (Chan et al., 2007), which ranked first in the SemEval-2007 coarse grained task. We modified the code to accept KB entries and their candidate senses. We tested both in WordNet-2.1 and 3.0; for the later we mapped Werdy's set of candidates to WordNet-2.1.

Most Frequent Sense (MFS). Selects the most frequent sense (according to WordNet frequencies) among the set of candidate senses of an entry. If there is a tie, we do not label. Note that this procedure differs slightly from the standard of picking the entry with the smallest sense id. We do not follow this approach because it cannot handle overlapping entries: if we have overlapping entries we will have ids which are not related to each other given that a set of ids is only defined for a single entry.

MFS back-off. When one of the above methods fails to provide a sense label (or provides more than one), we used the MFS method above with a threshold of 1. This procedure increased the performance in all cases.

Methodology. The disambiguation was performed with respect to coarse-grained sense clusters. The score of a cluster is the sum of the individual scores of its senses (except for IMS which provides only one answer per word); the cluster with the highest score was selected. Our source code and the results of our evaluation are publicly available[8].

The SemEval-2007 task was not designed for automatic entry recognition: for each word or multi-word expression it provides the WordNet entry and the POS tag. We proceeded as follows to handle multi-word entries. In the WSD step, we considered the candidate senses of all recognized entries that overlap with the gold entry. For example, we considered the candidate senses of entries *take*, *breath*, and *take a breath* for gold entry *take a breath*.

[6] http://www.sussex.ac.uk/Users/drh21/
[7] http://www.neo4j.org/
[8] http://people.mpi-inf.mpg.de/~corrogg/

The SemEval-2007 task uses WordNet-2.1 but Werdy uses WordNet-3.0. We mapped both the sense keys and clusters from WordNet-2.1 to WordNet-3.0 using DKPro. All senses in WordNet-3.0 that could not be mapped to any cluster were considered to belong to a single sense cluster each. Note that this procedure is fair: for such senses, the disambiguation is equivalent to a fine-grained disambiguation, which is harder.

Results. Our results are displayed in Tab. 3.3. We ran each algorithm with the gold KB entries provided by in the dataset (+ in column "gold entry) as well as the entries obtained by our method of Sec. 3.3 (-). We also enabled (+) and disabled (-) the pruning steps as well as the MFS back-off strategy. The highest F1 score was achieved by SimpleExtLesk (DKPro) with pruning and MFS back-off: 81.18 with gold entries and 78.52 with automatic entry recognition. In all cases, our syntactic and semantic pruning strategy increased performance (up to +10.85 F1 points). We next discuss the impact of the various steps of Werdy in detail.

Detailed Analysis. Tab. 3.4 displays step-by-step results for DKPro's SimpleExtLesk, for MFS, as well as SimpleExtLesk with MFS back-off, the best performing strategy. The table shows results when only some Werdy's steps are used. We start from a direct use of the respective algorithm with the gold entries of SemEval-2007 after each horizontal line, and then successively add the Werdy steps indicated in the table.

When no gold entries were provided, performance dropped due to the increase of sense candidates for multi-word expressions, which include the possible senses of the expression itself as well as the senses of the entry's parts that are themselves WordNet entries. Our entry recognizer tends to do a good job since it managed to correctly identify all the relevant entries except in two cases (i.e. "take up" and "get rolling"), in which the dependency parse was incorrect. The drop in F1 for our automatic entry recognition was mainly due to incorrect selection of the correct entry of a set of alternative, overlapping entries.

Syntactic pruning did not prune the correct sense in most cases. In 16 cases (with gold entries), however, the correct sense was pruned. Five of these senses were pruned due to incorrect dependency parses, which led to incorrect frame identification. In two cases, the sense was not annotated with the recognized frame in WordNet, although it seemed adequate. In the remaining cases, a general frame from WordNet was incorrectly omitted. Improvements to WordNet's frame annotations may thus make syntactic pruning even more effective.

Semantic pruning also improves performance. Here the correct sense was pruned for 11 verbs, mainly due to the noisiness and incompleteness of our VOS repository. Without using gold entries, we found in total 237 semantic matches between possible verbs senses and

possible object senses (200 with gold entries). We also found that our manual annotations in the VOS repository (see Sec. 3.6) did not affect our experiments.

The results show that syntactic and semantic pruning are beneficial for verb sense disambiguation, but also stress the necessity to improve existing resources. Ideally, each verb sense would be annotated with both the possible clause types or syntactic patterns in which it can occur as well as the possible senses of its objects. Annotations for subjects and adverbial arguments may also be beneficial.

3.8 Related Work

WSD is a classification task where for every word there is a set of possible senses given by some external resource (as a KB). Two types of methods can be distinguished in WSD: supervised and KB-based. A comprehensive overview of WSD systems can be found in Navigli (2009) and Navigli (2012). An overview of related work is displayed on Tab. 3.5.

Supervised systems Supervised systems (Dang and Palmer, 2005; Dligach and Palmer, 2008; Chen and Palmer, 2009; Zhong and Ng, 2010) train a classifier to assign senses to words, mostly relying on manually annotated data for training. Zhong and Ng (2010) is one of the best performing methods to date. It uses a linear classifier for each word type appearing in the training data. The system is trained on SemCor (Landes et al., 1998), DSO (Ng and Lee, 1997) and parallel data. As features it uses the POS tags of the surrounding words (3 in each direction), the surrounding words themselves (without stopping words) and a set of 11 collocations, which are ordered sequences of the surrounding words.

In principle, supervised systems suffer from low coverage since the training data is usually sparse. Some authors have tried to overcome this limitation by exploiting linked resources (like Wikipedia) as training data (Shen et al., 2013; Cholakov et al., 2014). Shen et al. (2013) generates a repository of senses in which each sense is a Wikipedia article, and the anchor text of the entities linked to the articles is used as contextual data for each sense. Cholakov et al. (2014) annotates text corpora automatically with verb senses via a pattern-based representation of the senses. The method first identifies representative patterns for verb senses in UBY (Gurevych et al., 2012), an integrated of several lexical-semantic resources such as WordNet, Wikipedia and FrameNet. The pattern representing each verb context is both syntactic (via POS tags) and semantic (via types like person or location). For instance, the sentence “But an insider told TODAY : ‘ There was no animosity.’ ” can be generalized as “*person tell location be feeling*”. Once the patterns for each verb sense have been identified, the authors label verb senses in a big text corpora by looking at occurrences

of those patterns. This way of constructing the patterns captures somehow the syntactic and semantic concordance between the verb and its arguments which we also exploit in Werdy in a more direct way.

KB-based methods The second WSD approach corresponds to the so-called KB methods (Agirre and Soroa, 2009; Navigli and Lapata, 2010; Ponzetto and Navigli, 2010; Miller et al., 2012; Agirre et al., 2014; Moro et al., 2014). They rely on a background KB (typically WordNet or extended versions (Navigli and Ponzetto, 2012)), where related senses appear close to each other. KB-based algorithms often differ in the way the KB is explored. Agirre and Soroa (2009) uses Personalized PageRank to explore WordNet and detect the most meaningful set of word senses in a sentence. It has been shown, however, that a key point to enhance performance is to include more semantic information into the KB (Ponzetto and Navigli, 2010; Miller et al., 2012). Ponzetto and Navigli (2010) extends WordNet by linking Wikipedia articles to generate a much richer semantic network. Miller et al. (2012) increases the contextual information of the sentence by including words which co-occur with the word context in a large corpus. Our framework fits this line of work in the sense that it also KB-based and it enriches the background knowledge in order to enhance performance of standard WSD algorithms.

Verb-sense disambiguation The disambiguation of nouns has received most of the attention in the WSD field, achieving significant results in both supervised and KB-based approaches. Zhong and Ng (2010) is a supervised system which was the best performing method in the Semeval-2007 coarse grained WSD task. It achieved more than 82,3% F1 in a coarse-grained setting for the disambiguation of nouns. Similar numbers (85,5%) have been achieved by the KB-based approach by Ponzetto and Navigli (2010). However, verb sense disambiguation is still an open issue. It is a more difficult task since the polysemy of verbs is bigger that the polysemy of nouns (2.17 vs. 1.24). Even more, the distribution of verbs use is, according to WordNet statistics, more skewed than the use of nouns. Some verb senses are so frequent that makes them difficult to distinguish by context; i.e. a veb sense may be so generic that it fits in almost every context. However, as stated before, understanding verbs is important for automatic text understanding: verbs are the key to understand the relation of the extraction and to structure the information appropriately. Dang and Palmer (2005) was a pioneering work on verb sense disambiguation. They use a classifier to disambiguate verbs in which the main feature are the semantic roles of the verb arguments in the clause.

Linguists have noted the link between verb senses and the syntactic structure and argument types (Quirk et al., 1985; Levin, 1993; Hanks, 1996), and supervised WSD systems were

developed to capture this relation (Dang and Palmer, 2005; Chen and Palmer, 2009; Dligach and Palmer, 2008; Cholakov et al., 2014). In Dang and Palmer (2005) and Chen and Palmer (2009), it is shown that WSD tasks can be improved with features that capture the syntactic structure and information about verb arguments and their types. They use features such as shallow named entity recognition and the hypernyms of the possible senses of the noun arguments. Dang and Palmer (2005) also included features extracted from PropBank (Palmer et al., 2005) from role labels and frames. Dligach and Palmer (2008) generated a corpus of verb and their arguments (both surface forms), which was used to incorporate a semantic feature to the supervised system.

In our work, we also incorporate syntactic and semantic information. Instead of learning the relation between the verb senses and the syntactic structure, however we incorporate it explicitly using the WordNet frames, which provide information about which verb sense should be consider for a given syntactic pattern. We also incorporate explicitly the semantic relation between each verb sense and its arguments using our VOS repository.

Word entry recognition To bring WSD to real-world applications, the mapping between text and KB entries is a fundamental first step. It has been argued that the existence of multi-word expressions imposes multiple challenges to text understanding tasks (Sag et al., 2002). The problem has been addressed by Arranz et al. (2005) and Finlayson and Kulkarni (2011). They find multi-word entries by matching word sequences allowing some morphological and POS variations according to a predefined set of patterns. Our method differs in that we can recognize words and multi-word expressions in a principled way by exploiting the syntactic structure of the sentence., in that we can discover KB entries that appear discontinuously and in that we do not select the correct entry but generate a set of potential entries. The selection of the entry is jointly performed with the disambiguation.

Resources Different resources of semantic arguments for automatic text understanding tasks have been constructed (Baker et al., 1998; Palmer et al., 2005; Kipper et al., 2008; Gurevych et al., 2012; Nakashole et al., 2012; Flati and Navigli, 2013). In Baker et al. (1998), Palmer et al. (2005) , Kipper et al. (2008) and Gurevych et al. (2012), the classification of verbs and arguments is focused toward semantic or thematic roles. Nakashole et al. (2012) uses semantic types to construct a taxonomy of binary relations and Flati and Navigli (2013) collected semantic arguments for given textual expressions. For instance, given the verb "break", they extract a pattern "break ⟨*body part*-1⟩". In contrast to existing resources, our VOS repository disambiguates both the verb sense and the senses of its arguments.

3.9 Conclusion

We presented Werdy, an unsupervised framework for word-sense recognition and disambiguation with a particular focus on verbs and verbal phrases. Our main contributions rely on a principled way to recognize words and multi-word expressions in natural language text and in a set of linguistic-based techniques to reduce the candidate set of senses for a given verb. Our results indicate that incorporating syntactic and semantic constraints improves the performance of verb sense disambiguation methods. This stresses the necessity of extending and improving the available syntactic and semantic resources, such as WordNet or our VOS repository.

Algorithm	Gold Entry	Pruning	MFS back-off	threshold /depth	Verbs (clause heads) P	R	F1	F1 points
Degree	+	-	+	5	73.54	73.54	73.54	
Centrality	+	+	+	11	79.61	79.61	79.61	+ 6.07
	+	-	-	5	73.99	71.58	72.77	
	+	+	-	8	79.91	78.52	79.21	+ 6.44
	-	-	+	5	70.41	70.41	70.41	
	-	+	+	10	76.46	76.46	76.46	+ 6.05
	-	-	-	4	71.05	68.90	69.96	
	-	+	-	10	76.81	75.81	76.30	+ 6.34
SimpleExtLesk	+	-	+	6	77.28	75.27	76.26	
(DKPro)	+	+	+	5	81.90	80.48	**81.18**	+ 4.92
	+	-	-	1	73.70	52.28	61.17	
	+	+	-	1	81.99	64.21	72.02	**+ 10.85**
	-	-	+	5	74.33	72.57	73.44	
	-	+	+	5	79.30	77.75	**78.52**	+ 5.08
	-	-	-	1	69.85	50.54	58.65	
	-	+	-	1	78.69	62.20	69.48	+ 10.83
SimpleExtLesk	+	-	+	5	77.11	75.27	76.18	
(WordNet::Sim)	+	+	+	5	80.57	79.18	79.87	+ 3.69
	+	-	-	1	74.82	68.98	71.78	
	+	+	-	1	79.04	75.27	77.11	+ 5.33
	-	-	+	6	74.12	72.35	73.22	
	-	+	+	7	77.97	76.46	77.21	+ 3.99
	-	-	-	1	71.36	65.66	68.39	
	-	+	-	1	76.20	71.92	74.00	+ 5.61
MFS	+	-	-	1	76.61	74.62	75.60	
	+	+	-	1	80.35	78.96	79.65	+ 4.05
	-	-	-	1	73.67	71.92	72.79	
	-	+	-	1	77.75	76.24	76.99	+ 4.20
IMS	+	-	+	n.a.	79.60	79.60	79.60	
(WordNet-2.1)	+	+	+	n.a.	80.04	80.04	80.04	+ 0.44
	-	-	+	n.a.	76.21	75.05	75.63	
	-	+	+	n.a.	77.53	76.36	76.94	+ 1.31
IMS	+	-	+	n.a.	78.96	78.96	78.96	
(WordNet-3.0)	+	+	+	n.a.	79.83	79.83	79.83	+ 0.87
	-	-	+	n.a.	75.77	74.62	75.19	
	-	+	+	n.a.	77.53	76.36	76.94	+ 1.75

Table 3.3 Results on SemEval-2007 coarse-grained (verbs as clause heads)

Steps Performed	threshold	P	R	F1	F1 points
SimpleExtLesk (DKPro)					
Plain with gold entries	**1**	**73.70**	**52.28**	**61.17**	
+ Entry Recognition	1	69.85	50.54	58.65	- 2.52
+ Syntactic Pruning	1	76.47	58.84	66.50	**+ 7.85**
+ Semantic Pruning	1	78.69	62.20	69.48	+ 2.98
+ Entry Recognition	1	69.85	50.54	58.65	- 2.52
+ Semantic Pruning	1	73.85	55.39	63.30	+ 4.65
+ Syntactic Pruning	1	79.33	61.21	69.10	**+ 7.93**
+ Semantic Pruning	1	81.99	64.21	72.02	+ 2.92
+ Semantic Pruning	1	78.11	56.90	65.84	+ 4.67
MFS					
Plain with gold entries	1	76.61	74.62	75.60	
+ Entry Recognition	1	73.67	71.92	72.79	- 2.81
+ Syntactic Pruning	1	75.77	74.14	74.95	+ 2.16
+ Semantic Pruning	1	77.75	76.24	76.99	+ 2.04
+ Entry Recognition	1	73.67	71.92	72.79	- 2.81
+ Semantic Pruning	1	77.09	75.43	76.25	**+ 3.46**
+ Syntactic Pruning	1	78.46	76.94	77.69	+ 2.09
+ Semantic Pruning	1	80.35	78.96	79.65	+ 1.96
+ Semantic Pruning	1	79.91	78.02	78.95	**+ 3.35**
SimpleExtLesk (DKPro) with MFS back-off					
Plain with gold entries	6	77.28	75.27	76.26	
+ Entry Recognition	6	74.33	72.57	73.44	- 2.82
+ Syntactic Pruning	5	76.65	75.00	75.82	+ 2.38
+ Semantic Pruning	5	79.30	77.75	78.52	+ 2.70
+ Entry Recognition	5	74.33	72.57	73.44	- 2.82
+ Semantic Pruning	5	78.19	76.51	77.34	**+3.90**
+ Syntactic Pruning	5	79.34	77.80	78.56	+ 2.30
+ Semantic Pruning	5	81.90	80.48	81.18	+ 2.62
+ Semantic Pruning	5	81.02	79.09	80.04	**+ 3.78**

Table 3.4 Step-by-step results

WSD	Supervised	Manual Annotated	Dang and Palmer (2005); Dligach and Palmer (2008); Chen and Palmer (2009);Zhong and Ng (2010)
		Linked Data	Shen et al. (2013); Cholakov et al. (2014)
	KB-based	Pure KB	Agirre and Soroa (2009)
		Enriched resources	Ponzetto and Navigli (2010); Miller et al. (2012) Del Corro et al. (2014); Moro et al. (2014)

Table 3.5 WSD related work map and (mostly recent) example citations

Chapter 4

FINET: Context-Aware Fine-Grained Named Entity Typing

4.1 Introduction

Named entity typing (NET) is the task of detecting the type(s) of a named entity in a given context. For instance, given the sentence "John plays guitar on the stage", our goal is to infer that "John" is a *guitarist*, a *musician* and a *person*. This work proposes FINET, a system for detecting the types of named entities that occur in short inputs—such as sentences or tweets—with respect to WordNet's super fine-grained type system (16k types of organizations, persons and locations).

Named entity typing is a fundamental building block for many natural-language processing tasks. NET is at the heart of information extraction methods for finding types for entities in a knowledge base[1] (KB) from natural-language text (Mitchell et al., 2015). Likewise, NET aids named entity disambiguation by reducing the space of candidates for a given entity mention. Entity types are an important resource for entity-based retrieval or aggregation tasks, such as semantic search (Hoffart et al., 2014) or question answering (Yahya et al., 2013). Finally, type information helps to increase the semantic content of syntactic patterns (Nakashole et al., 2012) or the extractions from open information extraction (Lin et al., 2012).

The extraction of explicit types has been studied in the literature, most prominently in the context of taxonomy induction (Snow et al., 2006). Explicit types occur, for example, in phrases such as "Steinmeier, the German Foreign Minister, [...]", "Foreign Minister Steinmeier", or "Steinmeier is the German Foreign Minister." These explicit types are

[1] In this chapter, we refer to WordNet as a type system and to a collection of entities and their types as a KB.

often extracted via patterns, such as the well-known Hearst patterns (Hearst, 1992), and subsequently integrated into a taxonomy. Pattern-based methods often have high precision but low recall: Types are usually mentioned when a named entity is introduced or expected to be unknown to readers, but often are not explicitly stated. The NET problem differs from taxonomy induction in that (1) the type system is prespecified, (2) types are disambiguated, and (3) types are associated with each occurrence of named entity in context.

Our FINET system makes use of explicit type extractions whenever possible. But even when types are not explicitly mentioned, sentences may give clues to the correct type. These clues can range from almost explicit to highly implicit. For example, in "Messi plays soccer", the type *soccer player* is almost explicit. The sentence "Pavano never even made it to the mound," however, only implicitly indicates that "Pavano" is a *baseball player*. A key challenge in NET is to extract such implicit, context-aware types to improve recall.

One way to extract implicit types is to train a supervised extractor on labeled training data, in which each entity is annotated with a set of appropriate types. The key problem of this approach is that labeled training data is scarce; this scarcity is amplified for fine-grained type systems. To address this problem, many existing systems generate training data by exploiting KBs as a resource of entities and their types (Yosef et al., 2012). A popular approach is to train an extractor on a corpus of sentences (e.g., on Wikipedia), in which each named entity has been associated with all its types known to the KB. The key problem with such an approach is that the so-obtained type information is oblivious to the context in which the entity was mentioned. For example, in both sentences "Klitschko is known for his powerful punches" and "Klitschko is the Mayor of Kiew," "Klitschko" will be associated with all its types, e.g., *boxer*, *politician* and *mayor*. As a consequence, the labels in the training data can be misleading and may negatively affect both precision and recall of the learned extractors. Moreover, such extractors are often biased towards prominent types but perform poorly on infrequent types, and they are generally problematic when types are correlated (e.g., most *presidents* are also *graduates* and *authors*).

FINET addresses the above problems by first generating a set of type candidates using multiple different extractors and then selecting the most appropriate type(s). To generate candidates, we make use of a sequence of extractors that range from explicit to highly implicit type extractors. Implicit extractors are only used when more explicit type extractors fail to produce a good type. Our extractors are based on patterns, mention text, and verbal phrases. To additionally extract highly implicit types for a named entity, FINET makes use of word vectors (Mikolov et al., 2013) trained on a large unlabeled corpus to determine the types of *similar* entities that appear in a *similar* context. This extractor is comparable to the KB

methods discussed above, but is unsupervised, and takes as candidates the types frequent within the related entities and contexts.

After type candidates have been generated, the final step of FINET selects the subset of appropriate types that fit the context. We leverage previous work on word sense disambiguation (WSD) in this step, as well as resources such as WordNet glosses, WordNet example sentences, and, if available, manually annotated training data.

FINET leverages ideas from state-of-the-art systems and extends them by (1) handling short inputs such as sentences or tweets (2) supporting a very fine-grained type hierarchy, and (3) producing types that match the context of the entity mention. Most existing systems are unable to extract more than a couple of hundred different types. For example, Hyena (Yosef et al., 2012), the system with the most fine-grained type system so far, focuses on a KB frequent set of 505 types from WordNet. Hyena lacks important types such as *president* or *businessman*, and includes *soccer player* but not *tennis player*. Instead of restricting types, FINET operates on the the entire set of types provided by WordNet, a popular, fine-grained type system with more than 16k types for persons, organizations, and locations.

We evaluated FINET on a number of real-world datasets. Our results indicate that FINET significantly outperforms previous methods.

4.2 Candidate Generation

In the candidate generation phase, we collect possible types for each entity mention. We start with preprocessing the input and subsequently apply a (i) pattern-based extractor, (ii) a mention-based extractor, (iii) a verb-based extractor, and (iv) a corpus-based extractor. The extractors are ordered by decreasing degree of explicitness of their extracted types.

Each extractor has a *stopping condition*, which we check whenever the extractor produced at least one type. When the stopping condition is met, we directly proceed to the type selection phase. The reasoning behind this approach is to bias FINET towards the most explicit types, i.e., when an explicit type is found, the stopping condition generally fires. Otherwise, when the stopping condition is not met, we enrich the set of candidate types of the extractor with their hypernyms. In this case, we expect types to be overly specific so that we want to allow the selection phase to be able to select a more general type. We also run subsequent extractors when the stopping condition is not met. Tab. 4.1 displays a summary of the extractors and their corresponding stopping conditions.

In what follows, we discuss preprocessing as well as each extractor and its corresponding stopping condition. All so-found type candidates are passed to the candidate selection phase, which we discuss in Sec. 4.3.

Extractor	Stopping Condition
Pattern-based (final)	Always stop
Pattern-based (non-final)	KB-lookup
Mention-based	KB-lookup
Verb-based	KB-lookup
Corpus-based	$\geqslant$ 50% of score in most frequent $\leqslant$ 10 types

Table 4.1 Extractors and their stopping conditions

4.2.1 Preprocessing

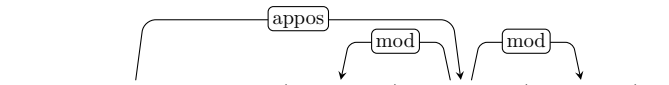
The preprocessing phase consists of 5 steps: (i) dependency parsing (Socher et al., 2013); (ii) co-reference resolution (Recasens et al., 2013); (iii) named entity recognition (NER) (Finkel et al., 2005), including the detection of coarse-grained types (i.e., *person*, *organization*, *location*); (iv) clause identification (chapter 2); and (v) word and multi-word expression recognition (chapter 3).

FINET restricts its candidate set to the hyponmys of the coarse-grained type of the named entity recognizer. Named entities with the same coarse-grained type occurring in a coordinating relation (e.g., "Messi and Ronaldo are soccer players") are linked so that they share the candidate set. Similarly, identical mentions share their candidate set (which is reasonable in short inputs).

FINET extractors operate either on the sentence or the clause level; see the next sections. A clause is a part of a sentence that expresses some statement or coherent piece of information and is thus a suitable units for automatic text processing tasks (see chapter 2). Finally, we identify multi-word explicit type mentions such as *Prime Minister* or *Secretary of Housing and Urban Development* (see chapter 3).

4.2.2 Pattern-based extractor

Our pattern-based extractor targets explicit type mentions. Explicit type mentions are commonly used to introduce entities when they first appear in text ("US President Barack Obama") or when their mention does not refer to the most prominent entity ("Barack Obama, father of the US President"). Following previous work (Hearst, 1992), we make use of a set of patterns to look for words or expressions that may refer to the type of a given named entity. We refer to the so-found expressions as *lexical types* (e.g., "father"). Once lexical types have



(a) A syntactic pattern

NAMED_ENTITY , NOUN

(b) A regular expression pattern

Figure 4.1 Patterns capturing appositions

been identified, we collect as candidate types the *types* i.e., WordNet synsets–to which the lexical type may refer (e.g., ⟨*father*-1⟩, . . . , ⟨*father*-8⟩, the eight senses of "father").

Our extractor makes use of two types of patterns: syntactic patterns, which operate on the dependency parse, and regular expression patterns, which operate on the input text. Syntactic patterns are preferable in that they do not rely on continuous chunks of text and can skip non-relevant information. However, mistakes in the dependency parse may lower recall. To cope with these potential mistakes, we additionally include regular expressions for some syntactic patterns. Fig. 4.1 shows an example of a syntactic pattern and a related regular-expression pattern. Both patterns produce lexical type "president" from "Barack Obama, president of the US," but only the syntactic pattern applies to "Barack Obama, the current US president."

Tab. 4.2 gives an overview of all our patterns. Most of the patterns also have a symmetric version (e.g., "The president, Barack Obama" and "Barack Obama, the president"), which is not displayed. We divide our patterns into final and non-final patterns. Final patterns generally have high precision and extract the lexical type exactly as it occurs in the text. When a final pattern produces a lexical type, we add the corresponding types to the candidate set and go directly to the type selection phase, i.e., we do not consider any other extractor. For non-final patterns, however, we expect erroneous extractions and thus proceed differently. In more detail, we perform a KB lookup for all so-found lexical types. The KB lookup, which we describe in detail in the next section, both prunes and expands the candidate set using a KB, and acts as a stopping condition, i.e., it decides whether to move to the next extractor or directly to the type selection phase. FINET can also be run without using KB lookups; see below.

We treat a pattern as non-final if it may or may not denote a lexical type (e.g., "the president of Argentina" vs. "the city of Buenos Aires") or if a transformation between verbs and nouns is required to obtain the lexical type (e.g. "Shakespeare's productions" to "producer"). To perform transformations, we make use of WordNet's *derivationally related forms*, which connect semantically related morphological variations of verbs and nouns. For

Pattern	Example
Final patterns	
Hearst I	{Presidents} such as [Obama] (and) [Bush]
Hearst II	{Presidents} like [Obama] (and) [Bush]
Hearst III	Obama (and) other {presidents}
Hearst IV	{Presidents} including [Obama] (and) [Bush]
Apposition	[Obama], (the) {president}
Copular	[Obama] is (the) {president}
Noun modifier	{President} [Barack Obama]
Among	[Joe Biden] among (other) {vice presidents}
Enough	[Messi] is enough (of) a {player}
As	[Messi] as {player}
Non-final patterns	
Location	{City} of [London]
Poss. + transf.	[Shakespeare]'s {productions}
by-prep + transf.	{productions} by [Shakespeare]

Table 4.2 Patterns for explicit type extraction

instance, the noun "production" is connected to the verb "produce," which in turn is connected to the noun "producer". These variations can be exploited for explicit type extractions; see Tab. 4.2. For example, we obtain the lexical type "producer" from "Shakespeare's productions". We treat such transformations as non-final because we may make mistakes in the path. Moreover, WordNet is highly incomplete in terms of derivational forms so that we may not be able to reach all the possible senses for "producer". Finally, morphological variations are insufficient in some cases. For instance, in "works by Schumann", we cannot reach "musician" or "artist" from "work", but we reach "worker" and there is no synset of "worker" that specifically denotes an artist.

4.2.3 Exploiting a knowledge base

Most of our extractors (optionally) leverage a knowledge base to (1) prune candidate types, (2) find additional candidate types, and (3) decide whether or not to consider subsequent extractors. To do so, we extract from a KB a repository of (entity mention, type)-pairs.[2] We use the KB conservatively, i.e., we consider KB evidence only if the extracted types match the ones in the KB.

[2]We used the Yago2 KB (Hoffart et al., 2013) in our experiments. Our repository contained roughly 9M different entity mentions and 20M (mention, type)-pairs.

The KB is leveraged via a *KB lookup*. Each KB lookup takes as input an entity mention e and a set T of candidate types found by the extractor (lexical or disambiguated). We first replace each lexical type in T by the set of types from WordNet that can refer to the lexical type (see previous section). Afterwards, for each type $t \in T$ separately, we check whether there is one or more matching types (e, t_{KB}) in the KB for the given mention. Type t_{KB} matches t if it is either identical, a hypernym, or a hyponym of t. For each match, the KB lookup outputs t. If t_{KB} is a hyponym of t, we additionally output t_{KB}, that is, a type more specific than the one found by the extractor. We leave the decision of whether t or t_{KB} is a more suitable type to the type selection phase. For example, for $e =$ "Messi" and $t = \langle$*player*-1$\rangle$, we output $\langle$*player*-1$\rangle$ and $\langle$*soccer_player*-1$\rangle$ (a hyponym) with our KB.

The KB lookup is *successful* if it outputs at least one type. Whenever an extractor performs a successful KB lookup, we add the resulting types to the candidate set and directly go to the type selection phase.

A KB lookup *fails* if no matching type was found in the KB, i.e., the KB does not provide sufficient information to guide type extraction. We then proceed differently: if a KB lookup fails, we add the complete set T (with lexical types replaced by types) to the candidate set and continue to the next extractor, i.e., we do not stop looking for additional candidate types.

As mentioned above, FINET can also be run without performing any KB lookups; the corresponding extractors then do not have a stopping condition. In our experimental study (Sec. 4.4), we experimented with both variants and found that KB lookups generally help.

4.2.4 Mention-based extractor

Our second extractor aims to extract type candidates from the entity mention itself. This approach is particularly effective for organizations, which often contain the lexical type in their name. Examples include "Johnson & Wales University," "Republican House," or "Massachusetts General Hospital".

Given an entity mention, we check if any of the words or expressions embedded in the name corresponds to a lexical type in WordNet. If so, we consider the corresponding types as potential candidates. For instance, for "Imperial College London", we extract lexical type "college" and obtain types $\langle$*college*-1$\rangle$ and $\langle$*college*-2$\rangle$ (as before, we consider only types matching the coarse-grained type) from WordNet. Similar to our handling of non-final patterns in the pattern-based extractor, we subsequently perform a KB lookup.

We extend the above procedure for entities tagged as *location*, because the set of (named-entity) locations is quite static and known. In more detail, we assume that the KB contains all locations and all their possible types. (Our experiments strengthened this assumption.) If a mention of a location (e.g., "Berlin") occurs in the repository and the above procedure did

not produce any candidates, we instead add all the corresponding types from the repository to the candidate set (e.g., ⟨*city*-1⟩) and move to the type selection phase.

4.2.5 Verb-based extractor

Verbs have been widely exploited as an element to determine the types or roles of its arguments: A verb sense imposes a restriction on the semantic type of its arguments (Quirk et al., 1985; Levin, 1993; Hanks, 1996; Baker et al., 1998; Palmer et al., 2005; Kipper et al., 2008). For instance, from the sentence "Ted Kennedy was elected to Congress," we know that Ted Kennedy is a *person* who can be elected. Corresponding types include ⟨*representative*-1⟩, ⟨*representative*-2⟩, or ⟨*politician*-1⟩. Our verb-based extractor leverages this insight to extract candidate types based on verbs. The extractor operates at the clause level.

A simple way to infer lexical types for entities acting as subjects or objects of a clause is *nominalization*, i.e., the transformation of the verb into so-called *deverbal nouns* (e.g., "play" into "player"). We exploit nominalization as follows. We apply a set of morphological transformations to the verb (Quirk et al., 1985). The set of transformation depends on the grammatical function of the entity, i.e., subject or object. If the entity mention acts as a subject, we try adding the suffixes "-er,", "-or," and "-ant" to the verb's lemma. If the mention acts as an object, we use suffixes "-ee" and "-ed" instead. To obtain candidate types, we again make use of the WordNet's derivationally related forms (DER). In particular, we consider as potential candidates all types referred to by one of the deverbal nouns and are connected to a sense of the verb via the DER relation. For instance, given clause "Messi plays in Barcelona," we collect for "Messi" all the senses of "player" that are connected (via the DER relation) to some sense of "play."; here ⟨*player*-1⟩, ⟨*musician*-1⟩ and ⟨*actor*-1⟩.

We also explore WordNet in a way that is not restricted to morphological variations of the verb. For instance, in sentence "John committed a crime," "commit" is a synonym of "perpetrate," which in turn can be morphologically varied to "perpetrator". We consider the morphological variations of all synonyms of a sense of the verb. Moreover, if the named entity is the subject of the clause, and if the clause also contains a direct object, we try to form a new lexical type by adding the direct object as a noun modifier of the deverbal noun. For example, from "Messi plays soccer", we form the (potential) lexical type "soccer player". If the lexical type exists in WordNet, we consider the respective types as potential candidates as well.

Another more indirect way of exploiting the semantic concordance between types and verbs is via a corpus of frequent (verb, type)-pairs, where the type refers to possible types of the verb's subject or object. As stated above, the set of argument types compatible with a verb is generally limited. For instance, "treat" is usually followed by types like ⟨*condition*-1⟩,

⟨*disease*-1⟩, or ⟨*patient*-1⟩. FINET, uses the corpora of Flati and Navigli (2013) and the VOS repository presented in chapter 3. Given a verb and an entity, we search for frequent candidate types (depending on whether the entity acts as a subject or object). For example, from "Messi was treated in the hospital," we obtain ⟨*patient*-1⟩ in this way.

Once potential candidates have been collected, we perform a KB lookup to decide how to proceed.

4.2.6 Corpus-based extractor

Our final extractor leverages a large unlabeled corpus to find entities that co-occur in similar contexts. The extractor is based on the distributional hypothesis (Sahlgren, 2008): similar entities tend to occur in similar context. For example, "Messi" and "Cristiano Ronaldo" may both be mentioned in the context of sport or, more specifically, soccer. Thus entity mentions similar to "Messi" in a sport context are likely to include other soccer players, such as "Cristiano Ronaldo". Our corpus-based extractor is related to semi-supervised KB methods in that it propagates the types of named entity mentions that may appear in a similar context as the entity under consideration. In contrast, however, it is fully unsupervised and does not require manually or automatically generated training data. Our method also differs in the way context is modeled and candidate types are generated.

Our corpus-based extractor makes use of word vectors (Rumelhart et al., 1988) trained on a large unlabeled corpus. A word vector is a semantic representation of a phrase and represents the semantic context in which the phrase occurs in the corpus. Phrases that are semantically related, and thus appear in similar contexts, are close to each other in the word vector space (e.g., with respect to cosine similarity). For instance, if "Messi" and "Cristiano Ronaldo" tend to co-occur with a similar sets of words, their word vectors are close. As another example, we may expect "Arnold Schwarzenegger" to be close to both other actors but also other politicians, since his name occurs in both contexts. In our work, we use word2vec (Mikolov et al., 2013), which provides a model trained on Google News to predict related words or phrases for a *query*, which is specified as a set of phrases. Given an integer k, word2vec outputs the set of k phrases that are most similar to the query.

Our corpus-based extractor uses (1) the input sentence to construct a set of relevant queries and (2) the word2vec query results and a KB to find candidate types. To construct a query for a given entity mention, we focus on the relevant part of the sentence, i.e., the part that is directly related to the entity. The relevant part consists of the clause in which the entity occurs as well as all subordinate clauses that do not contain another entity mention. Since word2vec is most effective when queries are short, we construct a set of small queries, each consisting of the named entity mention and some context information. In particular,

we construct a query for each noun phrase (of length at most 2) and for each other entity mention in the relevant part of the sentence. Moreover, if the named entity occurs as subject or object, we also take the corresponding verb and the head of the other object or subject as context. For example, the set of queries for "Maradona expects to win in South Africa" is {"Maradona", "South Africa"} and {"Maradona", "expect", "win"}.

For each query, we retrieve the 100 most related words or phrases along with their similarity score and union the results. We filter them using our KB of (entity mention, type)-pairs and retain only those phrases that correspond to entity mentions (with the correct coarse-grained types). Tab. 4.3 shows the top-15 results for query {"Maradona" "South Africa"} of type *person* as well as a subset of their types (assuming a correct and complete KB). We then enrich each mention by the set of their possible types from the KB. Here we exclude widespread but irrelevant implicit types such as ⟨*male*-1⟩, ⟨*female*-1⟩, ⟨*adult*-1⟩, ⟨*commoner*-1⟩, ⟨*friend*-1⟩, or ⟨*alumnus*-1⟩. We also include the types corresponding to the entity mention in the KB (with score 1). If there is sufficient evidence that some of the so-obtained types are most prominent, we take these types as a candidate mention. In our example, all of the top-15 persons have type ⟨*coach*-1⟩, which is a strong indication that Maradona may also be of type ⟨*coach*-1⟩ in our example sentence. We select prominent types as follows: we traverse the result list until we collect 50% of the total score of all results. We take all so-collected types as candidates. If no more than 10 different types were added this way, we directly go to the type selection phase. Otherwhise, we add all types to the candidate set.

4.3 Type Selection

The type selection phase selects the most appropriate type from the set of candidates of a given named entity. We use techniques originally applied for WSD, but adapt them to our setting. In more detail, WSD aims to disambiguate a lexical phrase (e.g., a noun or verb) with respect to a type system as WordNet; e.g., from "player" to ⟨*player*-1⟩. The main difference between classic WSD and our type selection method is that our goal is to decide between a set of types for an entity mention; e.g., from "Messi" to ⟨*soccer_player*-1⟩. Our type selection step can be used as-is for all inputs; it is not trained on any domain- or corpus-specific data.

4.3.1 Obtaining context

In essence, all WSD systems take a set of candidate types and contextual information as input, and subsequently select the most appropriate type. Such methods are thus almost

Mention of person	**Type**
"Diego Maradona"	⟨*coach*-1⟩, ...
"Parreira"	⟨*coach*-1⟩, ...
"Carlos Alberto Parreira"	⟨*coach*-1⟩, ...
"Dunga"	⟨*coach*-1⟩, ...
"Carlos Parreira"	⟨*coach*-1⟩, ...
"Carlos Dunga"	⟨*coach*-1⟩, ...
"Mario Zagallo"	⟨*coach*-1⟩, ...
"Zagallo"	⟨*coach*-1⟩, ...
"Beckenbauer"	⟨*coach*-1⟩, ...
"Jose Pekerman"	⟨*coach*-1⟩, ...
"Lavolpe"	⟨*coach*-1⟩, ...
"Joel Santana"	⟨*coach*-1⟩, ...
"Alberto Parreira"	⟨*coach*-1⟩, ...
"Ephraim Shakes Mashaba"	⟨*coach*-1⟩, ...
"Tele Santana"	⟨*coach*-1⟩, ...

Table 4.3 Top-15 persons from word2vec for query {"Maradona", "South Africa"}

directly applicable to our problem. The key challenge lies in the construction of candidate types, which we discussed in Sec. 4.2, and in the construction of context, which we discuss next. For each entity, we consider *entity-oblivious context* (from the input sentence) as well as *entity-specific context* (using lexical expansions).

We take all words in the sentence as entity-oblivious context (shared by all entities in the sentence). To construct entity-specific context, we make use of lexical expansions, which have been successfully applied in WSD (Miller et al., 2012). Its goal is to enrich context information to boost disambiguation accuracy. In our case, lexical expansions additionally help to differentiate between multiple entities in a sentence. We construct the entity-specific context using word vectors trained from a large unlabeled corpus. As in the corpus-based extractor, we construct a set of queries for the entity mention. In contrast to the corpus-based extractor, we take as context all so-obtained words that do *not* correspond to a named entity. For instance, the entity-specific context for the entity mention "Maradona" for query "Maradona South_Africa" is: "coach", "cup", "striker", "midfielder", and "captain". The full context for "Maradona" in "Maradona expects to win in South Africa" additionally includes the entity-oblivious context "expects", "win", "South Africa".

4.3.2 Selecting types

WSD systems fall into two classes: unsupervised, which rely on background knowledge such as WordNet and differ in the way this knowledge is explored (Ponzetto and Navigli, 2010), and supervised which require training data (Zhong and Ng, 2010). Here we take a combination of both approaches, i.e., we leverage WordNet and manually annotated data.

We train a Naive Bayes classifier to select the most appropriate type given its context. As described above, we represent context by a bag of words, each lemmatized. This simple form of context allows us to automatically generate training data from WordNet (as well as using manually labeled training data). Since WordNet provides useful information for each of the 16k relevant types, this approach combats the data sparsity problem that accompanies supervised systems. We construct appropriate context for each individual WordNet type. The context consists of all words appearing in the type's gloss and the glosses of its neighbors, similar to Extended Lesk (Banerjee and Pedersen, 2003). We also include for each type the neighbors from Ponzetto and Navigli (2010) and the corresponding verbs from the (verb, type)-repository in chaper 3. Finally, we add all words in sentences containing the type in SemCor[3] (Landes et al., 1998) and Ontonotes 5.0 (Hovy et al., 2006).

We trained a separate classifier for each of the coarse-grained types using the above training data, e.g., one classifier that selects a fine-grained type for only *persons*. To train the classifier, we create a single training point for each corresponding WordNet type (the target variable) and use the type's context as features. To map the coarse-grained types from our NER system to WordNet, we considered as persons all descendants of ⟨*person*-1⟩, ⟨*imaginary being*-1⟩, ⟨*characterization*-3⟩, and ⟨*operator*-2⟩ (10584 in total); as locations all descendants of ⟨*location*-1⟩, ⟨*way*-1⟩, and ⟨*landmass*-1⟩ (3681 in total); and as organizations all descendants of ⟨*organization*-1⟩ and ⟨*social group*-1⟩ (1968 in total). This approach of handling coarse-grained types suffers to some extent from WordNet's incompleteness, esp. with respect to persons and organizations. For instance, phrase "sponsored by Coca-Cola" implies that "Coca-Cola" is a "sponsor," but according to WordNet, only persons can be sponsors. Nevertheless, this approach worked reasonably well in our experiments.

4.4 Experiments

We conducted an experimental study on multiple real-word datasets to compare FINET with various state-of-the-art approaches. FINET is used as-is; it does not require training or tuning

[3]http://web.eecs.umich.edu/~mihalcea/downloads.html

for any specific dataset. All datasets, detected types, labels, and our source code are publicly available.[4]

4.4.1 Experimental Setup

Methods. We compare FINET to Hyena and Pearl, two recent systems for fine-grained NET.

Hyena (Yosef et al., 2012). Hyena is a representative supervised NET method that uses a hierarchical classifier. The features of the classifier include the words in the named entity mention, the words in the sentence and paragraph of the mention, as well as part-of-speech tags. Hyena performs basic co-reference resolution and marks entity mentions connected to a type in the KB using a binary feature. Similar to Ling and Weld (2012), Hyena is trained on Wikipedia mentions, each being annotated with its corresponding WordNet types from YAGO. Hyena's type system is restricted to 505 WordNet types from the top categories $\langle$*artifact*-1$\rangle$, $\langle$*event*-1$\rangle$, $\langle$*person*-1$\rangle$, $\langle$*location*-1$\rangle$, and $\langle$*organization*-1$\rangle$. Yosef et al. (2012) compared Hyena to a number of previous systems (Fleischman and Hovy, 2002; Rahman and Ng, 2010; Ling and Weld, 2012) and found that Hyena outperformed these systems. In our experiment, we used Hyena via its web service API (Yosef et al., 2013).

Pearl (Nakashole et al., 2013). Pearl is a semi-supervised NET system that leverages a large repository of relational patterns (Nakashole et al., 2012), which consists of roughly 300k typed paraphrases. Subjects and objects of each pattern carry type information. Pearl types named entity mentions by the most likely type according to its pattern database. Pearl's type system is based on around 200 "interesting" WordNet types. We ran Pearl in its *hard* setting, which performed best; the hard setting additionally makes use of (disjoint) groups of types that are unlikely to appear together in a sentence.

FINET. We ran FINET in two configurations: (1) using the KB lookup described in Sec. 4.2.3, (2) without using the KB lookup. This allows us to estimate the extent to which referring to a KB helps FINET for typing the more explicit types. Note that the corpus-based extractor makes use of the KB in both configurations.

Datasets. We used three different datasets in our experiments. Our datasets represent different real-world use cases. We created two new datasets (New York Times and Twitter) and use as a third dataset a subset of the CoNLL data, which provides gold annotations for coarse-grained NER types. We did not consider datasets such as in FIGER (Ling and Weld, 2012) or BBN (Weischedel and Brunstein, 2005) used in previous studies because these

[4] http://dws.informatik.uni-mannheim.de/en/resources/software/finet/

dataset are generally not suitable for fine-grained typing. The granularity of the type systems from the datasets is not as fine grained as it is required for this evaluation.

New York Times. The New Times Dataset consists of 500 random sentences from the New York Times corpus (Sandhaus, 2008), year 2007; we selected only sentences that contained at least one named entity. We extracted named entity mentions and their coarse-grained types using Stanford CoreNLP 4.4.1.

CoNLL. We sampled 500 sentences from the CoNLL 2003 dataset (Tjong Kim Sang and De Meulder, 2003), a collection of newswire articles with manually annotated entities and their coarse-grained labels. We directly used the provided annotations in our evaluation. The sentences in this dataset tend to be rather short and sometimes non-verbal (e.g., "Jim Grabb (U.S.) vs. Sandon Stolle (Australia)"). Most entities are prominent, i.e., we expect these entities to be present in our KB (as well as the KB used by existing methods).

Twitter. We sampled 100 tweets from recent tweets using Twitter API. We collected the first 100 tweets retrieved containing named entity mentions.

Type system. FINET's type system consists of more than 16k types with top categories persons, locations and organizations. We used the mapping between these top categories and WordNet types described in Sec. 4.3.2. Hyena (505 most frequent WordNet types) and Pearl (200 "interesting" WordNet types) consider a significantly smaller set of types. To compare the performance across different granularities, we classified each type as either coarse-grained (CG), fine-grained (FG) or super fine-grained (SFG). The CG types were $\langle$*artifact*-1$\rangle$, $\langle$*event*-1$\rangle$, $\langle$*person*-1$\rangle$, $\langle$*location*-1$\rangle$ and $\langle$*organization*-1$\rangle$. The FG types were those included in Pearl (Nakashole et al., 2013). All remaining types were considered SFG.

Labeling. All type extractions by all systems were independently labeled by two labelers. We adapted a pessimistic view, i.e., we considered an extraction correct if it was labeled correct by both labelers; otherwise we considered the extraction incorrect. The Cohen's kappa measure ranged 0.54–0.86, which indicates a substantial inter-annotator agreement.

4.4.2 Results

Description of Tab. 4.4. Our results are summarized in Tab. 4.4. We ran each method on each dataset, focusing on either CG, FG or SFG types. When a method did not produce a type of the considered granularity but a more fine-grained type instead, we selected its closest hypernym. For each configuration, the table shows the number of named entities for which types have been extracted, the total number of extracted types (more than one distinct type per named entity for some methods), the total number of correct types, and the precision of

System	Coarse-Grained (CG)			Fine-Grained (FG)			Super Fine-Grained (SFG)			Distinct	Avg. Depth		Cohen's
	Enti-ties	Total types	Correct types *(P)*	Enti-ties	Total types	Correct types *(P)*	Enti-ties	Total types	Correct types *(P)*	types	FG	SFG	kappa
New York Times (500 sentences)													
FINET	992	992	872 *(87.90)*	616	631	457 *(72.42)*	319	329	233 *(70.82)*	191	5.96	7.25	0.60
FINET (w/o KB l.)	992	992	872 *(87.90)*	598	613	436 *(71.13)*	294	304	204 *(67.11)*	174	5.98	7.18	0.58
Hyena	895	1076	779 *(72.40)*	770	1847	522 *(28.26)*	518	775	160 *(20.65)*	127	5.79	6.98	0.74
Pearl (hard)	15	15	5 *(33.33)*	2	2	0 *(–)*	–	0	– *(–)*	1	–	–	0.54
CoNLL (500 sentences)													
FINET	1355	1355	1355 *(1.0)*	1074	1086	876 *(80.66)*	668	679	510 *(75.11)*	136	6.09	7.38	0.62
FINET (w/o KB l.)	1355	1355	1355 *(1.0)*	1075	1087	869 *(79.94)*	661	672	498 *(66.13)*	134	6.06	7.35	0.62
Hyena	1162	1172	1172 *(1.0)*	1064	2218	1329 *(59.92)*	719	944	268 *(28.39)*	103	5.89	6.57	0.69
Pearl (hard)	18	18	18 *(1.0)*	8	11	5 *(45.45)*	–	–	– *(–)*	7	5.6	–	0.74
Twitter (100 tweets)													
FINET	135	135	123 *(91.11)*	103	104	69 *(66.35)*	54	54	33 *(61.11)*	40	6.25	7.64	0.58
FINET (w/o KB l.)	135	135	123 *(91.11)*	104	105	65 *(61.90)*	56	56	30 *(53.57)*	40	6.14	7.6	0.55
Hyena	125	146	105 *(71.91)*	117	280	75 *(26.79)*	91	129	21 *(16.28)*	42	6.11	6.19	0.67
Pearl (hard)	10	10	5 *(50.00)*	3	4	1 *(25.00)*	–	–	– *(–)*	3	6	–	0.86

Table 4.4 Summary of results

each method (*P*). The number of named entities for which types have been found and the total number of correct extractions can be seen as a loose measure of recall. In general, it is difficult to estimate recall directly for FG and SFG types because some entities may be associated with multiple types, and some with no FG type. To gain more insight into the extracted types, we also show the number of correct distinct types that have been extracted, and the average depth (shortest path from $\langle$*entity*-1$\rangle$ in WordNet) for both correct FG and correct SFG types. Finally, we list the Cohen's kappa inter-annotator agreement measure for each method.

Discussion. First note that Pearl extracted significantly fewer types than any other system, across all configurations. Pearl does not support SFG types. For CG and FG, we conjecture that Pearl's pattern database did not reflect well the syntactic structure of the sentences in our datasets so that often no match was found. In fact, Pearl's pattern set was generated from Wikipedia; its patterns may be less suitable for our datasets. This finding strengthens the case for the use of heterogeneous sources in semi-supervised methods.

Hyena performed better than Pearl and in many cases extracted the largest number of types. This is because Hyena tended to extract multiple types per named entity and, in almost all cases, provided at least one FG type. This more recall-oriented approach, as well as its context-unaware use of supervision, significantly reduced the precision of Hyena so that a large fraction of the extracted types were incorrect.

FINET had significantly higher precision across all settings, especially for SFG types, for which FINET achieved almost three times more precision than Hyena. One reason for this boost is that FINET is conservative: We provide more than one type per named entity only if the types were explicit (i.e., come from the pattern-based extractor). In all other cases, our type selection phase produced only a single type. FINET extracted the largest number of correct SFG types on each dataset. Hyena extracted more FG types, but with a significantly lower precision. The average depth of correct FG and SFG types in FINET was higher than that of Pearl and Hyena, FINET also tended to use more distinct correct types (191 in NYT vs. 127 for Hyena). Again, this more fine-grained typing stemmed from FINET's use of multiple extractors, many of which do not rely on supervision.

Note that FINET also has higher precision for CG types than Hyena. As described earlier, FINET makes use of the Stanford NER tagger to extract CG types (except in CoNLL, were we used the provided manual labels), and respects these types for its FG and SFG extractions. Hyena has lower precision for CG types than FINET because it sometimes outputs multiple CG types for a single named entity mention. Pearl does not make use of Stanfords NER tagger to extract CG types, but uses its pattern database instead. To ensure a fair comparison, we indirectly used the gold labels for CoNLL for Pearl and Hyena by discarding all produced FG and SFG types with an incorrect CG type.

FINET's extractors. Tab. 4.5 shows individual influence of each of FINET's extractors; here we used the NYT dataset with our full type system. The table shows the number of entities typed by each extractor and the precision of the resulting type after type selection. The mention-based extractor was the most precise and also fired most often; this was mainly due to locations. The pattern-based extractor also had a good precision and tended to fire often. The first three extractors, which focus on the more explicit types, together generated more than half of the extracted types; this indicates that explicit type extractors are important. There were also a substantial fraction of implicit types, which were covered by our corpus-based extractor. The verb-based extractor had lowest precision, mostly because of the noisiness and incompleteness of its underlying resources (such as the (verb,type)-repository). In fact, we expect overall precision to increase if this extractor is removed. However, this would hinder

Last used extractor	Entities	P
Pattern-based	180	71.11
Mention-based	219	82.65
Verb-based	47	48.94
Corpus-based	205	64.39

Table 4.5 Per-extractor performance on NYT (all types)

FINET to infer types from verbs. Thus instead of removing the extractor, we believe a better direction is to conduct more research into improving the underlying resources.

Error analysis. One major source of error for FINET were incorrect coarse-grained labels. We found that when CG labels were correct (by Stanford NER), the precision of FINET for FG types increased to more than 70% for all datasets. When FG labels were correct, the precision of SFG labels exceeded 90%.

Incompleteness of and noise in our underlying resources also affected precision. For example, some types in WordNet have missing hypernyms, which reduced recall; e.g., sponsor in WordNet is a person but cannot be an organization. WordNet is also biased towards US types (e.g., *supreme court* only refers to the US institutions). Our repositories of verbs and their argument types are incomplete and noisy as well. Finally, errors in the KB affected both KB lookups and our corpus-based extractor. One example of such errors are temporal discrepancies; e.g., in the sports domain, a person who used to be a $\langle$*player*-1$\rangle$ may now be a $\langle$*coach*-1$\rangle$. The KB types are also noisy, e.g., many soccer players in Yago2 are typed as $\langle$*football_player*-1$\rangle$ and the United Nations is typed as a $\langle$*nation*-1$\rangle$.

Finally, the type selection phase of FINET introduced mistakes (i.e., even when the correct type was in the candidate set, type selection sometimes failed to select it). This is especially visible for our verb-based extractor, which may produce a large number of nominalizations and thus make type selection difficult.

Hyena mainly suffered from the general problems of supervised systems for NET. For instance, since $\langle$*graduate*-1$\rangle$ or $\langle$*region*-1$\rangle$ are highly frequent in the KB, many persons (locations) were incorrectly typed as $\langle$*graduate*-1$\rangle$ ($\langle$*region*-1$\rangle$). Errors in the KB also propagate in supervised system, which may lead to "contradictory" types (i.e., an entity being typed as both $\langle$*person*-1$\rangle$ and $\langle$*location*-1$\rangle$).

4.5 Related Work

Taxonomy Induction and KB construction The NET problem is related to taxonomy induction (Snow et al., 2006; Wu et al., 2012; Shi et al., 2010; Velardi et al., 2013) and KB construction (Lee et al., 2013; Mitchell et al., 2015; Paulheim and Bizer, 2014), although the goals are different. Taxonomy induction methods aim to produce or extend a taxonomy of types, whereas KB construction methods aim to find new types for the entities present in some KB. In both cases, this is done by reasoning over a large corpus and each distinct entity is assigned a type. Those methods do not type each occurence of the entity but try to find measures to determine the best fitting type for them according to all the occurences in the corpus. In contrast, we are interested in typing each named entity mention individually according to each particular context in which it occurs using an existing type system. Nevertheless, FINET draws from ideas used in taxonomy induction or KB construction.

Existing systems are either based on patterns or the distributional hypothesis. In a pattern based approach, the system uses a set of fixed manually crafted or learned patterns to perform the extractions. Patterns gather types for each entity and a general measure (e.g., frequency) is used to determine which types are the most appropriate. In contrast, the distributional hypothesis, states that semantically related terms tend to occur in similar context. Co-occurency is the main aspect to assign types to entities in a distributional hypothesis setting. These two approaches are discussed and compared in Shi et al. (2010). In FINET, we make use of patterns (such as the ones of Hearst (1992)) in most of our extractors, and of the distributional hypothesis in our corpus-based extractor.

Open domain class extraction Open domain class extraction has recently gained significant attention (Pasca, 2013). It tends to infer lexical types for entities without a predefined type system, mostly from user intended input like search query logs (e.g. "List of US presidents"). These classes are constructed from a noun and a set of modifiers (e.g., "cars", "electric cars", "electric cars of Germany"), where the root is likely to correspond to a WordNet type. FINET shares characteristics with these systems in the sense that our explicit extractors try to construct the most specific lexical type for the entity, which is only latter mapped to the WordNet type hierarchy.

Open Information Extraction In the context of open information extraction, Yahya et al. (2014) developed a pattern-based semi-supervised method that attempts to extract propositions such as "president"(*"Barack Obama"*, *"US"*), in which the relation can be seen as a type. None of the elements in the proposition are disambiguated. FINET differs in that it supports implicit types, and produces disambiguated types.

Named entity recognition NER is the task of recognizing named entities in natural language text (Nadeau and Sekine, 2007). NER is a task which has achieved a significant performance in terms of precision and recall. A number of tools are openly available for NER. Finkel et al. (2005), for instance, is a CRF-based system which recognize named entities in raw text and classifies them according to a set of coarse grained types (e.g., *person*, *location*, *organization*, etc) with high performance (ca. 80-90% F1). We use this tool to recognize and type entities with their coarse-grained types with a precision of around 90% across datasets.

Semi-supervised NET systems A number of NET systems have been proposed in the literature specifically regarding fine grained typing. These systems generally make use of a predefined type hierarchy. Lin et al. (2012) is a semi-supervised system developed in the context of open information extraction which uses relational patterns to propagate type information from a KB to entity mentions acting as subject. This work is not restricted to named entities but focus on entities in general. It first recognize entities from noun phrases via a classifier, whose main features try to capture the time frame in which the entity occurs. The assumption is that usage patterns between unlikable entities and non-entities can be identified across time; patterns used for entities do not vary accoss time as well as patterns used for non-entities. The idea is that noun phrases from old text corpora that cannot be linked to a KB tend to be non-entities. Therefore, entities and non-entities can be characterized via patterns in old corpora. Finally, once the classifier has recognized an unlikable entity, types are propagated from known entities via relational patterns. Each linkable entity is characterized via relational patterns occurring in their Wikipedia page. So if an unlikable entity occurs in similar patterns to a linkable entity, types from Freebase are propagated.

Similarly, Pearl (Nakashole et al., 2013), another semi-supervised system, is based on a corpus of typed relation patterns (Nakashole et al., 2012) for around 200 WordNet classes extracted from Wikipedia sentences containing YAGO named entities. Each relational pattern links subject and object entity types. The idea is that given a pattern, if the pattern is in the repository the subject and objects entities are typed according to the information provided by the typed pattern in the repository. Compared to Lin et al. (2012), Pearl can type entities not only in the subject but also in the object of the relation. Due to its pattern-based design Pearl tends to have low recall.

Supervised NET systems An alternative approach is taken by supervised methods, which train classifiers based on linguistic features (Fleischman and Hovy, 2002; Rahman and Ng, 2010; Ling and Weld, 2012). Both Hyena (Yosef et al., 2013) and FINGER (Ling and Weld,

2012) use Wikipedia and a KB to generate automatic training data, in which each named entity is annotated according to its types in the KB. Hyena uses YAGO as a KB (type system is based on WordNet) while FINGER makes use of Freebase (Bollacker et al., 2008).

Hyena is implemented as a top-down hyerarchical classifier. It starts from the top categories and advances down in the hyerarchy accorrding to a threshold which indicates if the type should be further specified. Hyena relies on features such as the surrounding words (both in the sentence and in the paragraph) where the entity occurs and POS tags. It also incorporates a feature which indicates if the entity mention is assciated in YAGO to a specific type. Hyena is the most fine-grained type system so far with a set of 505 WordNet types. This subset is selected according to the prominence of the types. However, it lacks important types such as *president* or *businessman*, and includes *soccer player* but not *tennis player*.

FINGER trains a CRF which uses as features the tokens of the entity mention, the POS tags, the syntactic dependencies of the head of the entity mention, the Reverb relational pattern involving the mention, among others. FINGER type system consists of 112 freebase types organized in a hierarchy.

In contrast to supervised or semi-supervised FINET is less reliant on a KB or training data, which improves both precision (no bias against KB types) and recall (more fine-grained types supported). FINET is to date the system using the most fine-grained types. FINET operates on the the entire set of types provided by WordNet, with more than 16k types for persons, organizations, and locations.

Word Sense Disambiguation Our type selection phase is based on WSD (Navigli, 2012), a classification task where every word or phrase is disambiguated against senses from some external resource such as WordNet. Supervised WSD systems (Dang and Palmer, 2005; Dligach and Palmer, 2008; Chen and Palmer, 2009; Zhong and Ng, 2010) use a classifier to assign such senses, mostly relying on manually annotated data. KB methods as Agirre and Soroa (2009), Ponzetto and Navigli (2010), Miller et al. (2012) and Agirre et al. (2014) or the one presented in chapter 3 make use of a background KB instead. For a deeper treatment of WSD related work refer to the related work section on Chapter 3.

4.6 Conclusion

We presented FINET, a system for fine-grained typing of named entities in context. FINET generates candidates using multiple extractors, ranging from explicitly mentioned types to implicit types, and subsequently selects the most appropriate type. Our experimental study indicates that FINET has significantly better performance than previous methods. FINET

would benefit from the improvement of the underlying resources specially those concerned with the verb-based extractor.

Chapter 5

Conclusion and Future Directions

This work is centered on the initial stages of a bottom-up perspective to automatic text understanding. Conceptually, the goal of automatic text understanding is to generate a system that replicates human text understanding capabilities. In our bottom-up perspective, automatic text understanding can be thought of as a set of interleaved tasks which aim to construct a computer-based knowledge-base from natural language text. The knowledge-base, which is conceived as a set of computer-readable facts, should ideally capture all the information in the original text.

Under this vision, the tasks in the pipeline build upon each other, and each subsequent task achieves more semantic understanding with respect to the previous one, increasing the "understanding degree" and opening-up the set of possible end-user applications that can be developed. The complexity (or the intelligence) of the possible applications increases as the semantic information gets deeper. It could also be said that each application has a specific need in terms of the amount of semantic information (or "understanding degree"). The idea is that this computer-readable semantic information constitutes the knowledge available to the machine, which in addition to certain reasoning capabilities generate applications serving the most varied purposes like, for example, keyword search, semantic search, question answering or dialogue systems to mention just a few.

In this work we focus on the initial stages of this automatic text understanding pipeline. Open information extraction, word entry recognition and disambiguation, and named entity typing are fundamental building blocks to recognize textual expressions of facts in natural language text, and unveil the fundamental semantics of its key components. They can also serve specific end-user applications and constitute a valuable input for other tasks in the pipeline performing deeper text understanding.

Specifically, we contribute with three methods: ClausIE, Werdy and FINET. As a general concept, the methods respect three postulates that, we believe, should be respected by any

text understanding task: they are mostly unsupervised, domain independent and based on linguistic principles. Regarding this last element our hypothesis is that as language already entails a systematic way of expressing information, linguistic knowledge should play an important role in any automatic text understanding task.

Our first method, ClausIE, is an open information extraction system. In a bottom-up approach, open information extraction can be seen as the first step towards any automatic text understanding task. It attempts to identify propositions (i.e. a textual expression of a potential fact) in text and represent them in an amenable way for computers. Propositions can be used as input for tasks such as structured search, relation or event extraction, semantic role labeling, knowledge-based construction, among others.

ClausIE aims to discover propositions in text through a set of basic linguistic principles. In contrast to previous approaches, the method does not rely on any manual or automatic training data, and it conceptually separates the recognition of the information from its materialization. ClausIE achieves significantly higher precision and recall than previous open information extraction methods. As it is fundamentally based on deep syntactic analysis of the sentence, it is reasonable to assume that as long as dependency parsing techniques became more accurate and faster (something in principle expected), ClausIE will also become more accurate and scalable.

Regarding future directions, the natural next step for ClausIE would be to achieve more flexibility in the proposition generation phase. Allowing tunable expressions for relations would make it useful for specific applications which require specific relation forms. Also, allowing shorter arguments or even to allow the user to focus on some specific arguments would be an interesting improvement. All these elements would make the system more customizable and therefore more user-friendly. Another possible direction would be to enrich the non-verb mediated proposition extractors and to extract propositions by reasoning directly from textual data (e.g. the pattern and verb based extractors from FINET, without the type selection step).

Our second contribution, Werdy, provides a principled method to recognize words (and multi-word expressions) and disambiguate verbs, a key component of a relation in a proposition. Recognizing words or multi-word expressions is the first step to word sense and named entity disambiguation, or any task requiring to recognize the words or phrases present in natural language text. Regarding the disambiguation of the verb, understanding the verb sense can be useful in tasks such as relation extraction, entity typing, semantic role labeling, discourse parsing, etc.

Werdy recognizes words by working at the syntactic level, avoiding heuristics that usually rely on continuous text fragments. For the disambiguation part, Werdy makes use of the

verb context and a background dictionary. To map words to senses, Werdy selects the set of candidate senses for a verb according to the syntactic or semantic context in which the verb appears. Our experiments indicate that incorporating Werdy as a preprocessing task improves the performance on existing disambiguating methods in standard disambiguation tasks.

The natural future direction to improve Werdy is to work on its underlying resources, like WordNet and, more importantly, the VOS repository. Improving WordNet frames would have a direct impact on the syntactic pruning step. WordNet frames are currently a mix of syntactic and semantic frames but a bit imprecise at times. Mapping WordNet frames to clause types, for instance, would reduce the ambiguity of the current frames and the loss in performance which results from the arbitrary mapping between WordNet frames and clause types. On the other hand, making the VOS repository more complete would have a direct impact on Werdy performance via the semantic pruning step. First, by reducing mistakes arising from this incompleteness and second, by increasing its pruning power.

In our third method, FINET, we type named entities. A named entity is together with the verb a key component in a proposition. Entity types, are important to understand the semantics of a named entity. Entity types may be useful to prune the set of candidates in named entity disambiguation tasks, for relation extraction or relation clustering, allowing generalizations across propositions. Entity types are also relevant in end-user applications such as semantic search or question answering.

FINET, unlike previous approaches, aims to select the type of the entity which is closer to the context where it occurs. FINET is designed through a set of type extractors ranging from explicit to implicit extractors. The idea is to bias the system to select types which are as close to the context as possible. FINET exploits rules on how entity types are expressed in language. It also uses the most fine-grained type system so far with more than 16k types for persons, organizations and locations. FINET achieves high precision outperforming state-of-the-art methods on real-world datasets and it provides more fine-grained types that are also closer to the entity mention context.

FINET would benefit from improvements in both ClausIE and Werdy. In the first case, ClausIE determines the entity context. In the second, the boost in precision may have a big impact through the verb-based extractor which currently has a lower precision with respect to the other extractors. FINET would also benefit with improvements on the type selection step, mainly by enriching the context information of each synset. Furthermore, a better type description in the underlying knowledge-base would help in the corpus-based extractor but also in the other extractors which use a knowledge-base-lookup.

All our methods should not be thought as completely independent tasks but as part of the text understanding pipeline we have previously described. One method builds upon the other as we move to the higher end of the pipeline. Following ClausIE, both methods Werdy and FINET work at clause level. Werdy also requires the clause structure both to recognize word entries and in its syntactic pruning step. In FINET, a clause defines the scope in which the patterns apply, it intervenes in the nominalization of the verb and it also defines the contextual scope of the named entities. Finally, FINET also incorporates Werdy principles to recognize the words and multi-word expressions, and to exploit the interaction between a verb and its arguments. FINET also makes use of Werdy's VOS repository.

Overall, this work presented methods covering three fundamental building blocks of a bottom-up text understanding pipeline, aiming to provide strong foundations for deeper understanding tasks.

List of Figures

List of Tables

Bibliography

Agirre, E., de Lacalle, O. L., and Soroa, A. (2014). Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

Agirre, E. and Soroa, A. (2009). Personalizing pagerank for word sense disambiguation. In *Proceedings of EACL*, pages 33–41.

Akbik, A. and Broß, J. (2009). Wanderlust: Extracting Semantic Relations from Natural Language Text Using Dependency Grammar Patterns. In *Workshop on Semantic Search at WWWW*.

Akbik, A. and Löser, A. (2012). Kraken: N-ary facts in open information extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction at AKBC-WEKEX*, pages 52–56.

Arranz, V., Atserias, J., and Castillo, M. (2005). Multiwords and word sense disambiguation. In *Computational Linguistics and Intelligent Text Processing*, volume 3406 of *Lecture Notes in Computer Science*, pages 250–262.

Atkins, B. T. S. and Rundell, M. (2008). *The Oxford Guide to Practical Lexicography*. Oxford University Press.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley Framenet project. In *Proceedings of ACL*, pages 86–90.

Banerjee, S. and Pedersen, T. (2003). Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of IJCAI*, pages 805–810.

Banko, M., Cafarella, M. J., Soderl, S., Broadhead, M., and Etzioni, O. (2007). Open information extraction from the web. In *Proceedings of IJCAI*, pages 2670–2676.

Bast, H. and Haussmann, E. (2013). Open information extraction via contextual sentence decomposition. In *Proceedings of ICSC*, pages 154–159.

Biega, J., Mele, I., and Weikum, G. (2014). Probabilistic prediction of privacy risks in user search histories. In *Proceedings of the First International Workshop on Privacy and Secuirty of Big Data at PSBD*, pages 29–36.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*, pages 1247–1250.

Bunescu, R. C. and Mooney, R. J. (2005). A shortest path dependency kernel for relation extraction. In *Proceedings of HLT and EMNLP*, pages 724–731.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Jr., E. R. H., and Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *Proceedings of AAAI*.

Chan, Y. S., Ng, H. T., and Zhong, Z. (2007). Nus-pt: Exploiting parallel texts for word sense disambiguation in the english all-words tasks. In *Proceedings of SemEval*, pages 253–256.

Chen, J. and Palmer, M. (2009). Improving english verb sense disambiguation performance with linguistically motivated features and clear sense distinction boundaries. *Language Resources and Evaluation*, 43(2):181–208.

Cholakov, K., Eckle-Kohler, J., and Gurevych, I. (2014). Automated verb sense labelling based on linked lexical resources. In *Proceedings of EACL*, pages 68–77.

Christensen, J., Mausam, Soderland, S., and Etzioni, O. (2010). Semantic role labeling for open information extraction. In *Proceedings of the Workshop on Formalisms and Methodology for Learning by Reading at HLT-NAACL*, pages 52–60.

Dang, H. T. and Palmer, M. (2005). The role of semantic roles in disambiguating verb senses. In *Proceedings of ACL*, pages 42–49.

de Marnee, M.-C. and Manning, C. D. (2012). Stanford typed dependencies manual.

Del Corro, L., Abujabal, A., Gemulla, R., and Weikum, G. (2015). Finet: Context-aware fine-grained named entity typing. In *Proceedings of EMNLP*, pages 868–878.

Del Corro, L. and Gemulla, R. (2013). Clausie: clause-based open information extraction. In *Proceedings of WWW*, pages 355–366.

Del Corro, L., Gemulla, R., and Weikum, G. (2014). Werdy: Recognition and disambiguation of verbs and verb phrases with syntactic and semantic pruning. In *Proceedings of EMNLP*, pages 374–385.

Dligach, D. and Palmer, M. (2008). Improving verb sense disambiguation with automatically retrieved semantic knowledge. In *Proceedings of ICSC*, pages 182–189.

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. (2014). Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of SIGKDD*, pages 601–610.

Dong, X. L., Gabrilovich, E., Murphy, K., Dang, V., Horn, W., Lugaresi, C., Sun, S., and Zhang, W. (2015). Knowledge-based trust: Estimating the trustworthiness of web sources. *PVLDB*, 8(9):938–949.

Drumond, L., Rendle, S., and Schmidt-Thieme, L. (2012). Predicting rdf triples in incomplete knowledge bases with tensor factorization. In *Proceedings of SAC*, pages 326–331.

Etzioni, O., Banko, M., and Cafarella, M. J. (2006). Machine reading. In *Proceedings of AAAI*, pages 1517–1519.

Etzioni, O., Fader, A., Christensen, J., Soderland, S., and Mausam (2011). Open information extraction: The second generation. In *Proceedings of AAAI*, pages 3–10.

Evans, R. J. (2011). Comparing methods for the syntactic simplification of sentences in information extraction. *Literary and Linguistic Computing*, 26(4):371–388.

Fader, A., Soderland, S., and Etzioni, O. (2011). Identifying relations for open information extraction. In *Proceedings of EMNLP*, pages 1535–1545.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Bradford Books.

Feng, V. W. and Hirst, G. (2012). Text-level discourse parsing with rich linguistic features. In *Proceedings of ACL*, pages 60–68.

Ferragina, P. and Scaiella, U. (2010). Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of CIKM*, pages 1625–1628.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*, pages 363–370.

Finlayson, M. A. (2014). Java libraries for accessing the princeton wordnet: Comparison and evaluation. In *Proceedings of GWC*.

Finlayson, M. A. and Kulkarni, N. (2011). Detecting multi-word expressions improves word sense disambiguation. In *Proceedings of MWE*, pages 20–24.

Flati, T. and Navigli, R. (2013). Spred: Large-scale harvesting of semantic predicates. In *Proceedings of ACL*, pages 1222–1232.

Fleischman, M. and Hovy, E. (2002). Fine grained classification of named entities. In *Proceedings of COLING*, pages 1–7.

Foundation, W. (2015). Wikidata. [Online; accessed 20-July-2015].

Galárraga, L. A., Teflioudi, C., Hose, K., and Suchanek, F. (2013). Amie: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of WWW*, pages 413–422.

Gamallo, P., Garcia, M., and Fernández-Lanza, S. (2012). Dependency-based open information extraction. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, pages 10–18.

Grefenstette, E., Blunsom, P., de Freitas, N., and Hermann, K. M. (2014). A deep architecture for semantic parsing. *CoRR*, abs/1404.7296.

Gupta, R., Halevy, A., Wang, X., Whang, S. E., and Wu, F. (2014). Biperpedia: An ontology for search applications. *Proceedings of VLDB Endowment*, 7(7):505–516.

Gurevych, I., Eckle-Kohler, J., Hartmann, S., Matuschek, M., Meyer, C. M., and Wirth, C. (2012). Uby - a large-scale unified lexical-semantic resource based on lmf. In *Proceedings of EACL*, pages 580–590.

Hanks, P. (1996). Contextual dependency and lexical sets. *International Journal of Corpus Linguistics*, 1(1):75–98.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING*, pages 539–545.

Hernault, H., Prendinger, H., duVerle, D. A., and Ishizuka, M. (2010). HILDA: A Discourse Parser Using Support Vector Machine Classification. *Dialogue and Discourse*, 1(3):1–33.

Hoffart, J., Milchevski, D., and Weikum, G. (2014). Stics: Searching with strings, things, and cats. In *Proceedings of SIGIR (demo)*, pages 1247–1248.

Hoffart, J., Suchanek, F. M., Berberich, K., and Weikum, G. (2013). Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. (2011). Robust disambiguation of named entities in text. In *Proceedings of EMNLP*, pages 782–792.

Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., and Weischedel, R. (2006). Ontonotes: The 90In *Proceedings of HLT-NAACL (Companion Volume)*, pages 57–60.

Kilgarriff, A. and Rosenzweig, J. (2000). Framework and results for english senseval. *Computers and the Humanities*, 34(1-2):15–48.

Kipper, K., Korhonen, A., Ryant, N., and Palmer, M. (2008). A large-scale classification of English verbs. *Language Resources and Evaluation*, 42(1):21–40.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of ACL*, pages 423–430.

Krishnamurthy, J. and Mitchell, T. M. (2014). Joint syntactic and semantic parsing with combinatory categorial grammar. In *Proceedings of ACL*, pages 1188–1198.

Kuzey, E. and Weikum, G. (2014). Evin: Building a knowledge base of events. In *Proceedings of WWW (Companion volume)*, pages 103–106.

Landes, S., Leacock, C., and Tengi, R. I. (1998). *Building Semantic Concordances*. MIT Press.

Lee, T., Wang, Z., Wang, H., and Hwang, S.-w. (2013). Attribute extraction and scoring: A probabilistic approach. In *Proceedings of ICDE*, pages 194–205.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2014). DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*.

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of SIGDOC*, pages 24–26.

Levin, B. (1993). *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press.

Lin, T., Mausam, and Etzioni, O. (2012). No noun phrase left behind: detecting and typing unlinkable entities. In *Proceedings of EMNLP and CONLL*, pages 893–903.

Ling, X. and Weld, D. S. (2010). Temporal information extraction. In *Proceedings of AAAI*, pages 1385 – 1390.

Ling, X. and Weld, D. S. (2012). Fine-grained entity recognition. In *In Proceedings of AAAI*.

Mausam, Schmitz, M., Soderland, S., Bart, R., and Etzioni, O. (2012). Open language learning for information extraction. In *Proceedings of EMNLP and CONLL*, pages 523–534.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Miller, T., Biemann, C., Zesch, T., and Gurevych, I. (2012). Using distributional similarity for lexical expansion in knowledge-based word sense disambiguation. In *Proceedings of COLING*, pages 1781–1796.

Miller, T., Erbs, N., Zorn, H.-P., Zesch, T., and Gurevych, I. (2013). Dkpro wsd: A generalized uima-based framework for word sense disambiguation. In *Proceedings of ACL: System Demonstrations*, pages 37–42.

Milward, D. and Beveridge, M. (2003). Ontology-based dialogue systems. In *Proceedings of Workshop on Knowledge and Reasoning in Practical Dialogue Systems at IJCAI*, pages 9–18.

Min, B., Grishman, R., Wan, L., Wang, C., and Gondek, D. (2013). Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of HLT-NAACL*, pages 777–782.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of ACL and IJCNLP*, pages 1003–1011.

Mitchell, T., Cohen, W., Hruscha, E., Talukdar, P., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohammad, T., Nakashole, N., Platanios, E., Ritter, A., Samadi, M., Settles, B., Wang, R., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., and Welling, J. (2015). Never-ending learning. In *Proceedings of AAAI*, pages 2302–2310.

Mocanu, D., Rossi, L., Zhang, Q., Karsai, M., and Quattrociocchi, W. (2014). Collective attention in the age of (mis)information. *CoRR*, abs/1403.3344.

Moro, A., Raganato, A., and Navigli, R. (2014). Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Nakashole, N., Tylenda, T., and Weikum, G. (2013). Fine-grained semantic typing of emerging entities. In *Proceedings of ACL*, pages 1488–1497.

Nakashole, N., Weikum, G., and Suchanek, F. (2012). Patty: A taxonomy of relational patterns with semantic types. In *Proceedings of EMNLP*, pages 1135–1145.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2):10:1–10:69.

Navigli, R. (2012). A quick tour of word sense disambiguation, induction and related approaches. In *Proceedings of SOFSEM*, pages 115–129.

Navigli, R. and Lapata, M. (2010). An experimental study of graph connectivity for unsupervised word sense disambiguation. *EEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):678–692.

Navigli, R., Litkowski, K. C., and Hargraves, O. (2007). Semeval-2007 task 07: Coarse-grained english all-words task. In *Proceedings of SemEval*, pages 30–35.

Navigli, R. and Ponzetto, S. P. (2012). Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193(0):217–250.

Ng, H. T. and Lee, H. B. (1997). Dso corpus of sense-tagged english LDC97T12. Web Download.

Nickel, M., Tresp, V., and Kriegel, H.-P. (2012). Factorizing yago: scalable machine learning for linked data. In *Proceedings of WWW*, pages 271–280.

Palmer, M., Gildea, D., and Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Pasca, M. (2013). Open-domain fine-grained class extraction from web search queries. In *EMNLP*, pages 403–414.

Paulheim, H. and Bizer, C. (2014). Improving the quality of linked data using statistical distributions. *IJSWIS*, 10(2):63–86.

Pedersen, T., Patwardhan, S., and Michelizzi, J. (2004). Wordnet::similarity: Measuring the relatedness of concepts. In *Proceedings of HLT-NAACL (demo)*, pages 38–41.

Petroni, F., Del Corro, L., and Gemulla, R. (2015). Core: Context-aware open relation extraction with factorization machines. In *Proceedings of EMNLP*, pages 1763–1773.

Ponzetto, S. P. and Navigli, R. (2010). Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proceedings of ACL*, pages 1522–1531.

Qazvinian, V., Rosengren, E., Radev, D. R., and Mei, Q. (2011). Rumor has it: Identifying misinformation in microblogs. In *Proceedings of EMNLP*, pages 1589–1599.

Quirk, R., Greenbaum, S., Leech, G., and Svartvik, J. (1985). *A Comprehensive Grammar of the English Language*. Longman.

Rahman, A. and Ng, V. (2010). Inducing fine-grained semantic classes via hierarchical and collective classification. In *Proceedings of COLING*, pages 931–939.

Ravichandran, D. and Hovy, E. (2002). Learning surface text patterns for a question answering system. In *Proceedings of ACL*, pages 41–47.

Recasens, M., de Marneffe, M. C., and Potts, C. (2013). The Life and Death of Discourse Entities: Identifying Singleton Mentions. In *Proceedings of HLT-NAACL*, pages 627–633.

Riedel, S., Yao, L., McCallum, A., and Marlin, B. M. (2013). Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia. Association for Computational Linguistics.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699.

Sag, I. A., Baldwin, T., Bond, F., Copestake, A. A., and Flickinger, D. (2002). Multiword expressions: A pain in the neck for nlp. In *Proceedings of CICLing*, pages 1–15.

Sahlgren, M. (2008). The distributional hypothesis. *Italian Journal of Linguistics*, 20(1):33–54.

Sandhaus, E. (2008). The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 6(12).

Schulder, M. and Hovy, E. (2014). Metaphor detection through term relevance. In *Proceedings of the Second Workshop on Metaphor in NLP*, pages 18–26.

Shen, H., Bunescu, R., and Mihalcea, R. (2013). Coarse to fine grained sense disambiguation in wikipedia. In *Proceedings of *SEM*, pages 22–31.

Shi, S., Zhang, H., Yuan, X., and Wen, J.-R. (2010). Corpus-based semantic class mining: Distributional vs. pattern-based approaches. In *Proceedings of COLING*, pages 993–1001.

Snow, R., Jurafsky, D., and Ng, A. Y. (2006). Semantic taxonomy induction from heterogenous evidence. In *Proceedings of COLING-ACL*, pages 801–808.

Socher, R., Bauer, J., Manning, C. D., and Ng, A. Y. (2013). Parsing with compositional vector grammars. In *Proceedings of ACL*, pages 455–465.

Sonntag, D., Reithinger, N., Herzog, G., and Becker, T. (2010). A discourse and dialogue infrastructure for industrial dissemination. In *Proceedings of IWSDS*, pages 132–143.

Sonntag, D. and Schulz, C. (2014). A multimodal multi-device discourse and dialogue infrastructure for collaborative decision-making in medicine. In Mariani, J., Rosset, S., Garnier-Rizet, M., and Devillers, L., editors, *Natural Interaction with Robots, Knowbots and Smartphones*, pages 37–47.

Stede, M. (2012). *Discourse processing*. Synthesis lectures on human language technologies. Part of: Synthesis digital library of engineering and computer science.

Strzalkowski, T., Broadwell, G. A., Taylor, S., Feldman, L., Yamrom, B., Shaikh, S., Liu, T., Cho, K., Boz, U., Cases, I., et al. (2013). Robust extraction of metaphors from novel data. In *Proceedings of Workshop on Metaphor at ACL*, page 67.

Suchanek, F. M., Sozio, M., and Weikum, G. (2009). Sofie: a self-organizing framework for information extraction. In *Proceedings of WWW*, pages 631–640.

Surdeanu, M. and Ciaramita, M. (2007). Robust Information Extraction with Perceptrons. In *Proceedings of Automatic Content Extraction Workshop at NIST*.

Tandon, N., de Melo, G., Suchanek, F. M., and Weikum, G. (2014). Webchild: harvesting and organizing commonsense knowledge from the web. In *Proceedings of WSDM*, pages 523–532.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of HLT-NAACL*, pages 142–147.

Usbeck, R., Ngomo, A. N., Röder, M., Gerber, D., Coelho, S. A., Auer, S., and Both, A. (2014). AGDISTIS - graph-based disambiguation of named entities using linked data. In *Proceedings of ISWC*, pages 457–471.

Velardi, P., Faralli, S., and Navigli, R. (2013). Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707.

Venetis, P., Halevy, A. Y., Madhavan, J., Pasca, M., Shen, W., Wu, F., Miao, G., and Wu, C. (2011). Recovering semantics of tables on the web. *PVLDB*, 4(9):528–538.

Weischedel, R. and Brunstein, A. (2005). BBN Pronoun Coreference and Entity Type Corpus. Technical report, Linguistic Data Consortium.

West, R., Gabrilovich, E., Murphy, K., Sun, S., Gupta, R., and Lin, D. (2014). Knowledge base completion via search-based question answering. In *Proceedings of WWW*, pages 515–526.

Wilks, Y. and Brewster, C. (2009). Natural language processing as a foundation of the semantic web. *Found. Trends Web Science*, 1(38211;4):199–327.

Woods, W. A. (1975). What's in a link: Foundations for semantic networks. In Bobrow, D. G. and Collins, A. M., editors, *Representation and Understanding: Studies in Cognitive Science*, pages 35–82.

Wu, F. and Weld, D. S. (2010). Open information extraction using wikipedia. In *Proceedings of ACL*, pages 118–127.

Wu, W., Li, H., Wang, H., and Zhu, K. Q. (2012). Probase: A probabilistic taxonomy for text understanding. In *Proceedings of SIGMOD*, pages 481–492.

Yahya, M., Berberich, K., Elbassuoni, S., and Weikum, G. (2013). Robust question answering over the web of linked data. In *Proceedings of CIKM*, pages 1107–1116.

Yahya, M., Whang, S. E., Gupta, R., and Halevy, A. (2014). Renoun: Fact extraction for nominal attributes. In *Proceedings of EMNLP*, pages 325–335.

Yosef, M. A., Bauer, S., Hoffart, J., Spaniol, M., and Weikum, G. (2012). HYENA: Hierarchical Type Classification for Entity Names. In *Proceedings of COLING*, pages 1361–1370.

Yosef, M. A., Bauer, S., Hoffart, J., Spaniol, M., and Weikum, G. (2013). HYENA-live: Fine-Grained Online Entity Type Classification from Natural-language Text. In *Proceedings of ACL*, pages 133–138.

Zhong, Z. and Ng, H. T. (2010). It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of ACL (demo)*, pages 78–83.

Zouaq, A. (2011). An overview of shallow and deep natural language processing for ontology learning. In Wong, W., Liu, W., and Bennamoun, M., editors, *Ontology Learning and Knowledge Discovery Using the Web: Challenges and Recent Advances*.

Zouaq, A., Gagnon, M., and Ozell, B. (2009). Unsupervised and open ontology-based semantic analysis. In *Proceedings of LTC*, pages 245–256.