Extracting Information Networks from Text

by

Filipe de Sá Mesquita

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

This work is concerned with the problem of extracting structured information networks from a text corpus. The nodes of the network are recognizable entities, typically people, locations, or organizations, while the edges denote relations among such entities. We use state-of-the-art natural language processing tools to identify the entities and focus on extracting instances of relations. The first relation extraction approaches were supervised and relation-specific, producing new instances of relations known a priori. While effective, this paradigm is not applicable in cases where the relations are not known a priori or when the number of relations is high. Recently, open relation extraction (ORE) techniques were developed to extract instances of arbitrary relations while requiring fewer training examples. Because of their appeal to applications that rely on large-scale relation extraction, a major requirement for ORE methods is low computational cost. Several ORE approaches have been proposed recently, covering a wide range of NLP machinery, from "shallow" (e.g., part-of-speech tagging) to "deep" (e.g., semantic role labeling – SRL), thus raising the question of what is the trade-off between NLP depth (and associated computational cost) and effectiveness. We study this trade-off in depth, and make the following contributions. First, we introduce a fair and objective benchmark for this task, and report on an experimental comparison of 11 ORE methods shedding some light on the state-of-the-art. Next, we propose rule-based methods that achieve higher effectiveness at lower computational cost than the previous best approaches. Also, we address the problem of extracting nested relations (i.e., relations that accept relation instances as arguments) and n-ary relations (i.e., relations with $n > 2$ arguments). Previously, all methods for extracting these types of relations were based on SRL, which can be up to 1000 times slower than methods based on shallow NLP. Finally, we describe an elegant solution that starts with shallow extraction methods and decides, on-the-fly and on a per-sentence basis, whether or not to deploy deeper extraction methods based on dependency parsing and SRL. Our solution prioritizes extra computational resources for sentences describing relation instances that are likely to be extracted by deeper methods. We show experimentally that this solution can achieve much higher effectiveness at a fraction of the cost of SRL.

Preface

Most of the research conducted for this thesis forms part of a project led by Professor Denilson Barbosa within the NSERC Business Intelligence Network on "Discerning Intelligence from Text". Professor Denilson Barbosa has provided guidance for the work presented in this thesis and assisted with the manuscript composition by providing editorial feedback.

Chapter 1 and Chapter 2 are my original work. Chapter 3 appeared at the AAAI International Conference on Weblogs and Social Media, Data Challenge Workshop [65], the ACM Transactions on the Web Journal [62], the ACM SIGMOD/PODS Ph.D. Symposium [63] and the NIST Text Analysis Conference [67]. The technical apparatus, experimental analysis and manuscript composition presented in this chapter are the result of a collaboration between Yuval Merhav and I. Yuval Merhav and I are equal contributors for this work and received guidance from Denilson Barbosa, Wai Gen Yee and Ophir Frieder. I am the sole contributor for the improvements introduced in SONEX 2.0 (Section 3.9). Ying Xu, Aditya Bhargava, Mirko Bronzi and Grzegorz Kondrak assisted with SONEX 2.0's evaluation as presented in Section 3.9.

Chapter 4 is my original work and appeared at the AAAI International Conference on Weblogs and Social Media [64]. The work in Chapter 5 and Chapter 6 appeared at the ACL Conference on Empirical Methods in Natural Language Processing [66]. I was responsible for designing EXEMPLAR and the benchmark for open relation extraction. Jordan Schmidek assisted with part of the implementation of EXEMPLAR and competitor methods. He also assisted on the manuscript composition and experimental analysis. Chapter 7 is my original work.

To my wife Camila

For her self-giving and self-sacrificing love.

Acknowledgements

First and foremost, I would like to thank Jesus for being my Lord and my God. Without Him, life has no purpose. I would like to thank my wife Camila for bearing with a financially poor, mentally absent and mildly depressed graduate student for 7 years. Eu te amo muito, amore. I am also thankful for Sofia, the most precious graduation gift I could have ever asked for. Você é muito especial pro papai, filha!

I would like to thank my advisor, Professor Denilson Barbosa, for pushing me over my limits and persevering through our ups and downs. I am sorry for my stubbornness. I have the utmost respect (and no envy) for your job. Many thanks to the members of the supervisory committee, Randy Goebel, Davood Rafiei, Greg Kondrak, Joerg Sander, Marek Reformat and Giuseppe Carenini, for investing time into improving this work. I would also like to thank my collaborators: Yuval Merhav, Jordan Schmidek, Zhaochen Guo, Mirko Bronzi, Ying Xu and Aditya Bhargava. This work was only possible because of you.

Paizão e Mãezinha, obrigado pelo apoio incondicional, pelas orações e pelas palavras sempre cheias de sabedoria. Priscila, Débora e Thalita, vocês são as melhores irmãs do mundo. Amo muito vocês, família querida. Também quero agradecer meu sogro e sogras (Francisco, Hilda e Tota), cunhados (Bruna e Gabriel) e concunhado/amigo–irmão (Micael) pelo apoio durante o nascimento da Sofia e a defesa do doutorado. Sou muito grato por vocês terem me aceitado na sua família (sabendo que eu ia tirar a Camila de perto de vocês).

I would like to thank Eli Cortez, Steve Meredith, Altigran Soares da Silva, Mario Nascimento and Jon Putz for their valuable advice and support during critical moments of this journey. Eli, você é um grande amigo–irmão. Obrigado por revisar a tese e por me motivar a terminá-la. Steve, Altigran, Mario and Jon, I am grateful for having such great mentors.

I acknowledge the productive conversations with Gerhard Weikum, Evangelos Milios, Raymond Ng, Paolo Merialdo, Tamer Özsu, Rachel Pottinger, Frank Tompa, Renée Miller, Iluju Kiringa and Laks Lakshmanan. Thanks for your kindness and patience.

I would like to thank Tom Hendrickson, Michael Deering, George Minett and everyone at Mitre Media for believing in me and my vision of leveraging data and the scientific method as the foundation for business growth.

This journey would be impossible to bear without these great friends, listed in no particular order: Junior & Jenise, Henrique & Juliana, Steve & Maria Rita, Graham & Eva, Adrian & Keristan, Peter & Kirsten, Julian & Pascale, Logan & Jasmine, Darryl & Denae, Dale & Kathleen, Jon & Esther, Philip & Kyla, Fred & Lilly, Marlos & Poliana, Getulio & Viviane, Brian & Chelsey, Filipe & Rebeca, Filipe & Ana, Andre & Suellene, Rick & Jean, Daniel & Carol, Ben & Andrea, Steve & Nicole, Chris & Natalie, Pimpy & Jeanette, Micah & Kristi, Steve & Jamie, Ben & Alicia, Pirooz & Niousha, Chris & Erika, Josh & Libriel, Kenyê & Ana, Markeetoo & Ninna, Siza & Marta, Levi Lelis, Jackson Zi, Família Magalhães Sousa, Rafael Kitauchi, Jean & Fernanda, Heyde Marques, Beto Oliveira, Milton & Sâmara, Pr. Manoel & Lena, Pr. Júnior & Arthunilza, Pr. Jaime & Luisa e Pr. Horta & Ana. Thanks for sharing life with me, friends.

Sou muito grato a todos das famílias “de Sá” & “Mesquita” pelo carinho e suporte. Sou grato ao vô Evilázio e as vós Marias. Sou grato aos tios e tias: Lúcia, Lúcio, Nelson & Rosanila, Oliveira & Socorro, Thelma, Isabel, Flávio & Janice, Daniel & Neide, Danilo, Otoni, e Bento & Lena. Também sou grato aos primos e primas.

Finally, I acknowledge the generous support of the Natural Sciences and Engineering Research Council of Canada through BIN (the Business Intelligence Network) and Alberta Innovates – Technology Futures.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Information extraction (IE), the task of providing a structured representation of information expressed in text, is a long-standing challenge in natural language processing (NLP) which has been re-invigorated with the ever-increasing availability of textual content online. Examples of invaluable sources of information in textual format are news, encyclopedias, scientific literature, and the blogosphere (i.e., the collection of social media sites). The automatic extraction of information from these corpora promises a viable approach for acquiring knowledge and discovering the issues that engage society in thousands of collective and parallel conversations online.

Two prominent information extraction tasks are *named entity recognition* and *relation extraction*. The former is concerned with detecting mentions to real-world entities (e.g., people, organizations) in a text corpus, while the latter refers to detecting relations involving these entities. For instance, consider the following sentence:

> "Before Georgia was invaded by Russia, its infrastructure became a target of destabilizing cyber attacks."

From this sentence, a named entity recognition system would detect the entities "Georgia" and "Russia", while a relation extraction system would identify that this pair of entities ("Georgia", "Russia") is an instance of the relation "invaded by". This thesis is concerned with the problem of relation extraction and builds upon previous work for named entity recognition.

The traditional approach to relation extraction requires training data for every relation of interest [1, 11, 12, 23, 26, 39, 69, 97]. Methods that follow this approach rely on supervised learning, where a classifier is trained to identify instances of *one single relation*. Particularly, traditional methods require the user to provide a set of annotated sentences describing instances of this relation. Each sentence must be annotated with a pair of entities and a flag indicating whether this pair is an instance of the given relation or not. Examples of annotated sentences for the relation "invaded by" are:

> ($+$, "Before **Georgia** was invaded by **Russia**, its infrastructure became a target of destabilizing cyber attacks.").
>
> ($+$, "After twelve years of not complying with the UN Security Council, **Iraq** was eventually invaded by the **United States**.").
>
> ($-$, "After twelve years of not complying with the **UN Security Council**, Iraq was eventually invaded by the **United States**.").

where entity pairs are highlighted in bold, '$+$' indicates that the annotated pair is a relation instance and '$-$' indicates that the annotated pair is not an instance. Sentences flagged with '$+$' and '$-$' are called positive and negative examples, respectively. The effort of producing these examples *for each relation* is considerably high, making traditional methods a poor choice when the number of target relations is large.

Open relation extraction (ORE) offers an alternative to the traditional approach by extracting unseen relations as they come [6, 19, 28, 45, 59, 71, 86, 95]. The task in ORE is to recognize all relations described in a corpus. To do so, ORE systems must extract instances of any relation and the *predicate* (or label) that describes this relation from the text alone. While some ORE methods require no training data whatsoever, other methods require annotated sentences for training. For ORE methods that do require training data, sentences are annotated with a predicate in addition to the entity pair. For example, consider the following annotated sentence:

> ($+$,"Before **Georgia** was $\underline{\text{invaded by}}$ **Russia**, its infrastructure became a target of destabilizing cyber attacks.")

where the entity pair is in bold and the predicate is underlined. Unlike traditional methods, ORE methods use these sentences to train a classifier that can detect instances of *any* relation, as opposed to instances of a specific relation. These ORE methods assume that people describe most relation instances using a limited number of syntactic patterns, regardless of the relation. Therefore, a classifier can learn these patterns to expose the relation instances described in a corpus. Because the effort of training an ORE method does not depend on the number of target relations, ORE scales better than traditional relation extraction. Therefore, ORE is suitable for applications that require the extraction of a large (or even unknown) number of relations.

ORE methods expose the content of large text corpora as *information networks*[1], where nodes are named entities and edges represent relation instances. These networks summarize the relationships and events described in a corpus. For example, Figure 1.1 shows an *ego*-centric network [42] around the entity "Barack Obama"[2]. This network was built with data extracted from blog posts collected between August and September of 2008, before the United States Presidential Elections. The self-evident power of the network in Figure 1.1 to illustrate the discussions in the blogosphere is very compelling: it accurately shows important entities discussed during the election, and the most prominent relations amongst them. The figure also shows some unexpected connections, such as Britney Spears and Paris Hilton; they were used in a campaign advertisement by John McCain, who tried to associate Barack Obama with the two celebrities who "are often portrayed as frivolous and irresponsible" [20].

1.1 Applications

One important application of ORE and information networks is to support *open-domain question answering* (Open QA), where questions are not bound to any ap-

[1]Information networks differ from knowledge bases in that their goal is to represent every relation instance described in a corpus regardless of whether this instance is a fact or not. However, these networks can be used as a source of information for the automatic construction of knowledge bases.

[2]This network was automatically extracted by one of our methods (SONEX) from a large sample of the blogosphere; the analysis and visualization of the network was done with NodeXL (`http://nodexl.codeplex.com/`).

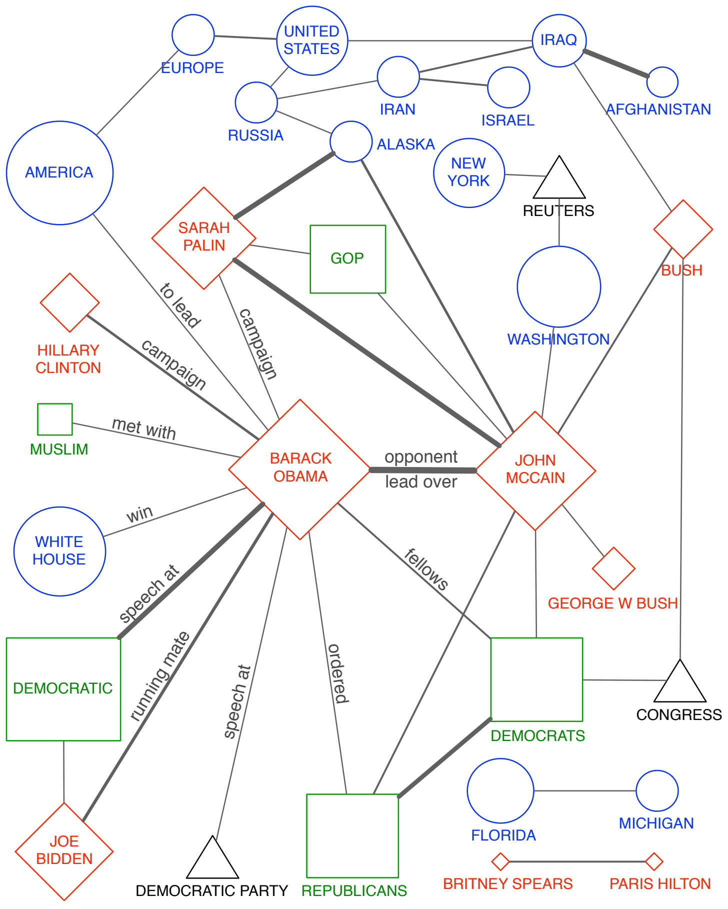


Figure 1.1: Ego-centric perspective of the information network extracted by SONEX from the Spinn3r dataset, focusing on Barack Obama. The Shapes and colors represent entity types: red diamonds indicate Persons, black triangles indicate Organizations, blue circles indicate Locations and green squares indicate Miscellaneous. The size of the nodes indicate their centrality in the network, and the width of the edges indicate their support, measured by the number of sentences that express that relation. For clarity, only edges with highest support are shown.

plication domain [37]. Open QA systems such as Watson [30] extract answers from existing knowledge bases and textual sources. The authors of Watson report that using Freebase[3], a large knowledge base about several domains, can at best help with only 25% of the questions, forcing Watson to rely on textual sources alone for the remaining questions. In this setting, ORE is a better choice than traditional relation extraction since relations of interest are not known in advance.

Information networks can also support entity search [17], which differentiates

[3]http://www.freebase.com.

itself from web search by answering keyword-based queries with a ranking of entities as opposed to a ranking of web pages. The score for an entity relies on the available information about this entity, including every event and relationship in which it is involved. The sheer number of possible events and relationships make traditional relation extraction impractical for entity search.

Information networks have been used for knowledge base population [85], the task of augmenting an incomplete knowledge base with entities and relation instances described in a corpus. These networks have also been used to acquire common sense knowledge [3, 87] and derive logical inference rules from text [53, 54, 84].

1.2 Background

In this thesis, we address the problem of extracting relation instances from a given corpus following the ORE paradigm. Unlike most work on ORE, we do not limit ourselves to binary relations – those that involve only two entities. Instead, we investigate the large-scale extraction of binary, n-ary, flat and nested relations. The problem of ORE is defined as follows.

Definition 1 (Open Relation Extraction)**.** *Given a corpus, extract a set of named entities E and a set of relation instances $R = \{r_1, \ldots, r_m\}$. A relation instance is a tuple $r_i = (p, a_1, \ldots, a_n)$, where p is a* predicate *and a_j is an argument. An argument can be an entity or a relation instance (i.e., $a_j \in E \cup R$). The role of an argument is defined by the function $\rho(r_i, a_j) \rightarrow \{$`subject`, `direct_object`, `prep_object`$\}$, where `prep_object` folds into many roles, one for each preposition in a language.*

Entities. A *named entity* is a textual descriptor (usually a proper name) of a real-world object or abstract concept (e.g., "Capitalism"). Named entity recognition (NER) is the task of identifying named entities in a corpus and their types from a pre-defined set of types [70]. Traditional NER systems focus on recognizing proper names of people, organizations, locations and miscellaneous (i.e., other proper names). The more recent open-domain NER is the task of recognizing entities of any type [27]. This approach is perhaps more suitable for ORE, enabling an ORE method to extract relations involving any type of entity.

The task of grouping names and other *entity mentions* (e.g., noun phrases, pronouns) that refer to the same named entity is called *coreference resolution* [75]. NER and coreference resolution are open problems and continue to receive attention from the NLP community. Solving these problems is outside of the scope of this thesis. Instead, we rely on off-the-shelf tools to recognize and resolve named entities. Other ORE methods in the literature avoid NER altogether and consider each noun phrase in text as a separate entity.

Relations and instances. A relation specifies a class of events (e.g., "invaded by") or relationships (e.g., "colony of") and its instances. A *relation instance* specifies the entities involved in one *individual* event or relationship. In particular, an instance is a tuple $(p, a_1, \ldots, a_n)$, where p is a *predicate* (or label) and a_i is an *argument*. An example of instance is ("invaded by", "Georgia", "Russia"). An argument can be an entity or another relation instance, as discussed later. A relation is a set of all instances that have the same predicate.

Arity. Relations can be classified by the number of arguments they accept. A *binary* relation accepts exactly two arguments and its arguments are tuples of the form (p, a_1, a_2)[4]. Moreover, a generalization of binary relations are *n-ary relations* — relations whose instances present n arguments ($n > 0$). For example, consider the sentence:

> "**Russia** invaded **Georgia** through the region of **South Ossetia**."

This sentence describes the instance ("invaded", "Russia", "Georgia", "South Ossetia"), where each entity plays one of the following *roles*, respectively: `invader`, `defender` and `entry point`.

[4]For better readability, we may also represent binary relation instances as (a_1, p, a_2). It will be clear from context which of these two forms is being used.

Roles. Each argument in an instance plays a different role and this role is defined by the argument's position. Traditional methods require the user to provide the role for each position (e.g., `invader` in the example above). However, ORE methods do not have access to user-provided roles and need to assign roles that can be used for any relation. For this, ORE methods define roles based on grammatical elements. Most ORE methods address the problem of extracting binary relations only and defines that a_1 plays the role of a `subject` and a_2 plays the role of an `object`. For n-ary relations, we adopt additional roles subdividing `object` into more specific roles: `direct_object` and `prep_object`. The role `prep_object` folds into many different roles, one for each preposition of a language.

Nesting. We say that a relation is *flat* when it only accepts entities as arguments. Conversely, *nested* relations accept both entities and other relation instances as arguments. To see an example of nested relations, consider the sentence:

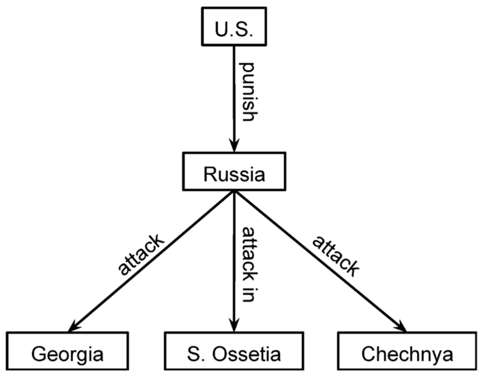
> "The **Associated Press** <u>reported</u> that **Russia** <u>invaded</u> **Georgia** through **South Ossetia**."

Observe that the "Associated Press" reported an invasion event, which can be expressed as a relation instance. Therefore, the reporting event can be expressed as a nested relation instance:

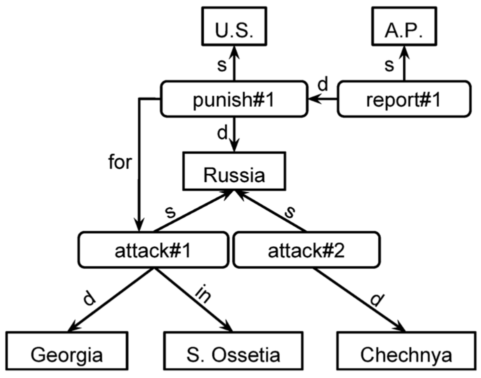
> $\big($"reported", "Associated Press", $\big($"invaded", "Russia", "Georgia", "South Ossetia"$\big)\,\big)$

Nested relations serve to expose the context of an instance, including its source (e.g., who reported an event), repercussion (e.g., who criticized an event) and relationships with other instances (e.g., what happened after an event).

Information networks. An information network, such as the network illustrated in Figure 1.2(a), is a graph where nodes are entities and edges are relation instances. We say that this networks is *flat*, since it cannot represent nested relations. To represent nested relations, we define a *reified* version of information networks. A reified network is a graph where a node is either an entity or a relation instance.



(a) A standard information network exposing events and relationships involving Russia. Entities are nodes and relation instances are labeled, directed edges. An edge's origin node plays the role of a `subject` and the target node plays the role of an `object`. The edge label is the predicate.



(b) A reified version of the same network, showing n-ary and nested relations. A node represents either an entity or an instance. Edges connect instances (origin node) to one of their arguments (target node). A label represents one of the following roles: `subject` (labeled 's'), `direct object` (labeled 'd') and `prepositional object` (labeled with a preposition).

Figure 1.2: Example information networks partly extracted from the the sentence "The A.P. reports that the U.S. will punish Russia for its attack to Georgia in the region of South Ossetia".

Edges connect an instance to one of its arguments. Figure 1.2(b) illustrates a reified information network.

1.3 Challenges

Efficiency. ORE methods are often applied to large corpora and, as a consequence, are required to use as few computational resources as possible. This requirement limits the use of sophisticated features extracted with deep NLP tools. Therefore, the greatest challenges for ORE is detecting predicates and arguments through a shallow representation of a sentence's structure.

Predicate detection. Unlike traditional relation extraction, ORE must extract the predicates from the text. Predicates can be a single word (e.g., "reported") or phrases composed by multiple words ("gave for adoption"). These words are often non-contiguous in the text. Deciding which words compose a predicate is a challenge for ORE. To see this, consider the following example sentence:

“**Bob** gave **Alice** for adoption weeks after she was born.”

where “Bob” and “Alice” are entities and “gave for adoption” is a predicate describing the relation between them. Most ORE methods would detect “gave” as a predicate and miss the words “for adoption”. For this example, “gave” would be an incomplete predicate.

Argument detection. The detection of arguments is also a challenging task. ORE methods often struggle to detect whether an entity appearing near to a predicate is an arguments of this predicate or not. For an example, consider the following sentence:

“**Bramlett** missed a tip-in and **Jefferson** put up an 8-foot air ball as the buzzer sounded.”

where “Bramlett” and “Jefferson” are entities. A method looking at the text between these entities (i.e., “**Bramlett** missed a tip-in and **Jefferson**”) may incorrectly infer that “Jefferson” is an argument of “missed”.

Nested relations. When dealing with nested relations, ORE methods must determine whether the argument of a predicate is an entity or an instance expressed in the same sentence. For an example, consider the following sentence:

“The **Associated Press** reported **Russia**’s conflict with **Georgia**.”

Observe that “Russia” and “Georgia” form an instance describing the conflict between them (“conflict with”, “Russia”, “Georgia”). Furthermore, observe that this instance and “Associated Press” form another instance whose predicate is “reported”. However, without a deep representation of the sentence above, a method may incorrectly recognize “Russia” (as opposed to the conflict) as an argument of “reported”.

1.4 Problem Statement

Massive corpora such as the Web require ORE methods to focus on both *effectiveness* (high precision and recall) and *efficiency* (low computational cost). An important discussion regarding the trade-off between effectiveness and efficiency is whether ORE methods should rely on “shallow” (e.g., part-of-speech tagging) vs “deep” (e.g., full parsing) NLP techniques. One side of the argument favors shallow NLP, claiming deep NLP techniques cost orders of magnitude more and provide much less dramatic gains in terms of effectiveness [19]. The counterpoint, illustrated with a recent analysis on a industrial-scale Web crawl [25], is that the diversity with which information is encoded in text is too high. Framing the debate as “shallow” versus “deep” is perhaps convenient, but nevertheless an oversimplification. Broadly speaking, relation extraction methods can be grouped according to the level of sophistication of the NLP techniques they rely upon: (1) shallow parsing, (2) dependency parsing and (3) semantic role labelling (SRL). This thesis sheds more light into the debate by addressing the following problems:

Problem 1. *Understand the trade-off between computational cost and effectiveness in ORE.*

Problem 2. *Investigate whether it is possible achieve the same level of effectiveness achieved by state-of-the-art ORE systems at considerably lower computational cost.*

To address Problem 1, we propose new benchmarks for ORE and compare the effectiveness and computational cost of over 10 ORE methods. To address Problem 2, this thesis studies ORE methods that are significantly less expensive than methods based on deep NLP, while providing the same level of effectiveness. Prior to this thesis, only methods based on semantic role labeling (SRL) were able to extract n-ary, nested relations [19]. According to our experiments, SRL-based methods can be up to 1000 times slower than methods based on shallow parsing. We investigate methods that can extract all types of relations while requiring a fraction of the computational cost required for SRL.

1.5 Outline

We start by discussing the current state of ORE in Chapter 2. In addition to discussing existing ORE methods, we discuss the existing methodologies for evaluation these methods.

Chapter 3 discusses SONEX [62], a method that efficiently extracts binary relations. SONEX identifies every entity pair (e.g., “Google”, “Apple Inc.”) and all sentences where this pair is mentioned together. From these sentences, SONEX extracts a context (e.g., a list of surrounding words) for the pair and apply clustering techniques to group together pairs with similar contexts. SONEX sees each cluster of entity pairs as a relation. This method aims at high efficiency by analyzing all sentences mentioning an entity pair at once, as opposed to analyze each sentence individually.

In Chapter 4, we discuss a method called Meta-CRF [64] for extracting *nested* relations from text. In this chapter, we describe a machine learning approach leveraging dependency parsing to recognize whether the argument of a relation is an entity or another relation. As expected, the effectiveness gained by applying dependency parsing comes at the expense of additional computational time.

Chapter 5 introduces the EXEMPLAR [66] method and addresses the extraction of n-ary relations. EXEMPLAR relies on handcrafted rules over dependency parsing to detect the precise relationship between an argument and a relation. Unlike its competitors, EXEMPLAR’s rules detect each argument of a relation individually by looking at the path between an entity and a relational word. This allows EXEMPLAR to extract any number of arguments for a particular relation. Our experiments show that our method outperforms its competitors while requiring less computational time.

In Chapter 6, we propose a novel benchmark for ORE [66]. This benchmark allowed us to study the trade-off between effectiveness and efficiency by comparing several ORE methods in a fair and objective way. Somewhat surprising, our evaluation indicates that EXEMPLAR is more effective and more efficient than SRL-based methods when extracting both binary and n-ary relations. EXEMPLAR also presents a higher effectiveness level than methods based on shallow parsing, although it is much less efficient than these methods.

In Chapter 7, we discuss Efficiens, a method that can apply shallow parsing, dependency parsing or SRL over each sentence. While the previous methods present fixed levels of effectiveness and efficiency, Efficiens determines its level of efficiency and tries to maximize its own effectiveness. It does so by taking a user-defined time budget and carefully choosing which tools to apply to each sentence while respecting the budget. Efficiens applies dependency parsing and SRL to sentences that are more likely to benefit from the additional information provided by these tools and, as a result, increase the overall number of correctly extracted instances.

Chapter 8 concludes with an overview of the contributions of this work and possible directions for ORE.

Chapter 2

Background and Related Work

2.1 Natural Language Processing Tools

Information extraction relies on a pipeline of NLP tools to extract sentences and their syntactic structures from a given document. NLP tools represent a document as a list of tokens. Sentence splitting tools detect tokens that mark the end of a sentence. Part-of-speech (POS) taggers annotate each token with one of the POS tags defined by the Penn Treebank [58]. Full parsing tools produce complete parse trees, based on constituency or dependency grammars, for each sentence. On the other hand, shallow (or partial) parsing tools detect flat non-overlapping *segments* (i.e., sequence of consecutive tokens) representing non-recursive phrases, such as verb phrases, noun phrases and prepositional phrases.

2.2 Named Entity Recognition

NER aims at identifying named entities and their types, which are traditionally classified into person, organization, location and miscellaneous [70]. Existing NER tools are able to recognize named entities with accuracy around 90% in well-written texts [31]. The methods presented in thesis leverage two state-of-the-art NER systems: the Stanford NER [31] and the Illinois Named Entity Tagger [76]. Our methods can work with any NER system.

A named entity may be mentioned in many different ways in a corpus. The task of chaining together all mentions that refer to the same entity is known as coreference resolution [4]. The scope of this task can be either a single document (within-document) or the entire corpus (cross-document) [4]. One approach to cross-document coreference is linking entity mentions to entries on an external knowledge-base such as Wikipedia [22]. Our methods rely on the Orthomatcher tool [10] from the GATE framework[1] for within-document coreference resolution. Moreover, our methods also rely on an adaptation of the entity linking method proposed by Cucerzan [22]. However, our methods can work with any coreference resolution tool.

2.3 Open Relation Extraction

The problem addressed in this work is relation extraction, the task of detecting relations among named entities. Traditionally, relation extraction has been formulated as a classification problem, where the task is to decide whether or not a pair of entities (usually co-occurring in the same sentence) is an instance of a given relation [97]. For example, a method trained to recognize the relation "competitor of", must decide whether a given pair ("IBM"–"Apple Inc.") is an instance of this relation or not.

Three traditional relation extraction approaches are prominent: supervised learning, bootstrapping and distant supervision. Methods based on *supervised learning* use classifiers that exploit linguistic and statistical features from hand-tagged sentences [12, 21, 23, 32, 39, 49, 79, 97]. To alleviate the burden of annotating tens of thousands of sentences, recent approaches rely on other sources of training data. Bootstrapping methods require the user to provide a small set of entity pairs belonging to a target relation [1, 11, 26]. These pairs are used to find patterns that can extract more pairs of this relation. Similarly, distant supervision methods [69] leverage databases to collect many entity pairs that are instances of a relation. Using a large corpus, these methods extract all sentences containing any of these entity pairs. Extracted sentences are then used as training data for a supervised classifier.

Although effective in many applications, the traditional paradigm presents limitations. Traditional methods learn to extract a single predefined relation. This

[1]`http://gate.ac.uk/`

requires a user to provide examples for each relation of interest, which is impractical when the number of relations is large. Distant supervision methods, while not requiring training effort, are limited to the relations readily available in databases.

The large-scale extraction of unlimited, unanticipated relations has been termed open relation extraction (ORE) [6].There are two distinct approaches for ORE: (1) unsupervised learning and (2) unlexicalized learning.

2.3.1 Unsupervised methods

Unsupervised methods apply clustering algorithms to produce relations. They rely on the *the context* of an entity pair, that is, the tokens surrounding them in all sentences where they appear together. These methods work under the assumption that the context of an entity pair often describes the relation of this pair. Therefore, they see relation extraction as a clustering problem, where entity pairs are the objects to be clustered, and clusters contain instances of the same relation.

Hasegawa et al. [43] introduced this approach and proposed the use of single tokens (unigrams) in between an entity pair as its context. Following the vector space model [57], an entity pair is represented by a context vector, where each dimension corresponds to a single token. The weight given to each dimension is defined by the popular $tf \cdot idf$ weighting scheme. Context vectors (along their respective entity pairs) are then clustered via hierarchical agglomerative clustering (HAC), using the cosine measure to calculate the similarity between two vectors. Each produced cluster specifies the instances of a relation and the token with the highest average weight within a cluster is the relation's label. Hasegawa et al. evaluated this approach using the 1995 New York Times corpus; they analyzed the data set manually and identified 48 relations containing 242 entity pairs in total. As discussed in Chapter 3, this thesis extends this approach by proposing additional context features (e.g., bigrams, trigrams), a novel weighting scheme that outperforms $tf \cdot idf$ and a larger dataset for evaluation. In addition, we discuss how to overcome limitations inherent to HAC, such as its inability to identify multiple clusters (i.e., relations) for an entity pair and its high time complexity (at least quadratic).

2.3.2 Unlexicalized methods

Unlexicalized methods rely on supervised learning to recognize the general structure in which relations are expressed in English, allowing them to detect instances of any relation. These methods rely on syntactic information only (e.g., POS tags, functional words, dependencies) and ignore content words (e.g., nouns, verbs). Such a framework contrasts with traditional relation extraction, where content words are prominent features used to detect instances of a particular relation.

TextRunner. The inspiring work of TextRunner [5] introduced the unlexicalized approach. For a pair of noun phrases in an individual sentence, TextRunner applies a Naive-Bayes classifier to decide whether the tokens in between the pair describe a relation instance or not. TextRunner is called self-supervised since it uses heuristic rules to produce its own positive and negative examples of how relations are expressed in English. These rules rely on dependency parsing, which can be too computational expensive for large corpora. Therefore, dependency parsing is applied only to a small subset of sentences to produce training examples. These examples are used to learn a classifier that uses POS tagging only, which requires much less computational resources than dependency parsing.

A new version of TextRunner uses Conditional Random Fields as opposed to a Naive Bayes classifier [6]. In this new version, relation extraction is seen as a sequence labeling problem. The tokens in between a pair comprise the input sequence and a sequence of produced labels, one for each token, is the output sequence. Each label defines whether the associated token is a relational token (i.e., it describes the relation between the pair) or not. TextRunner was evaluated on 10 relation types and its performance was compared to the performance of KnowItAll [26], a traditional relation extraction system. The authors also applied small manual evaluations and estimation techniques to evaluate the actual performance of the system when extracting relations from the web. While TextRunner can only extract binary relations, we propose a CRF-based method that is able to extract both binary relations and nested relations in Chapter 4.

WOE. WOE [92] is a method inspired by TextRunner that proposes another way to construct training for ORE. WOE uses the relations in Wikipedia Infoboxes to find sentences in an unlabelled corpus that mention these relations. These sentences are then used as training data; WOE was evaluated with three corpora: WSJ from Penn Treebank [58], Wikipedia[2], and the Web.

ReVerb. TextRunner's authors introduced ReVerb as a next generation ORE system by showing its advantages over TextRunner [28] and WOE. The authors claim that ReVerb is able to extract better predicates by using simple POS patterns as opposed to a supervised classifier. ReVerb extracts three types of predicates: verb, verb+preposition and verb+noun+preposition. The latter predicate type is particularly useful to describe light verb constructions such as "made a deal with". First, ReVerb extracts all phrases matching the aforementioned predicate types and marks them as candidate predicates. For each phrase, ReVerb then heuristically choose the entities surrounding each phrase (i.e., immediately before and after it) as the relation arguments. Since this heuristic may produce many false positives, a logistic regression classifier is applied to assign a confidence score to the relation instance. This allows users to define a confidence threshold according to their sensitivity to false positives and their need to extract a large number of instances. For instance, a high threshold eliminates many instances but produces fewer false positives. ReVerb have been shown to outperform TextRunner in an experiment with 500 sentences [28].

OLLIE. So far we have discussed unlexicalized methods that leverage POS tagging only. Recent methods have explored the benefit of dependency parsing to ORE. These methods look at the dependency path between two entities as a potential description of the relation between them. OLLIE [59] tries to decide whether the words in a dependency path describe a relation instance by learning patterns. This process is as follows. The authors of OLLIE collected 110,000 high-confidence instances extracted by ReVerb from a large Web collection (ClueWeb [3]). Next,

[2]`http://wikipedia.org`

[3]`http://lemurproject.org/clueweb09.php/`

they retrieved 18 million sentences containing these instances and run a dependency parser over all these sentences. For each sentence, OLLIE tried to transform the dependency path between the two arguments into an extraction pattern. This is done by keeping only syntactic restrictions in the nodes (e.g., verbs are matched with POS tags only) and abstracting all preposition edges to a general "prep_*" edge. Some constraints on the path are enforced to avoid creating purely syntactic patterns that do not generalize to all relations. Paths that failed to pass these constraints are turned into semantic-lexical patterns. This is done by grouping all sentences presenting the same failed path and attaching a lexical restriction to one or more nodes. A lexical restriction is a list of words that are allowed for one specific node. Once all extraction patterns are produced, OLLIE tries to match them to new sentences to find new relation instances.

PATTY. Another method based on dependency parsing is used by PATTY [71]. The goal of PATTY is construct a taxonomy of relations, where one may find synonym relations such as "is romantically involved with" and "is dating". PATTY also tries to find relations (e.g., "is dating") that are subsumed by others (e.g., "knows"). Observe that the problem addressed by PATTY is slightly different from ORE; however, PATTY applies an ORE extractor to find the list of relations to be included in the taxonomy. Our discussion focus on PATTY's extractor only. This extractor first detects all named entities in a sentence. For every pair of named entities, it then finds the path in the dependency tree between the two named entities. The extractor then decides whether the path is a relation pattern using heuristics, one of them being that the path must start with one of the following dependency edges: nsubj (nominal subject), rcmod (relative clausal modifier) and partmod (participial modifier).

TreeKernel. The TreeKernel [95] method presents an alternative to extraction patterns for ORE by leveraging SVM tree kernels. TreeKernel consider two problems: (1) whether a sentence describes *any* relation instance involving a given pair of entities and (2) whether a sentence describes a *particular* relation instance (r,

a_1, a_2), where r, a_1 and a_2 are given. Although the first problem is useful to study implicit relations and other relations not mediated by a verb or a noun, this thesis focuses on the second problem. TreeKernel produces candidate relations by applying a set of syntactic patterns, which expands those used by ReVerb. TreeKernel then produces candidate instances by combining all possible entity pairs with candidate relations from a sentence (within certain constraints). Given a candidate instance, TreeKernel extracts either the path between the two entities, if it includes all relational words; or the shortest paths between each entity and a relational word, separately. A tree kernel classifier then decides whether the candidate instance is indeed a relation instance using as features the similarity between the input paths and the paths provided during training. Training data was hand-tagged by the authors over 756 sentences from the Penn Treebank [58]. By using a tree kernel, this method is not limited to a finite number of features and, as a consequence, may present a higher classification accuracy. However, tree kernels require additional computational time and it is not clear whether the extra time pays off in many applications.

SRL-IE. Recently, a method based on semantic role labeling (SRL), called SRL-IE, has shown that the effectiveness of ORE methods can be improved with semantic features [19]. The task of SRL is to identify the arguments of a (verb or noun) predicate. A predicate is a single word that must be mapped into one of the semantic frames, as defined in repositories such as PropBank [74] and NomBank [68]. An argument is any phrase in the sentence acting as an agent, patient, instrument or any other semantic role available for the predicate in the repository. The SRL-IE method [19] uses rules to map SRL annotations into ORE instances. This method treats the predicate (along with its modifiers) as the predicate. The relation arguments are all entities labeled by the SRL system as arguments (including adjuncts), regardless of their roles. We implemented our version of SRL-IE by relying on the output of two SRL systems: Lund [45] and SwiRL [86]. SwiRL is trained on PropBank and therefore is only able to label arguments with verb predicates. Lund, on the other hand, is trained on both PropBank and NomBank, making it able to extract relations with both verb and noun predicates. Most SRL systems employ many classifiers on top of a pipeline of NLP tools that includes POS tagging and dependency parsing. This architecture often makes them more expensive than the ORE methods discussed so far.

2.3.3 Beyond binary relations

While most ORE methods extract binary relations only, OLLIE and SRL-IE are able to extract n-ary ($n > 2$) relations. OLLIE produces n-ary instances by merging binary instances extracted from a the same sentence that have the same subject and predicates (without the preposition). For example, the binary instances

(“A”, “met with”, “B”) and (“A”, “met in”, “C”)

would be merged to form the n-ary instance

(“A”, “met”, “with B”, “in C”).

SRL-IE extracts n-ary relations by detecting each argument of a predicate individually. Rather than detecting the relation between a pair of arguments, SRL-IE detects the relationship between an argument and a predicate. This approach allows SRL-IE to extract any number of arguments for a predicate seamlessly. SRL-IE also detects nested relations when a predicate is contained by an argument in a parse tree.

2.4 Evaluation of Relation Extraction

The creation of evaluation datasets for ORE is an extremely time-consuming task. Figure 2.1 compares current evaluation methods for ORE in two dimensions: (1) scale and (2) ability to measure precision and recall.

Precision and recall at large scale. The evaluation of “true” precision and recall at large scale is an open problem in ORE. Manually annotating every relation described in millions of documents is virtually impossible. We discuss alternatives as follows.

	Precision Only	Precision & Recall
Small Scale	Result inspection	Manual benchmark
Large Scale	Database as ground truth	?

Figure 2.1: Comparison of evaluation methods for open relation extraction.

Manual benchmark. Benchmarks are often created by human annotators, who analyze a given corpus to identify relations described in it; these relations are considered the ground truth for the corpus. The first widely-used RE benchmark was produced by the Message Understanding Conference in 1998 (MUC-7), the last conference of the series [55]. This benchmark contains 3 relations (employee_of, manufacture_of and location_of) and 200 news articles divided equally in training and test sets. MUC was succeeded by the ACE program, who produced benchmarks almost annually from 2003 to 2008. The 2008 benchmark presents 415 news articles annotated with a total of 25 relations [72]. Because of their focus on traditional RE and limited number of relations, ACE and MUC benchmarks are often inappropriate for open RE.

Chapters 6 discusses a novel benchmark for ORE. Our benchmark is the first to support the evaluation of a wide range of ORE methods by annotating predicates and their arguments in thousands of sentences. Our annotations allow several variations for a particular predicate, covering all possible styles of predicates produced by ORE methods.

Result inspection. A more affordable evaluation method is to manually inspect the relations extracted by a system [5, 43, 98, 99]. The effort necessary to inspect a system's output is much lower than inspecting the whole corpus. However, this method can only measure precision. ReVerb [28] use this method on multiple systems and measure their recall by using as ground truth the union of all extracted instances deemed as correct. It is worth noting that this measure is not true recall, but rather an upper bound on true recall. In addition, the recall measured this way cannot be compared by independent studies.

Database as ground truth. An automatic method for evaluating a open RE system is to use a database as ground truth [1, 65, 69]. By restricting the evaluation to the relations found in the database, one may measure the precision of a system with a comprehensive database. Occasionally, the precision may be affected by relation instances that are described in the corpus that are not true anymore (e.g., "Pluto is a planet"). A greater problem concerns measuring recall. Because a corpus often describe fewer instances than the database, using the database as ground truth may over-penalize a system. As discussed in Chapter 3, this problem can be mitigated by evaluating only the entity pairs that appear in both the system's extracted instances and the database. While this approach cannot assess true recall, it can provide an estimation of precision.

Chapter 3

SONEX: A Clustering-Based Approach

We started our pursuit for an effective and efficient ORE method by extending a clustering-based approach proposed by Hasegawa et. al. [43]. Our system, SONEX, identifies entities and extract sentences that relate such entities, followed by using text clustering algorithms to identify the relations within the information network. We propose a new term weighting scheme that significantly improves on the state-of-the-art in the task of relation extraction both when used in conjunction with the standard $tf \cdot idf$ scheme, and also when used as a pruning filter. We describe an effective method for creating ORE benchmarks for that relies on a curated online database that is comparable to the hand-crafted evaluation datasets used in the literature. From this benchmark we derive a much larger dataset which mimics realistic conditions for the task of ORE. We report on extensive experiments on both datasets which shed light not only on the accuracy levels achieved by ORE tools but also on how to tune such tools for better results.

The work presented in this chapter appeared at the AAAI International Conference on Weblogs and Social Media (ICWSM'10), Data Challenge Workshop [65], the ACM Transactions on the Web Journal [62], the ACM SIGMOD/PODS Ph.D. Symposium [63] and the NIST Text Analysis Conference [67].

Entity 1	**Context**	**Entity 2**
⟨Barack Obama, `PER`⟩	and vice presidential running mate and his running mate Sen. received a fundraising bump after he named	⟨Joe Biden, `PER`⟩
⟨John McCain, `PER`⟩	running mate has chosen as his running mate apparently even didn't bother Googling	⟨Sarah Palin, `PER`⟩

Table 3.1: Entity pairs from the Spinn3r dataset.

3.1 Overview

We assume a set E of unique *entities* in the network. Each entity is represented as a $\langle name, \mathit{type}\rangle$-pair. We assume a set T of *entity types*, which are usually automatically assigned to each recognized entity; in our work we consider the types `PER` (for Person), `ORG` (Organization), `LOC` (Location) or `MISC` (Miscellaneous).

An edge (l, e_1, e_2) in the network represents a *relation instance* (i.e., a relationship or event) with label (i.e., predicate) l involving entities e_1, e_2, such as

$$r = (\text{opponent}, \langle\text{Barack Obama}, \texttt{PER}\rangle, \langle\text{John McCain}, \texttt{PER}\rangle).$$

The *domain* of an instance is defined by the types of the entities in it; for instance, the domain of r above is `PER`–`PER`. A *relation* consists of the set of all edges (i.e., instances) that have the same label. We call a relation *homogeneous* if all its instances have the same domain. Finally, a *network* consists of a set of entities and a set of instances involving such entities.

Identifying a relation instance (if one exists) involving entities e_1, e_2 is done by analyzing the sentences that mention e_1 and e_2 together. An *entity pair* is defined by two entities e_1 and e_2 together with the *context* in which they co-occur. For our purposes, the context can be any textual feature that allows the identification of the relation for the given pair. As an illustration, Table 3.1 shows two entity pairs and their context. In this case, the context consists of the exact text *in between* the entities from all sentences in the corpus where they are mentioned together. As we discuss later, we actually employ standard Information Retrieval techniques to *extract* the context from the text in the sentences. Like with an instance, the *domain* of a pair consists of the types of the entities in that pair. The *popularity* (or support) of an entity pair is defined by the number of sentences connecting the two entities.

Problem Definition. We can now define our problem more precisely. Given a collection of documents (blog posts in the work described here), the problem addressed by SONEX is to extract a flat information network (recall Section 1.2) containing all relation instances described in the collection. This problem has been termed Open Relation Extraction (ORE) in the literature [6].

Challenges. Many challenges exist in developing an ORE solution. First, recognizing and disambiguating entities in a multi-document setting remains a difficult task [48]. Second, the unsupervised nature of the problem means that one has to identify all relations from the text only. This is done by identifying *relational* terms in the sentences connecting pairs of entities. Relational terms are words (usually one or two) that describe a relation (for instance, terms like "running mate", "opponent", "governor of" are relational terms, while "fundraising" and "Googling" are not). Finally, another massive challenge is that of evaluating the resulting relations extracted by the ORE system: as discussed further below, the state-of-the-art in the literature relies on small-scale benchmarks and/or manual evaluation. However, neither approach applies to the domain we address (blogs).

It is worth mentioning that, besides the technical challenges mentioned above, there are other practical issues that must be overcome if one wants to deploy any ORE system in the blogosphere. For instance, often, bloggers copy text from each other, leaving a high number of duplicate content. This, in turn, introduces considerable bias in the final network extracted by the system. One common solution is to work on distinct sentences. Also, most algorithms involved in ORE are computationally intensive, and considerable engineering is required to arrive at practical systems.

Outline and Contributions. Figure 3.1 and Figure 3.2 show the steps of our ORE solution, SONEX. Figure 3.1 illustrates the workflow that analyzes blog posts to produce sentences annotated with named entities. For instance, this workflow produces sentences like

"⟨Obama, `PER`⟩ meets ⟨Pope Benedict, `PER`⟩ at the ⟨Vatican, `LOC`⟩"

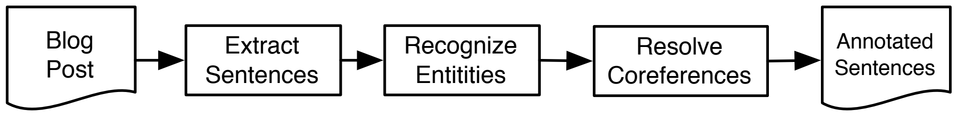


Figure 3.1: Workflow for extracting annotated sentences from the actual blog posts.

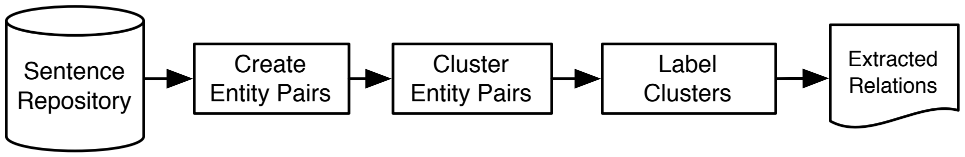


Figure 3.2: The workflow for relation extraction.

where each entity represented as a $\langle name, type\rangle$-pair. This workflow uses off-the-shelf tools as discussed in Section 3.1.1. These annotated sentences are the input for the relation extraction workflow, which is illustrated in Figure 3.2 and discussed in the following sections. This workflow starts by creating entity pairs and extracting their context (Section 3.1.2). For instance, our example sentence contains the entity pair: (⟨Obama, `PER`⟩, ⟨Pope Benedict, `PER`⟩). The context of this pair includes the word "meets" (from our example) and any other word appearing in between this pair from any sentence in the corpus. After extracting entity pairs, SONEX uses a clustering algorithm to group pairs with similar context together (Section 3.1.3). Finally, SONEX finds a representative term (usually one or two words) from each cluster and assign it as the relation label. (Section 3.1.4)

We deployed SONEX on the ICWSM 2009 Spinn3r corpus of weblog posts [13], focusing on posts in English (25 million out of 44 million in total), collected between August 1st, 2008 and October 1st, 2008. The total size of the corpus is 142 GB (uncompressed). It spans a number of big news events (e.g., 2008 Olympics, US presidential election, the beginning of the financial crisis) as well as everything else one might expect to find in blog posts. SONEX runs in a distributed fashion, lending itself as a highly scalable solution: using 10 commodity desktop PCs, we were able to extract entity pairs from 10 million blog posts per day.

SONEX builds on state-of-the-art text clustering methods to group the entity pairs into (un-labeled) relations. We tested various algorithms, using different textual features for the context for the entity pairs. We observed that the customary

$tf \cdot idf$ weighting scheme is often sub-optimal in the task of relation extraction as it does not take into account the context in which a relation is defined. Thus, we use a novel weighting scheme that combines $tf \cdot idf$ with what we call the *domain frequency* (df), which exploits semantic information about the relations being extracted. We show that our new weighting scheme outperforms the state of the art.

As for the evaluation, we developed a method that exploits a curated database (Freebase in this work) to generate a benchmark, specific for the Spinn3r corpus, suitable to evaluate the output of SONEX. Our resulting benchmark is comparable in size to the best hand-crafted ones described in the literature, but is (of course) restricted to the entity pairs that appear in the intersection of the curated database and the Spinn3r corpus. We complement this fully unsupervised evaluation with a manual evaluation considering several thousands of possible pairs, to assess the performance of SONEX on a more realistic scenario.

In summary, our contributions are as follows:

- We present the first large-scale study on using a text clustering-based approach to ORE on the blogosphere, indicating promising results;
- We introduce a novel weighting scheme that outperforms the classical $tf \cdot idf$ in the task of relation extraction;
- We develop a fully automated and rigorous method for testing the accuracy of relation extraction system, tailored to a specific corpus.

3.1.1 Annotating Sentences with Named Entities

The first step in SONEX is processing the blog posts in the corpus to obtain *annotated* sentences in which named entities are mentioned. From these sentences, we construct the entity pairs which are then clustered during the relation identification (discussed in next section). Figure 3.1 illustrates the workflow for extracting the annotated sentences from the blog posts. The process starts with the identification of sentence boundaries (using LingPipe[1]), followed by a conversion of each sentence into plain (ASCII) text for easier manipulation. (In the process, HTML tags

[1] `http://alias-i.com/lingpipe`

and entities referring to special characters and punctuation marks are dealt with); this is accomplished with the Apache Commons library[2] and Unicode characters are converted into ASCII using the LVG component of the SPECIALIST library[3]).

The second step consists of identifying entities in each sentence and assigning their types. For this, we use the LBJ Tagger[4], a NER system [76]. LBJ relies on the so-called BILOU scheme: the classifiers are trained to recognize the Beginning, the Inside, the Outside and the Last tokens of multi-token entities as well as single token (Unit-length) entities. It has been shown that this approach outperforms the more widely used BIO scheme [76], which recognizes the Beginning, the Inside and the Outside of an entity name only. LBJ assigns one of four types (`PER`, `ORG`, `LOC` or `MISC`) to each entity it identifies.

The final step is to identify names that refer to the same real-world entity. This is accomplished using a *coreference* resolution tool to group these names together. In this work, we used Orthomatcher from the GATE framework[5], which has been shown experimentally to yield very high precision (0.96) and recall (0.93) on news stories [10]. Observe that the coreference resolution is performed for entities within a blog post only.

Architecture. SONEX comprises three independent modules (Figure 3.3): Server, Entity Extractor and Relation Extractor. The Server and the Entity Extractor implement the workflow in Figure 3.1 as follows. The Server fires multiple threads for reading blog posts from the corpus, sending such posts to one Entity Extractor process, and collecting the results from all Entity Extractors, storing them in a local database of annotated sentences. Each Entity Extractor fires a number of threads to process the blog posts from the post queue (usually, we set the number of threads to match the number of cores in the host machine). Each thread performs the entire workflow of Figure 3.1 on a single post. Annotated sentences produced by each thread are stored in the sentence queue and eventually sent back to the Server for

[2]`http://commons.apache.org/lang/`
[3]`http://lexsrv3.nlm.nih.gov/SPECIALIST/`
[4]`http://l2r.cs.uiuc.edu/~cogcomp/software.php`
[5]`http://gate.ac.uk/`

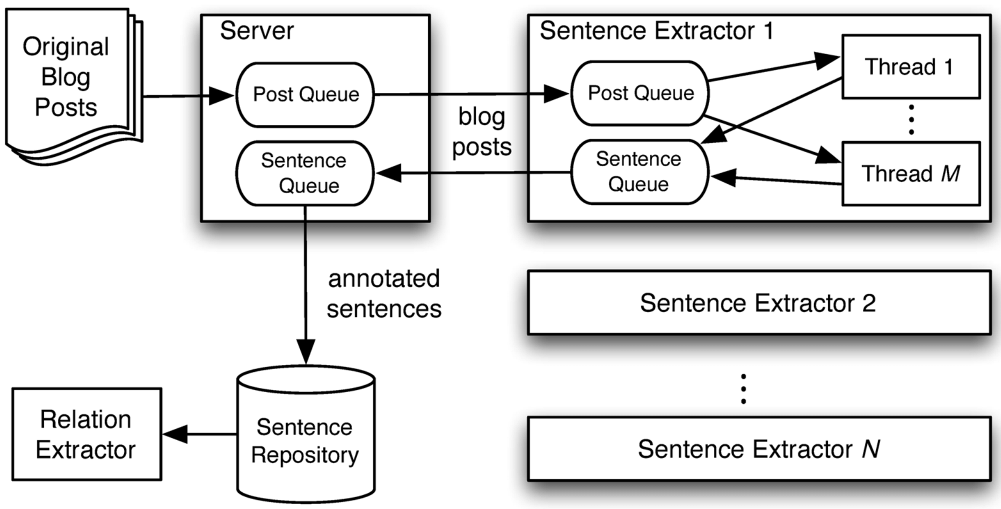


Figure 3.3: SONEX architecture. The Server sends blog posts to Entity Extractors, which parse the posts and send them back to the server to be saved in a database. The Relation Extractor takes parsed posts from the database and identifies relations between entities.

storage. We do not store sentences with less than two entities since they are not used to create entity pairs. In our current implementation, we use Berkeley DB[6] as the back-end engine for storing the annotated sentences.

Handling duplicate sententences. To avoid storing duplicate sentences, we ignore sentences whose MD5 [77] signature collides with the signature of a stored sentence. We found that 20% (around 10 million) of the sentences containing two or more entities were duplicates.

3.1.2 Creating Entity Pairs

Figure 3.2 illustrates the process of relation identification per se, which is done once all annotated sentences are extracted from the corpus. The first step is to build the entity pairs from the repository of extracted sentences. To accomplish this, we implemented a filtering step that allows us to choose which sentences to be considered for the analysis. For the experiments reported here, we used two filtering criteria: (1) the number of words separating the entities in the sentence,

[6]`http://www.oracle.com/technetwork/database/berkeleydb/.`

which we fix to no longer than 5 as suggested by previous work [43]; and (2) the *support* for the entity pair, defined as the number of sentences that contain the entity pair, which we vary in different experiments as discussed later. Once the sentences are filtered, building the entity pairs consists of extracting the textual features used for clustering, as discussed below.

Representing Entity Pairs. Following [43], we adopt the Vector Space Model (VSM) to represent the context of the entity pairs. That is, we collect the intervening features between a pair of entities for each co-occurrence in the entire dataset, constructing the context vector of the pair. Every pair is represented by a single vector. Our ultimate goal is to cluster entity pairs that belong to the same relation. Regardless of the clustering algorithm in use, the feature space plays an essential role. SONEX currently can use any of the the following features:

- **Unigrams**: The basic feature space containing all stemmed [46] single words in the context of a entity pair, excluding stop words.

- **Bigrams**: Many relations may be better described by more than one word (e.g., Vice President). For this reason, we include word bigrams, that is, two words that appear in sequence.

- **Part of Speech Patterns (POS)**: Banko and Etzioni claim that many binary relations in English are expressed using a compact set of relation-independent linguistics patterns [6]. We assume that a context sentence contains one relation at most. Hence, using the Stanford POS Tagger [88], we extract one of the predefined part of speech patterns listed in Table 3.2 from sentences. If a context sentence contains more than one pattern, only the highest ranked one is extracted. We ranked the patterns according to their frequency on sentences as estimated by previous work [6].

In building the vectors, we remove all stop words. We consider a feature to be a stop word only if all of its terms appear in the stop words list (e.g., “capital of” is not removed since it contains one term that is not a stop word).

Rank	PoS Pattern	Example
1	to+Verb	to acquire
2	Verb+Prep	acquired by
3	Noun+Prep	acquisition of
4	Verb	offered
5	Noun	deal

Table 3.2: Ranked part of speech patterns used by SONEX.

3.1.3 Clustering Entity Pairs

We use Hierarchical Agglomerative Clustering (HAC) to cluster entity pairs. HAC is a good option for our task since it does not require the number of clusters in advance. Also, it is used by [43] and was reported to outperform K-Means for our task [80, 98]. The HAC algorithm starts by placing each entity pair in a distinct cluster and produces a hierarchy of cluster by successively merging clusters with highest similarity. In our experiments, we cut this hierarchy at a pre-determined level of similarity by defining a *clustering threshold*. For example, if the clustering threshold is 0.5, we stop the clustering process when the highest similarity between two clusters is below or equal 0.5.

To measure the similarity between two clusters, we compared the single, complete, and average link approaches. Single link considers only the similarity between the closest two entity pairs from distinct clusters, while complete link considers the furthest ones. The average link considers the average similarity between all entity pairs from distinct clusters [36, 56].

3.1.4 Labeling Clusters

The last phase is to label every cluster with a descriptive name. Following the state-of-the-art in this area [35, 89], SONEX uses information from the cluster itself to extract candidate labels as follows:

- **Centroid**: The centroid of each cluster (arithmetic mean for each dimension over all the points in the cluster) is computed, and then the feature with the largest mean value is selected as the cluster's label.

- **Standard Deviation (SDEV):** A disadvantage of the centroid method is that the mean can be too biased towards one pair. To mitigate this problem, we propose to penalize terms with large standard deviation among the cluster's pairs. In this method, the feature to be selected as the label is the one that maximizes the value of the mean divided by its standard deviation among all the pairs within a cluster.

3.2 Weighting Schemes Used in SONEX

As discussed earlier, the contexts of entity pairs are represented using the vector space model. The state-of-the-art in text clustering assigns weights to the terms according to the standard $tf \cdot idf$ scheme. More precisely, for each term t in the context of an entity pair, tf is the frequency of the term in the context, while

$$idf = \log\left(\frac{|D|}{|d : t \in d|}\right), \tag{3.1}$$

where $|D|$ is the total number of entity pairs, and $|d : t \in d|$ is the number of entity pairs containing term t. The standard cosine similarity is used to compute the similarity between context vectors during clustering.

Intuitively, the justification for using idf is that a term appearing in many documents (i.e., many contexts in our setting) would not be a good *discriminator* [78], and thus should weigh proportionally less than other, more *rare* terms. For the task of relation extraction however, we are interested specifically in terms that describe relations. Note that in our settings a document is a context vector of one entity pair, which means that the fewer pairs a term appears in, the higher idf score it would have. Consequently, it is not necessarily the case that terms that are associated with high idf weights would be good relation discriminators. On the other hand, popular relational terms that apply to many entity pairs would have relatively lower idf weights. To overcome this limitation, we use a new weight that accounts for the relative discriminative power of a term within a given relation domain, as discussed next.

Term	IDF	DF (ORG–ORG)	Term	IDF	DF (ORG–ORG)
ubiquitious	11.6	1.00	blogs	6.4	0.14
sale	5.9	0.80	services	5.9	0.13
parent	6.8	0.78	instead	4.0	0.12
uploader	10.5	0.66	free	5.0	0.12
purchase	6.3	0.62	similar	5.7	0.12
add	6.1	0.33	recently	4.2	0.12
traffic	7.0	0.55	disappointing	8.2	0.12
downloader	10.9	0.50	dominate	6.4	0.11
dailymotion	9.5	0.50	hosted	5.6	0.10
bought	5.2	0.49	hmmm	9.3	0.10
buying	5.8	0.47	giant	5.4	< 0.1
integrated	7.3	0.44	various	5.7	< 0.1
partnership	6.7	0.42	revealed	5.2	< 0.1
pipped	8.9	0.37	experiencing	7.7	< 0.1
embedded	7.6	0.36	fifth	6.5	< 0.1
add	6.1	0.33	implication	8.5	< 0.1
acquired	5.6	0.33	owner	6.0	< 0.1
channel	6.3	0.28	corporate	6.4	< 0.1
web	5.8	0.26	comments	5.2	< 0.1
video	4.9	0.24	according	4.5	< 0.1
sellout	9,2	0.23	resources	6.9	< 0.1
revenues	8.6	0.21	grounds	7.8	< 0.1
account	6.0	0.18	poked	6.9	< 0.1
evading	9.8	0.16	**belongs**	6.2	< 0.1
eclipsed	7.8	0.16	authors	7.4	< 0.1
company	4.7	0.15	hooked	7.1	< 0.1

Table 3.3: Unigram features for the pair *Youtube[ORG] – Google[ORG]* with IDF and DF (ORG–ORG) scores

3.2.1 The Domain Frequency

It is natural to expect that the relations extracted in SONEX are strongly correlated with a given context. For instance, marriage is a relation between two persons and thus belongs to the domain `PER`–`PER`. We exploit this observation to boost the weight of relational terms associated with marriage (e.g., “wife”, “spouse”, etc.) in those clusters where the domain is also `PER`–`PER`. We do it by computing a *domain frequency* (df) score for every term. The more dominant a term in a given domain compared to other domains, the higher its df score would be.

We start with a motivating example before diving into the details about how we compute *domain frequency*. We initially built SONEX with the traditional $tf \cdot idf$

and were unsatisfied with the results. Consequently, we examined the data to find a better way to score terms and filter noise. For example, we noticed that the pair *Youtube[ORG] – Google[ORG]* (associated with the "Acquired by" relation) was not clustered correctly. In Table 3.3 we listed all the Unigram features we extracted for the pair from the entire collection sorted by their domain frequency score for ORG–ORG (recall that these are the intervening features between the pair for each co-occurrence in the entire dataset). For clarity the terms were not stemmed.

Clearly, most terms are irrelevant which make it difficult to cluster the pair correctly. We listed in bold all terms that we think are useful. Besides "belongs", all these terms have high domain frequency scores. However, most of these terms do not have high IDF scores. Term frequencies within a pair are also not helpful in many cases since many pairs are mentioned only a few times in the text. Next, we define the domain frequency score.

Definition. Let P be the set of entity pairs, let T be the set of all entity types, and let $D = T \times T$ be the set of all possible relation domains. The *domain frequency* (df) of a term t, appearing in the context of some entity pair in P, in a given relation domain $i \in D$, denoted $df_i(t)$, is defined as

$$df_i(t) = \frac{f_i(t)}{\sum_{1 \leq j \leq n} f_j(t)}, \tag{3.2}$$

where $f_i(t)$ is the frequency with which term t appears in the context of entity pairs of domain $i \in D$, and n is the number of domains in D.

Specificity of the df. Unlike the idf score, which is a *global* measure of the discriminating power of a term, the df score is domain-specific. Thus, intuitively, the df score would favour specific relational terms (e.g., "wife" which is specific to personal relations) as opposed to generic ones (e.g., "member of" which applies to several domains). To validate this hypothesis, we computed the df scores of several relational terms found in the clusters produced by SONEX on the main Spinn3r corpus (details in the next section).

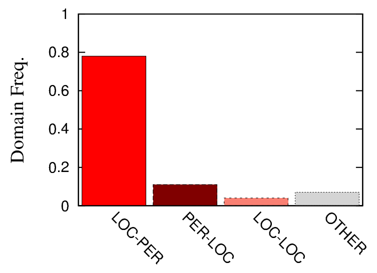
Figure 3.4 shows the relative df scores of 8 relational terms (**mayor**, **wife**, **CEO**, **acquire**, **capital**, **headquarters**, **coach**, and **author**) which illustrate well the strengths of the *df* score. We can see that for the majority of terms (Figure 3.4(a)–(f)), there is a single domain for which the term has a clearly dominant *df* score: `LOC–PER` for **mayor**, `PER–PER` for **wife**, `ORG–PER` for **CEO**, etc.

Dependency on NER Types. Looking again at Figure 3.4, there are two cases in which the *df* score does not seem to discriminate a reasonable domain. For **coach**, the dominant domain is `LOC–PER`, which can be explained by the common use of the city (or state) name as a proxy for a team as in the sentence "Syracuse football coach Greg Robinson". Note, however, that the problem in this case is the difficulty for the NER to determine that "Syracuse" refers to the university. These are some examples of correctly identified pairs in the **coach** relation but in which the NER types are misleading:

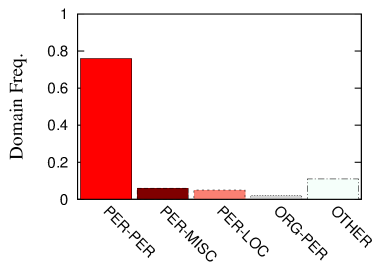
- `LOC–PER` domain: (England, Fabio Capello); (Croatia, Slaven Bilic); (Sunderland, Roy Keane).
- `MISC–PER` domain: (Titans, Jeff Fisher); (Jets, Eric Mangini); (Texans, Gary Kubiak).

This problem is compounded further for the case of **author**, as book titles (or part of them) are often proper names of places, persons, organizations and other kinds of entities, making the task of type identification extremely difficult. Some examples from our experiments are:

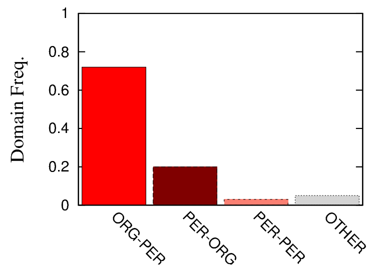
- `PER–PER` domain: (Eoin Colfer, Artemis Fowl); (J.K. Rowling, Harry Potter)
- `PER–ORG` domain: (Donna Cutting, The Celebrity Experience); (Adam Smith, The Wealth of Nations)
- `PER–LOC` domain: (George Orwell, Animal Farm); (Cormac Mccarthy, The Road)



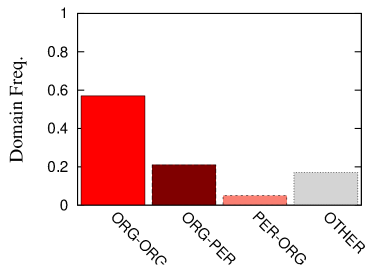
(a) **mayor**.



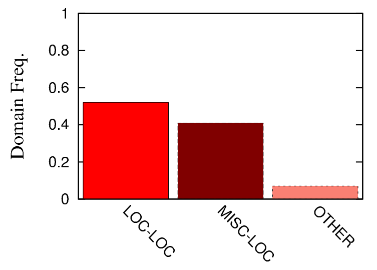
(b) **wife**.



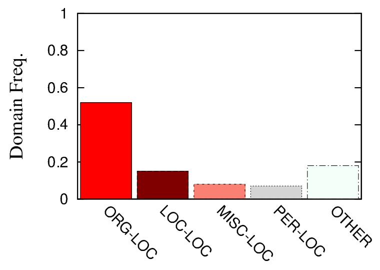
(c) **CEO**.



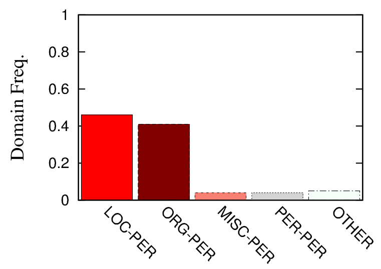
(d) **acquire**.



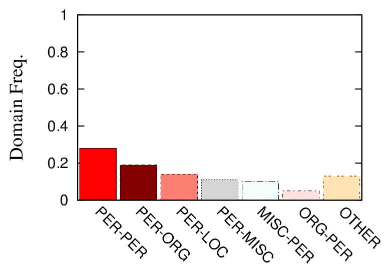
(e) **capital**.



(f) **headquarters**.



(g) **coach**.



(h) **author**.

Figure 3.4: Domain Frequency examples.

3.2.2 Using the df Score

We use df score for two purposes in our work. First, for clustering, we compute the weights of the terms inside all vectors using the product $tf{\cdot}idf \cdot df$. Second, we also use the df score as a filtering tool, by removing terms from vectors whenever their df scores lower than a threshold. Going back to the *Youtube[ORG] – Google[ORG]* example in Table 3.3, we can see that minimum df filtering helps with removing many noisy terms. We also use maximum IDF filtering which helps with removing terms that have high df scores only because they are rare and appear only within one domain (e.g., ubiquitious (misspelled in source) and uploader in this example).

As we shall see in the experimental evaluation, even in the presence of incorrect type assignments made by the NER tool, the use of df scores improves the accuracy of SONEX. It is also worth mentioning that computing the df scores can be done fairly efficiently, and as soon as all entity pairs are extracted.

3.3 Setup of Experimental Evaluation

Evaluating ORE systems is a difficult problem, especially in the scale with which we employ SONEX, namely, the blogosphere. To the best of our knowledge, no public benchmark exists for the task of information extraction from informal text such as those often found in social media sites. Furthermore, existing relation extraction benchmarks, such as ACE RDC 2003 and 2004 (recall Section 2), are built from news corpora, whose texts are produced and revised by professional writers and journalists, and, clearly, do not represent the challenges of the task on blogs. Given the lack of benchmarks, the current evaluation approaches rely on manual evaluations (e.g., [43, 80]), whose main limitation is that they do not scale. In fact, it is not even clear whether a manual evaluation through crowd-sourcing (e.g., using Mechanical Turk) would be feasible given that ORE systems such as SONEX extract hundreds of thousands of relation instances from millions of blog posts.

A different approach to evaluating an information extraction system is to rely on an existing database as the ground truth [48]. This approach, often employed in constrained information extraction settings usually focusing on a specific domain, has the main advantage that it allows for an automatic (and objective) evaluation. However, one disadvantage of this approach is that precision and recall must always be evaluated against the relation instances that lie in the *intersection* between the corpus and the reference database.

Our approach. We combine the two methods above in our work. We build a *reference dataset* for automatic evaluation by automatically matching entity pairs in our clustering task against a publicly available curated database. We call the resulting dataset INTER (for intersection) from now on. From INTER, we derive a clean ground truth against which we verify by hand. We build a larger dataset by adding approximately 30,000 entity pairs from our original set into INTER, to study the accuracy of our system in a more realistic scenario. We call this second database NOISY.

The 30,000 entity pairs in NOISY represent approximately 30% of the total number of extracted entity pairs. We initially created five different samples representing 10%, 20%, 30%, 40% and 50% of all extracted entity pairs. We got significantly more features with 30% than with 10% and 20%, but only a few more with 40% and 50%. In addition, we observed that the results for 40% and 50% are similar to those for 30%. Hence, our NOISY dataset is likely to be a representative sample of all entity pairs while requiring significantly less processing time.

We evaluate SONEX by reporting precision, recall, and f-measure numbers for our system running on INTER and NOISY against the ground truth, in a variety of settings. We also report the manual evaluation (conducted by volunteers) of samples of the instances identified by SONEX but which are outside of the ground truth. Finally, we report the semantic similarity between the labels identified by SONEX and those in the ground truth.

3.3.1 Building the Ground Truth

We build the ground truth by automatically matching the entity pairs in our task against a publicly-available, curated database. For the results reported, we used

Relation	**Freebase Types**	**Domain**	**# Pairs**
Capital	Country – City/Town	`LOC–LOC`	77
Governor	State – Governor	`LOC–PER`	66
Marriage	Person Name – Person Name	`PER–PER`	42
Athlete Representing	Olym. athlete – Country	`PER–LOC`	40
Written work	Author – Work written	`PER–MISC`	26
Headquarters	Company – City/Town	`ORG–LOC`	21
President	Country – President	`LOC–PER`	20
Prime Minister	Country – Prime Minister	`LOC–PER`	18
City Mayor	City/Town – Mayor	`LOC–PER`	15
Company Founded	Company Founder – Company	`ORG–PER`	12
Acquired by	Company – Company	`ORG–ORG`	11
Films Produced	Film Producer – Film	`PER–MISC`	11
House Speaker	US House of Represent. – Speaker	`ORG–PER`	7
Album by	Musical Artist – Musical Album	`PER–MISC`	6
Single by (song)	Musical Artist – Musical Track	`PER–MISC`	6
Football Head Coach	Football Head Coach – Footb. Team	`ORG–PER`	5
Products	Company – Product	`ORG–MISC`	4
Basketball Coach	Basketball Coach – Basket. Team	`ORG–PER`	3
Vice President	Country – Vice President	`LOC–PER`	3
Bishop	City/Town – Bishop	`LOC–PER`	2
Total			395

Table 3.4: Relations in the ground truth. The column **Freebase Types** shows the types assigned by Freebase, while the column **Domain** shows the closest types that can be inferred by our NER system.

Freebase[7], a collaborative online database maintained by an active community of users. At the time of writing, Freebase contained over 12 million interconnected *topics*, most of which correspond to entities in our terminology. Entities in Freebase are connected through *properties*, which correspond to relations. For example, "Microsoft" is connected to "Bill Gates" through the property "founders".

Choosing Relations for the Ground Truth. To identify which relations were described by the Spinn3r dataset, we picked three samples of 1,000 entity pairs each. The first sample contains pairs whose support is greater than 300 sentences; the second contains pairs whose support is between 100 and 300 sentences, while the third sample contains pairs whose support is between 10 and 100 sentences. We matched[8] every entity in this sample against the topics in Freebase. Our ground

[7]`http://www.freebase.com`
[8]Using exact string matching.

Support Level	**Number of Pairs**
≥ 10	395
≥ 15	300
≥ 20	247
≥ 25	214
≥ 30	176
≥ 35	147
≥ 40	133

Table 3.5: Cumulative distribution of number of pairs as a function of support for the INTER dataset.

truth then consists of those pairs of topics from Freebase that match entities in our sample *and* are connected both in Freebase (through a property) and in our sample (by forming an entity pair). We clean the resulting set of entity pairs by standardizing the NER types for all entities which are automatically extracted, hence having all relations *homogenous* as a result.

Table 3.4 shows the relations in our ground truth and their respective domains and cardinalities.

3.3.2 Discussion

As outlined above, we test SONEX on two datasets: INTER, which consists of the pairs in the intersection between Freebase and entity pairs extracted from the Spinn3r corpus, and NOISY, which consists of INTER augmented with approximately 30,000 more entity pairs derived from Spinn3r.

The INTER dataset poses, in many ways, similar challenges to those used in the state-of-the-art in ORE for evaluation purposes. Two significant differences are that INTER contains many more relations than in other works that rely on manual evaluation (e.g., [43] use only two relations), and that INTER contains many entity pairs whose support is lower than the minimum support used in previous works. Both [43] and [80] set the minimum support for clustering at 30 sentences, under the justification that this yields better results. (We confirm this observation experimentally in Section 3.4.4). Instead of 30, we set the minimum support for entity pairs in INTER at 10 sentences. Table 3.5 shows a cumulative distribution of the number of pairs for various levels of support in INTER.

While INTER reproduces the experimental conditions as in a manual evaluation, it is hardly a representative of the realistic conditions that would be faced by any practical ORE system designed for the blogosphere. We design NOISY with the intent of testing SONEX on a more challenging scenario, by adding thousands of entity pairs that make the clustering task much harder, serving, in a sense, as "noise". Others have used the same approach, but at a much smaller scale: [80] added 800 "noise" pairs into a ground truth of 200 pairs, while we add approximately 30,000 entity pairs into a ground truth of 395 pairs.

It is important to note that by this ground truth we built we do not attempt to evaluate the absolute true recall/precision. The problem with a true precision/recall evaluation is this: we can only find the intersection of what the system produces and what is in the reference database (Freebase in our case), but this does not give true precision (as there are many correctly extracted instances which are not in Freebase), nor true recall (as there are instances in the database which are not in the corpus–and hence could never be extracted in the first place). For recall, the problem is particularly worse because it does not matter how much of Freebase is extracted by the system, what really matters is how much of Freebase is actually in the corpus. To find that out, however, we need a perfect extraction system, as one cannot build a gold-standard manually on 25M blog posts. We are confident to say that if we did build such a true recall evaluation set from Freebase, the recall of the output of any Open IE system on Spinn3r would be extremely low, as the Spinn3r data were crawled during a two month period.

3.3.3 Metrics

We measure the similarity between the *relations* extracted by SONEX and the relations defined in our ground truth, using precision, recall and their harmonic mean, the f(1)-measure [57]. When evaluating clusters, high precision is achieved when most pairs that are clustered together by the ORE system indeed belong to the same relation in the ground truth. Conversely, high recall occurs when most of the pairs that belong to the same relation in the ground truth are clustered together. As customary, we interpret f-measure as a proxy for "accuracy" in our discussion.

More precisely, we define two sets S, F containing pairs of entity pairs that belong to the same relation in the output of SONEX and in the ground truth, respectively:

$$S = \{(p, q) \mid p \neq q, \text{ and } p \text{ and } q \text{ are clustered together by SONEX}\}$$
$$F = \{(p, q) \mid p \neq q, \text{ and } p \text{ and } q \text{ belong to the same relation in the ground truth}\}$$

With these, we define:

$$\text{precision} = \frac{|S \cap F|}{|S|}, \ \text{recall} = \frac{|S \cap F|}{|F|}, \ \text{and f-measure} = \frac{2 \cdot P \cdot R}{P + R}.$$

3.4 Results on the INTER dataset

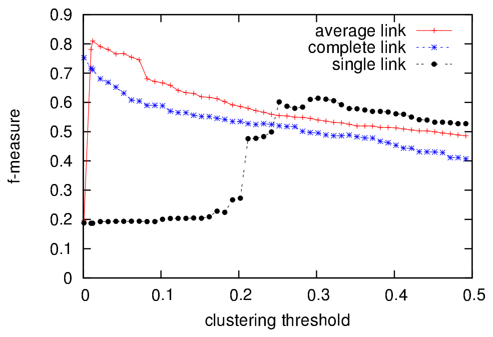
We now report the results on INTER. The first experiment we performed concerned identifying the best settings of the clustering algorithm, as well as the best textual features for clustering. The second experiment studied the impact of pruning terms from the contexts according to their weights (using idf and df scores).

Terminology. For clarity, we will refer to the clustering threshold (recall Section 3.1.3) as τ in the sequel.

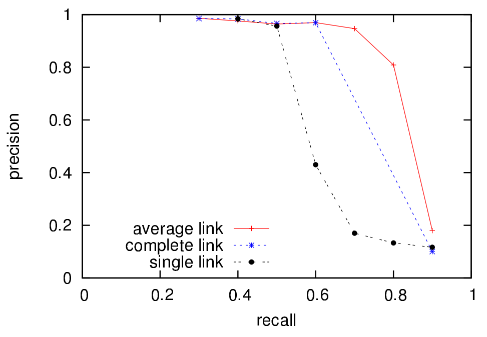
3.4.1 Comparison of Clustering Methods

Figure 3.5(a) shows the quality of the clusters produced by three different clustering approaches: single, complete, and average link, for $0 \leq \tau \leq 0.5$ (we omit results for $\tau > 0.5$ as the accuracy consistently decreased for all methods in this scenario). In these tests, we use unigrams to build the vectors, and $tf \cdot idf$ as the weighting scheme. The Cosine similarity is used throughout all of the experiments.

The behaviour of the single link approach was as follows. For $0 \leq \tau \leq 0.2$, this approach yields few relations but with many pairs in them, resulting in high recall but low precision. When $\tau \approx 0.3$, we observed a large improvement in precision at the expense of recall, yielding the best f-measure for this method. However, for $\tau > 0.3$, the decrease in recall is more significant than the increase in precision; consequently, the f-measure value drops.



(a) Clustering methods.



(b) Precision vs recall.

Figure 3.5: Comparison of clustering methods on INTER (features: unigrams, weights: $tf \cdot idf$).

Method	P	R	F1	τ
Single link	0.96	0.46	0.61	0.3
Complete link	0.97	0.62	0.75	0.001
Average link	0.80	0.82	0.81	0.012

Table 3.6: Results for single, complete and average link method when using the best threshold for each of them.

The behaviour of both average and complete link are much easier to characterize. Complete link yields fairly high precision at the expense of recall even for very small values of τ; further, recall drops consistently as τ increases, without any noticeable increase in precision. Average link truly serves as a compromise between the two other methods. As with single link, we observed precision increasing with τ grows, but the best f-score is achieved with a much smaller threshold ($\tau \approx 0.01$). We also noticed a drop in recall as τ grows for average link; however, this drop is not nearly as severe as with complete link.

Figure 3.5(b) sheds more light on how each method trades precision and recall. The graph shows the maximum precision for different levels of recall (ranging from 0 to 1 in 0.1 intervals). We observe that all three methods present high precision for recall below 0.45. For single link, the precision quickly decreases for recall values approaching 0.5, while complete and average link still maintain high precision for these recall values. However, average link is able to maintain better recall for higher precision levels than complete link. Recall values above 0.9 are only achieved when all pairs are grouped into a few big clusters, which leads to poor precision values

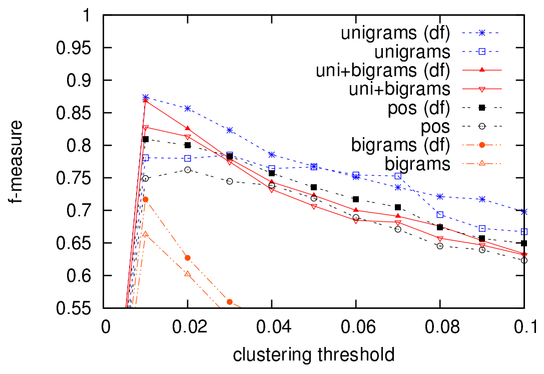


Figure 3.6: Comparison of textual features on INTER, *tf·idf* Vs. *tf·idf·df* (clustering method: average link).

below 0.2; this is common to all methods.

Table 3.6 shows the best results of each method in our first experiment. Overall, the highest accuracy of all methods is achieved by average link (0.81), outperforming both complete link (0.75) and single link (0.61). For this reason, we used average link as the clustering method for all other experiments we conducted. It is worth mentioning that while we show only the results obtained with unigrams as the clustering feature, we observed the same behaviour with the other features as well.

3.4.2 Comparison of Clustering Features

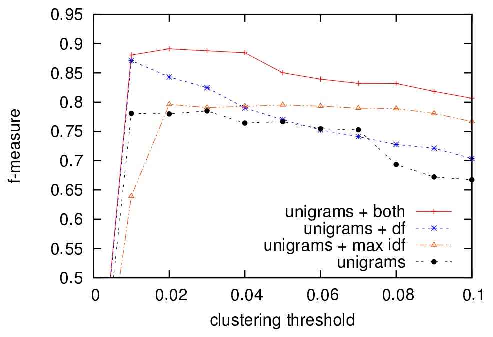
Figure 3.6 shows the performance of the different features implemented by SONEX (recall Section 3.1.2) when using the standard $tf \cdot idf$ weighting scheme compared to $tf \cdot idf \cdot df$.

Several observations are possible from this graph. First, all features performed well, except for bigrams alone. Second, the combination unigrams+bigrams performs the best overall both when $tf \cdot idf$ alone is used (f-score of 0.82), as well as when df is also used (f-score of 0.87). The part of speech (POS) feature is slightly outperformed by the combination unigrams+bigrams (which, as a matter of fact, subsumed the POS features in our tests). Finally, the use of df increases the f-measure with all features, sometimes substantially (the highest increase was close to 12% in the case of unigrams).

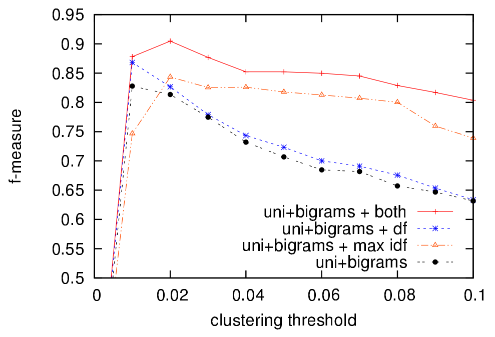
A closer analysis on the impact of the *df* score revealed that it helps most in cases when a given pair's context vector includes proportionally more non-relational terms with high *idf* as opposed to actual relational terms, thus confirming our intuition for the usefulness of this weight. In general, the majority of mis-clustered pairs were those whose contexts contained little useful information about their relationship. For example, the most useful text connecting the pair (⟨AARP, `ORG`⟩, ⟨Washington, `LOC`⟩), belonging to the **headquarters** relation in the ground truth, were "convention in", "gathering this morning in", "poverty event in", and "conference in". Worse still, some entity pairs do not have any context once one removes stop words. Finally, one reason for the poor results produced by using bigrams in isolation is low recall, caused by its inability to extract relations from entity pairs with only one word among them, as in the sentence "⟨Delaware, `LOC`⟩ senator ⟨Joe Biden, `PER`⟩".

It is interesting that the POS feature performed lower than Unigram+Bigrams. The results show that while the best run obtained high precision (0.92), its recall value is significantly lower than the best results achieved by the Unigram+Bigrams feature (0.64 Vs. 0.80). Low recall is expected in rule-based systems. Consider for example the sentence *"Carmen Electra and ex Dave Navarro were..."*. The term *"ex"* is important for this pair but the tagger tags it as a foreign word and we cannot extract it using our POS patterns. If we could train the Stanford tagger on large annotated web text, we maybe could have improved its accuracy on the Spinn3r collection. However, even then, we have other issues such as sparsity and variability of the relations. For example, we extracted the pair (⟨Eamon Sullivan, `PER`⟩, ⟨Australia, `LOC`⟩) from Freebase. This pairs belongs to the "Athlete Representing" relation. This is a difficult relation since many pairs do not include the explicit relation such as "PER representing LOC". Consider for example: *"Eamon Sullivan takes silver for Australia"*. The phrase *"takes silver for"* is the only context we extracted for this pair, and the extracted POS pattern is *"silver for"*. This will not match the POS pattern extracted from the pair (⟨Shawn Johnson, `PER`⟩, ⟨US, `LOC`⟩) in the sentence *"Shawn Johnson won silver putting the US ..."*

Since the best features were unigrams in isolation and the combination uni-



(a) Feature: unigrams.



(b) Feature: uni+bigrams.

Figure 3.7: Comparison of weight-based pruning on different features on INTER using average link

Feature	**Clustering Method**	**Max** *idf*	**Min** *df*	**f-score**
Unigrams	avg. link; $\tau = 0.02$	5	—	0.79
	avg. link; $\tau = 0.01$	—	0.4	0.87
	avg. link; $\tau = 0.02$	5	0.4	0.89
Uni+Bigrams	avg. link; $\tau = 0.02$	5	—	0.84
	avg. link; $\tau = 0.01$	—	0.4	0.86
	avg. link; $\tau = 0.02$	5	0.4	0.90

Table 3.7: Summary of results on INTER.

grams+bigrams, we only use them in our last experiment with INTER.

3.4.3 Effectiveness of Pruning by Weight

Now, we show the effect of pruning terms from the contexts of entity pairs according to their idf and df scores. We use a *maximum* idf threshold to filter out terms that appear in the context of too few entity pairs. Conversely, we use a *minimum* df threshold to prune terms *within* a given context only (in other words, a term with low df score on a given domain may still be used in the context of another entity pair, from a different domain). We experimented with each filter in isolation, and found empirically that the best results were achieved when the maximum idf threshold was set to 5, and the minimum df threshold was set to 0.4.

Figures 3.7(a) shows the effects of each pruning criterion in isolation, and in combination for when using unigrams only. A similar plot for unigrams+bigrams is shown in Figure 3.7(b). A general trend is clear in the two scenarios: both pruning

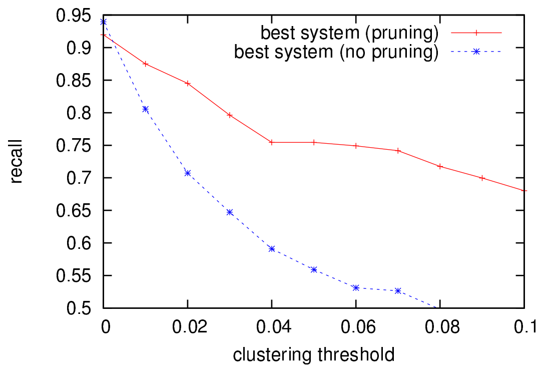


Figure 3.8: Recall "with" and "without" feature pruning

methods improve accuracy in a wide range of values for τ. The best f-score (0.89 for unigrams, and 0.90 for unigrams+bigrams) is achieved when both pruning strategies are used. Table 3.7 shows the best overall results on INTER we achieved.

One important issue to consider is that aggressive pruning can have a negative effect on recall. Figure 3.8 shows the best version of the system with Vs. without feature pruning (refers to Figure 3.7(b) "uni+bigrams + both" Vs. "uni+bigrams"). There is a negative effect on recall in threshold zero (pruning makes a few extra pairs "empty" of context), but we can see that recall drops faster without pruning. Interestingly, we see that pruning can also have a positive effect on recall; not only precision. The main reason is that Noisy features increase the number of candidate clusters a pair can be merged with, which makes low precision clusters, but also increases the number of clusters that contain a specific relation; this has a negative effect on recall, as recall is maximized when all the pairs belonging to a specific relation are in the same cluster.

3.4.4 Effectiveness of Pruning by Support

We also assessed the impact of pruning pairs by their support on the accuracy of the results, motivated by the fact that, in a sense, the support of a pair (i.e., the number of sentences for that pair) can be used as crude measure of "popularity" for that pair. We partitioned our evaluation pairs into three groups, according to their support:

- *high*, with support greater than 39 sentences;

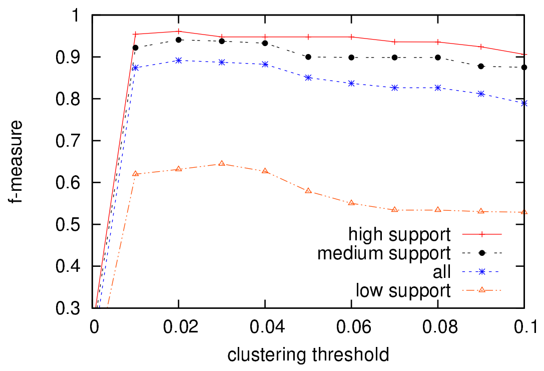


Figure 3.9: Effect of support on accuracy in INTER using average link, with unigrams and pruning by df and idf

- *medium*, with support between 18 and 39 sentences; and

- *low*, with support less than 18 sentences.

This partition yields three roughly equi-sized subsets: *high* has 133 pairs, *medium* has 132 pairs, and *low* has 130 pairs.

Figure 3.9 shows the accuracy levels for the different partitions in INTER. (To facilitate the reading of the results, accuracy, in this experiment, is measured relative to the pairs in the given partition.) The graph shows a direct correlation between support and accuracy. This is expected, because the higher the support, the higher the number of terms in the contexts (which, in turn, allows for better clustering). There is a marked difference in accuracy levels when contrasting low-support with medium-support, and medium-support with high-support, indicating that the relationship between support and accuracy is not linear.

3.4.5 Pruning by In-degree

Another interesting feature to look at in blogs, which is part of the metadata of the Spinn3r collection and related to the previous experiment on support, is the effect of the number of in-coming links on performance. Figure 3.10 (a) shows the in-degree distribution of the sites in the Spinn3r collection. We can see that the majority of the sites have a low in-degree (there are $8,989,885$ sites with in-degree of zero). It is worth mentioning that in this experiment we excluded sites that were assigned an in-degree of “-1” in the collection (i.e., unknown). There are two interesting questions to ask:

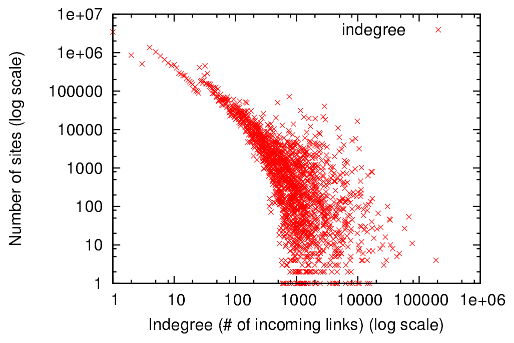
1. Do the more “popular” blogs provide more reliable content in a way that performance (accuracy or speed) can be improved?
2. Sites with extremely high in-degree can be sometimes spam; can we improve SONEX performance by excluding such sites?

Figure 3.10 (b) shows the performance of SONEX (INTER, unigrams, no pruning) on different subsets of the blogs based on in-degree values. The top 10% and top 20% of sites contain most of the ground truth pairs but not all of them; to avoid low recall due to missing pairs, we removed such pairs from the ground truth of these two subsets. The results show that top 10% and top 20% achieved the lowest scores among the subsets. This bolsters the results from Figure 3.9 that there is a correlation between “support” of a pair and performance. Interestingly, the run on top 50% achieved only slightly lower results than the best run. This shows that the top 50% provides enough support for successfully cluster the pairs, which is useful since this subset is significantly smaller than the entire collection and processing it is significantly faster. We also see the effect of excluding the top sites by indegree. Excluding the top 10% and top 20% does not affect the performance; however, excluding the top 50% sites hurts the performance quite a bit, which is not surprising given the results achieved by using only the top 50% subset.

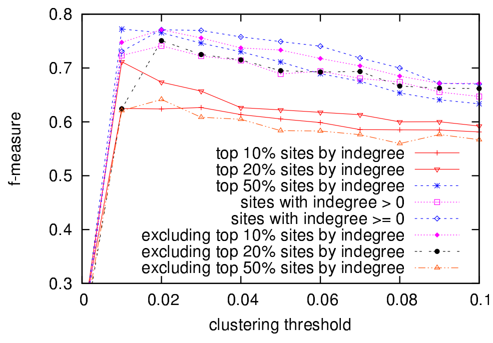
3.4.6 Summary of Observations on INTER

We can summarize our findings on the experiments on INTER as follows:

- **Clustering Method**.
  - Average link consistently outperformed single and complete link in terms of f-measure, as shown in Figure 3.5(a);
  - Complete link produced the fewest incorrect instances (i.e., achieved highest precision), as shown in Table 3.6;



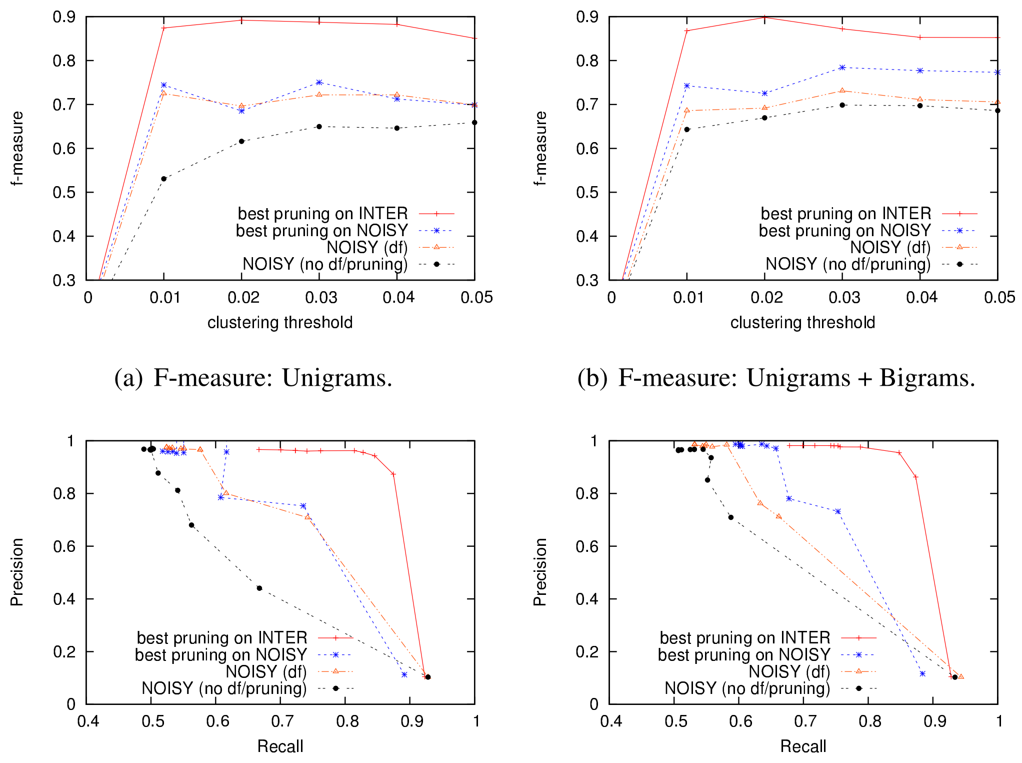
(a) **distribution**.



(b) **performance**.

Figure 3.10: In-degree: The number of incoming links.

- **Features**.
  - The best clustering features are unigrams and the combination of unigrams+bigrams, as shown in Figure 3.6;
- **Weighting**.
  - Using $tf \cdot idf \cdot df$ (instead of $tf \cdot idf$ alone) increased accuracy up to 12%, as shown in Figure 3.6;
- **Pruning**.
  - pruning terms by maximum idf and minimum df improves accuracy substantially, as shown in Figures 3.7(a) and 3.7(b);
- **Results**.
  - F-measure values as high as 0.9 were achieved using average link on the combination unigrams+bigrams, with $\tau \approx 0.02$, and $tf \cdot idf \cdot df$, when entity pairs are pruned by $df \geq 0.4$ and $idf \leq 5$, as shown in Figure 3.7(b). This is an 11% improvement over our baseline setting ($tf \cdot idf$ with unigrams) as proposed by Hasegawa et. al. [43].



(a) F-measure: Unigrams.

(b) F-measure: Unigrams + Bigrams.

(c) Precision-Recall: Unigrams.

(d) Precision-Recall: Unigrams + Bigrams.

Figure 3.11: Experiments on NOISY

3.5 Results on the NOISY Dataset

We now report our results on NOISY, contrasting them with those on INTER, in order to highlight the challenges of extracting relations in more realistic conditions. First, we show results of an automatic evaluation of SONEX on NOISY, in which, as customary, we perform the clustering on all pairs in NOISY but report the results on only those pairs known to be in the ground truth. Since average link clustering achieved the best results for NOISY as well, we do not report results for single and complete link. Next, we complement this analysis with results of a manual evaluation on the pairs not in the ground truth, performed by volunteer Computing Science students in our lab.

Dataset	wife	CEO	capital	author	coach	mayor	acquire	HQ
INTER	0.97	0.99	1.00	1.00	0.99	0.99	1.00	0.92
NOISY	0.76	0.92	0.52	0.21	0.41	0.89	0.57	0.52
	PER–PER	ORG–PER	LOC–LOC	PER–MISC	ORG–PER	LOC–PER	ORG–ORG	ORG–LOC

Table 3.8: *df* scores for the dominant domains (INTER vs NOISY).

3.5.1 Comparison of Clustering Features

Figure 3.11 shows a comparison of SONEX's accuracy on NOISY when using two different feature sets: unigrams and unigrams+bigrams. The most striking observation is the substantial decrease of the results on NOISY compared to the best results on INTER. We observed a similar drop for the other feature sets as well (POS and bigrams alone). Such a decrease is, of course, expected: NOISY contains not only thousands more entity pairs than INTER, but also hundreds (if not thousands) more *relations* as well, making the clustering task much harder in practice. Moreover, many context vectors contain terms related to more than one relation because sometimes there is more than one relation between the entities in the pair. Given the approximate nature of text clustering, it is only to be expected that some entity pairs which are clustered together on INTER will be clustered separately on a larger set of entity pairs. In the INTER dataset this is not a problem since the total number of relations is small, and the pair is likely to end up in a cluster representing its strongest relation.

Another challenge when dealing with NOISY is that it contains, as expected, considerably more NER mistakes, thus affecting the effectiveness of our *df* score. While on INTER we can expect to find homogeneous relations since we manually corrected all NER type mistakes for every entity pair, on NOISY, this is virtually hopeless. Now, all domains are inferred from the types assigned by the NER tool; as a result, all *df* scores decrease. For example, the *df* score for **wife** in the PER-PER domain drops from 0.97 on INTER to 0.76 on NOISY. Table 3.8 lists a comparison between the *df* scores generated using the INTER and NOISY datasets. Nevertheless, Figure 3.11 shows that using the minimum *df* pruning strategy still yields considerably better results than performing no pruning.

Relation	Domain	# Pairs (INTER)	Incorrect Types	Precision
Capital	`LOC–LOC`	182 (64)	29 (16%)	0.87
Governor	`LOC–PER`	139 (63)	0	0.72
Athlete	`PER–LOC`	7 (4)	0	0.71
Marriage	`PER–PER`	129 (13)	1 (0.8%)	0.77
Written work	`PER–MISC`	105 (17)	60 (57%)	0.84
Headquarters	`ORG–LOC`	28 (11)	0	0.86
President	`LOC–PER`	245 (18)	84 (34%)	0.64
Prime Minister	`LOC–PER`	134 (17)	51 (38%)	0.67
City Mayor	`LOC–PER`	31 (14)	0	0.94
Company Founded	`ORG–PER`	98 (9)	9 (9%)	0.47
Total		1098 (230)	234 (21%)	0.75

Table 3.9: Manual evaluation of the clusters of the 10 largest relations.

3.5.2 Manual Evaluation of Clusters

We now report on a manual evaluation of SONEX on entity pairs which are not in INTER (nor in the ground truth). Since manual evaluation is expensive and error-prone, we restrict this exercise to samples of the 10 largest clusters (each corresponding to each of the 10 largest relations in our ground truth–recall Table 3.4) found by SONEX. The evaluation was performed by 8 volunteers in our lab. Each volunteer was given a relation label, and a list of entity pairs. We report only precision since recall is unknown, which in this case indicates the fraction of entity pairs in each sample that, in the volunteer's opinion, truly belongs to the given relation. The results shown here correspond to clusters produced using the settings that produced the best results (recall Figure 3.11).

Table 3.9 shows the precision results obtained in this evaluation. Overall, 75% of the 1098 entity pairs evaluated were deemed correct by the volunteers. However, the precision of individual relations varied greatly (ranging from 0.47 for "Company Founded" to 0.94 for "City Mayor"). The "Company Founded" cluster contains many Company – CEO pairs that do not belong to the "Company Founded" relation; this yields significantly lower results than average. Since there is a large overlap between founders and CEOs in real life (e.g., CEO(Bill Gates, Microsoft) and Founder(Bill Gates, Microsoft)), a cluster containing "Company Founded" pairs would have context related to both founders and CEOs. Therefore, the CEO portion of the context attracts other pairs that belong solely to the CEO relation (e.g., CEO(Eric Schmidt, Google)). Since the HAC algorithm assigns every pair to only one cluster, it would be able to separate both relations with high precision only if every pair in the intersection of “Company Founded” and “Company CEO” is only mentioned in one of the contexts, which is unrealistic to expect.

Two additional observations are worth mentioning. First, the impact of the large number of entity pairs on identifying relations defined by general terms such as “president” was significant. (Further evidence of this issue is given in the “Athlete Representing” cluster in Table 3.10, explained in the next section.) Second, the fraction of entity pairs where at least one entity is assigned an incorrect type from the NER tool is disproportionately higher for domains involving type `LOC`[9] (4th column in Table 3.9). The majority of the mistakes are `LOC` entities labeled as `MISC`.

3.5.3 Summary of Observations on NOISY

The following general observations can be made from our analysis:

- **Clustering Method.**
  - As with INTER, the best clustering method was average link.
- **Features.**
  - As with INTER, the best clustering features were unigrams and unigrams+bigrams;
- **Weighting.**
  - The *df* score improved the results, despite the high number of incorrect domains due to NER mistakes, as shown in Table 3.8;
- **Pruning.**

[9]A close look at Table 3.9 also shows a large number of errors for the `PER–MISC`, but this is not surprising, as the `MISC` type by definition contains many kinds of entities.

  - Pruning terms by maximum idf and minimum df improves accuracy substantially, as shown in Figures 3.11(a) and 3.11(b);

- **Results.**
  - F-measure values as high as 0.79 were achieved using average link on the combination unigrams+bigrams, using pruning by df and idf, as shown in Figure 3.11.
  - The incorrect type identification by the NER tool is disproportionately higher for locations, as shown in Table 3.9;

- **Manual Evaluation of Clusters.**
  - Precision levels around 75% on average and as high as 94% were achieved, as shown in Table 3.9.

It is worth mentioning that SONEX significantly improved over our baseline settings ($tf \cdot idf$ with unigrams) on both INTER and NOISY.

3.6 Evaluation of Relation Labels

This experiment aims at evaluating the quality of the labels assigned to the clusters by SONEX. As discussed in Section 3.1.4, we use two methods for extracting labels from the contexts of the pairs in the resulting clusters: using the *centroid* term, as in the state-of-the-art, and a variant that smoothes out term weights to avoid outliers, which we call SDEV.

Both the Centroid and SDEV methods select one *stemmed term* from the context of one or more entity pairs in the cluster, and, as such, do not always produce a meaningful label. As an illustration, Table 3.10 shows the top-3 stemmed terms and their mean weights within a cluster for the 4 biggest relations on NOISY, across all feature sets we considered. We obtain a more readable (un-stemmed) label for each cluster based on the frequency of original terms corresponding to a given stem. For example, in our experiments, “capit” becomes “capital” since this is the most frequent term among all terms that stem to “capit”.

Feature	First (weight)	Second (weight)	Third (weight)
Capital cluster			
unigrams	capit (8.0)	citi (4.7)	coast (2.5)
unigrams+bigrams	capit (14.0)	citi (8.1)	citi of (8.0)
bigrams	s capit (13.2)	capit of (12.8)	citi of (12.7)
POS	capit (4.1)	citi of (3.4)	capit of (3.3)
Governor cluster			
unigrams	gov (220.1)	governor (118.0)	govenor (0.8)
unigrams+bigrams	gov (287.3)	governor (149.5)	s governor (11.4)
bigrams	s governor (22.4)	attorney gener (15.7)	s gov (7.1)
POS	gov (95.1)	governor (62.8)	sen (23.4)
Marriage cluster			
unigrams	wife (37.9)	husband (25.3)	hubbi (6.3)
unigrams+bigrams	wife (30.2)	husband (20.0)	hi wife (1.4)
bigrams	s ex (12.8)	father of (4.9)	the father (4.1)
POS	husband (26.5)	boyfriend (8.7)	wife of (6.3)
Athlete Representing cluster			
unigrams	rep (17.5)	convent (17.4)	congressman (5.6)
unigrams+bigrams	secret (2.6)	met (2.41)	met with (1.0)
bigrams	ambassador as (2.6)	the righteous (2.3)	left of (2.3)
POS	left of (2.6)	winningest (2.1)	decor (1.3)

Table 3.10: Top stemmed terms for the 4 biggest relations, represented by the biggest cluster extracted for each relation.

Further Evidence of the Difficulty of Extracting General Relations. In passing, we point to Table 3.10 to return to the difficulty of automatically extracting the "Athlete Representing" relation, connecting athletes and their countries[10], whose main relational term is represent. Because there are many different relations expressed through the verb represent besides being an Olympic athlete (e.g., as an elected official), entity pairs from this relation end up clustered within these other relations *and* vice-versa. Evidence of this is the fact that among the top-3 terms for this cluster we can find terms indicating political representation ("congressman" and "ambassador") as well as other very generic terms ("met with").

3.6.1 Evaluation Method

There is still no standard evaluation methodology for cluster labelling, and there are no standard benchmarks to compare alternative labelling methods [14]. To evaluate

[10]This was a popular topic at the time the Spinn3r data was collected, right after the Beijing Olympic games.

Relation	**Omiotis**			**Manual**		
	Centroid	**SDEV**	**Difference**	**Centroid**	**SDEV**	**Difference**
Capital	5.00	5.00	*0.00*	5.00	4.50	*0.50*
Governor	3.97	2.03	*1.94*	4.42	3.45	*0.97*
Athlete Repr.	1.00	4.76	*3.76*	2.66	4.30	*1.10*
Marriage	3.97	3.97	*0.00*	4.60	4.60	*0.00*
Author	3.92	2.96	*0.96*	4.18	4.89	*0.71*
Headquarters	4.55	1.15	*3.40*	4.47	3.97	*0.50*
President	4.60	4.60	*0.00*	4.52	4.52	*0.00*
Prime Minister	4.89	4.89	*0.00*	3.72	3.72	*0.00*
Mayor	5.00	5.00	*0.00*	4.80	4.80	*0.00*
Founder	5.00	5.00	*0.00*	5.00	4.00	*1.00*
Average	4.19	3.94	*0.25*	4.33	4.27	*0.06*

Table 3.11: INTER labels.

Relation	**Omiotis**			**Manual**		
	Centroid	**SDEV**	**Difference**	**Centroid**	**SDEV**	**Difference**
Capital	4.60	2.91	*1.69*	4.30	2.09	*2.21*
Governor	3.91	2.05	*1.86*	4.17	1.35	*2.82*
Athlete Repr.	1.29	1.88	*0.59*	1.95	2.47	*0.52*
Marriage	3.66	2.90	*0.76*	3.96	4.30	*0.34*
Author	2.86	3.05	*0.19*	3.73	3.75	*0.02*
Headquarters	3.60	3.35	*0.25*	3.20	3.31	*0.11*
President	3.15	2.35	*0.80*	4.58	3.52	*1.06*
Prime Minister	4.56	4.56	*0.00*	3.68	3.70	*0.02*
Mayor	4.73	5.00	*0.27*	4.81	4.81	*0.00*
Founder	3.00	3.00	*0.00*	3.50	3.25	*0.20*
Average	3.54	3.10	*0.64*	3.78	3.25	*0.52*

Table 3.12: NOISY labels.

an extracted label, we resort to the semantic relatedness of the un-stemmed term with the relation label corresponding to the relevant cluster in our ground truth. Semantic relatedness methods are widely used in text mining tasks [34]; we used Omiotis[11], a system for measuring the relatedness between words, based on WordNet[12] [90]. Omiotis reports relatedness in a scale from 1 to 5, where 1 indicates very weak relatedness and 5 indicates very strong relatedness. In addition, we repeat a similar evaluation conducted manually and using the same relatedness scale, for comparison. This manual assessment was done by four Computing Science

[11]`http://omiotis.hua.gr`

[12]`http://wordnet.princeton.edu/`

Annotators	Observed Agreement (By Chance)	Kappa	Weighted Kappa
A1–A2	61.11% (26.92%)	0.46	0.68
A1–A3	53.70% (24.18%)	0.38	0.62
A1–A4	55.56% (23.29%)	0.42	0.63
A2–A3	59.26% (27.91%)	0.43	0.66
A2–A4	50.00% (24.86%)	0.33	0.62
A3–A4	52.94% (24.07%)	0.38	0.65

Table 3.13: Annotators Agreement with Kappa Statistics

students in our lab who did not participate in the previous evaluation.

Table 3.11 presents the average relatedness levels for the Centroid and SDEV methods on the INTER dataset, while Table 3.12 shows results for the clusters produced from the NOISY dataset, after removing entity pairs that are not in the intersection. In consistency with the clustering results, the INTER labels are more accurate than the ones obtained on NOISY. Also, on average, Centroid outperforms SDEV on all evaluations. Overall, the relatedness scores reported by Omiotis are very close to the manually assigned ones, with minor differences ranging from 0.14 to 0.33 on average. No labels show high discrepancy between the Omiotis and manual assessment. The only relation with consistent assessment of low score on NOISY is "Athlete Representing". From our observations we learn that this relation is not often mentioned explicitly in the text which makes it hard to identify using our approach that only looks for clues in text between two entities. Varied results between relations is very common in previous works as well, with systems that use $tf \cdot idf$ only for weighting [43].

Table 3.13 shows the degree of agreement among the annotators using the Kappa statistical measure [33]. The degree of agreement ranges from 50% to 62%, which is significantly better than the degree of agreement by chance. However, these results demonstrate the challenge of finding labels that satisfy different users. We do not have any instance where none of the annotators agree on and also no high discrepancy among annotators (e.g., scores 1 and 5 for the same label). Most disagreements differ by a single score. The "Kappa" column only considers exact matches between annotators. On the other hand, the "Weighted Kappa" column considers the difference between scores (e.g., 4 is closer than 3 when compared

Relation	**INTER**		**NOISY**	
	Centroid	**SDEV**	**Centroid**	**SDEV**
Capital	capital	city	capital	living
Governor	gov	delegation	gov	today
Athlete Representing	won	representing	rep	representative
Marriage	husband	wife	wife	married
Written work	book	wrote	book	book
Headquarters	headquarters	based	headquarters	headquarters
President	president	president	presidential	political
Prime Minister	prime	prime	minister	minister
Mayor	mayor	mayor	mayor	mayor
Founder	founder	cofounder	chairman	head

Table 3.14: Label examples.

with 5).

Example Relation Labels. Table 3.14 shows the highest ranked relation labels extracted by SONEX on both INTER and NOISY for the ten largest clusters.

3.7 Comparison to ReVerb

In the previous sections, we presented an extensive evaluation of SONEX and how it extends the work of Hasegawa et al [43] for extracting relations from the blogosphere. In this section, we perform a comparative experiment between our system and ReVerb, a state-of-the-art self-supervised system [28]. We use the implementation distributed by the authors[13].

One of the challenges of evaluating systems like SONEX and ReVerb is that they outcomes are not the same. ReVerb recognizes instances at sentence level. For example, ReVerb would extract the relation "is opponent of" from the sentence "Obama is opponent of McCain". Each of these sentence-level instances is often evaluated as correct or incorrect by human judges. On the other hand, SONEX extracts relations at corpus level. In order to compare both systems at the same level, we convert ReVerb's instances into corpus-level ones by applying a simple aggregation method proposed by TextRunner [5]. All pairs of entities connected through a specific relation (e.g. "is opponent of") in at least a sentence are grouped together

[13]`http://reverb.cs.washington.edu/`

Systems	**Purity**	**Inv. Purity**
ReVerb	**0.97**	0.22
SONEX	0.96	**0.77**

Table 3.15: Comparison between SONEX and ReVerb.

into one cluster. Observe that this process may produce *overlapping clusters*, i.e., two clusters may share entity pairs.

The evaluation measures used in the previous sections cannot be used to evaluate overlapping clustering. Therefore, we adopt the following measures for this experiment: purity and inverse purity [2]. These measures rely on the precision and recall of a cluster C_i given a relation R_j:

$$\text{precision}(C_i, R_j) = \frac{|C_i \cap R_j|}{|C_i|} \qquad \text{recall}(C_i, R_j) = \text{precision}(R_j, C_i)$$

Purity is computed by taking the weighted average of maximal precision values:

$$\text{purity} = \frac{1}{M} \sum_i \arg\max_j \quad \text{precision}(C_i, R_j) \tag{3.3}$$

where M is the number of clusters. On the other hand, inverse purity focuses on the cluster with maximum recall for each relation:

$$\text{inverse purity} = \frac{1}{N} \sum_j \arg\max_i \quad \text{recall}(C_i, R_j) \tag{3.4}$$

where N is the number of relations. A high purity value means that the produced clusters contain very few undesired entity pairs in each cluster. Moreover, a high inverse purity value means that most entity pairs in a relation can be found in a single cluster.

The INTER dataset was used in this experiment. We provide both SONEX and ReVerb with the same sentences and entities. We configured SONEX with the best setting we found in previous experiments: average link with threshold 0.02, the unigrams+bigrams feature set, the $tf \cdot idf \cdot df$ weighting scheme and pruning on idf and df.

Table 3.15 presents the results for SONEX and ReVerb. Observe that both systems achieved very high purity levels, while SONEX shows a large lead in inverse

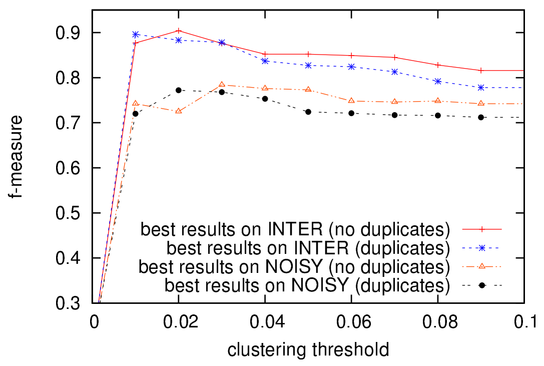


Figure 3.12: Results on Inter and NOISY: Duplicates Vs. No Duplicates

purity. The low inverse purity value for ReVerb is due to its tendency of scattering entity pairs of a relation into small clusters. For example, ReVerb produced clusters such as “is acquired by”, “has been bought by”, “was purchased by” and “was acquired by”, all containing small subsets of the relation “Acquired by”. This phenomenon can be explained by ReVerb’s reliance on sentence-level instances and the variety of ways a relation can be expressed in English. These results show the importance of relation extraction methods that work beyond sentence boundaries, such as SONEX.

3.8 Applying ORE on the Blogosphere

We highlight the main issues we encountered in extracting relations from the blogosphere and how we address them.

1. Duplicate content that skewed the true distribution of terms. Figure 3.12 shows that duplicates indeed hurt performance on both INTER and NOISY; hence, we eliminated duplicates. The problem with duplicates is that they affect the weights assigned to features. The more times a feature appears in a context of a pair, the greater effect it has (the term frequency part). But what we really want to know is the term frequency of the features in different occurrences of the pair (such as different blog posts) rather than counting the same occurrence (same source) multiple times just because the text was duplicated. This also affects the domain frequency and IDF weights.

2. Misspellings and noise in general. We used a filtering on the *idf* score to get rid of noisy terms (e.g., "ubiquitious" and "hmmm" in Table 3.3)

3. The performance of the NER tool was shown to be lower for blog posts than for news articles (see [76]). The filtering based on domain frequency helped us to get rid of many misclassified or wrong entities (i.e., wrong boundary). For example, the NER created the pair ($\langle$Ted Mcginley, `PER`$\rangle$, $\langle$Children, `ORG`$\rangle$) from the sentence "Ted Mcginley from Married with Children". Since both the entity type and boundary are wrong, it is better to exclude this pair. The only feature we extracted for this pair is "married". Since the domain frequency for "married" in the domain PER–ORG is extremely small, this feature was filtered and consequently the pair was filtered for having no features. Overall, we filtered 3336 pairs (approximately 11% of the NOISY dataset) using the filterings on *idf* and *df* in the experiments with the NOISY dataset.

These are all blog related, although not unique only to blogs, and are relevant to the web at large.

3.9 SONEX 2.0

This section presents SONEX 2.0, an improved version of our system whose goal is to increase efficiency and effectiveness. We discuss some of the limitations of SONEX 1.0 and explain how SONEX 2.0 overcome these limitations. This work appeared at the ACM SIGMOD/PODS Ph.D. Symposium [63] and the NIST Text Analysis Conference [67].

3.9.1 Improving Clustering

In order to cluster context vectors, SONEX 1.0 uses the Hierarchical Agglomerative Clustering (HAC) algorithm [57]. One of the advantages of HAC over many other clustering algorithms is that it does not require the number of clusters as a parameter for stopping the clustering. The user can instead provide a similarity threshold. The clustering stops when the similarity of the clusters being merged is below the threshold. Not requiring the number of clusters is a desirable property since one can seldom estimate the number of relations described in a collection *a priori*. On the other hand, HAC does not scale well for a large number of vectors. This is because the time and space complexity for HAC is at least quadratic in the number of vectors [57]. Therefore, SONEX 1.0 can be infeasible when the number of context vectors is massive. In addition, HAC is not capable of extracting more than one relation for a pair of entities. This make SONEX a poor choice for pairs that present multiple relations expressed in the corpus.

To address both limitations, SONEX 2.0 adopts the buckshot clustering algorithm [24]. Buckshot addresses HAC's quadratic complexity by applying HAC over a *sample* of the vectors, instead of all vectors. By clustering a sample size $\sqrt{N}$, where N is the total number of vectors, HAC can find cluster centroids in linear time and space. It is well-accepted that the centroids of clusters produced from a *representative sample* are often as good as the centroids of the clusters produced from all vectors [24]. To better handle entity pairs with multiple relations, buckshot is able to assign a vector to multiple clusters as long as the similarity between the vector and the cluster centroids is above a threshold.

The steps of the buckshot approach are as follows:

1. **Choose a sample**. The original buckshot approach recommends choosing a random sample of size $\sqrt{N}$. This approach works well when entity pairs are required to present a minimum support (i.e., number of sentences where they appear together). However, when all entity pairs are taken into consideration (regardless of their support), we observed that the random approach chooses many vectors containing only a few non-zero features. Since these vectors do not provide much variations on how a particular relation is expressed, many duplicate clusters are likely to be produced — one for each variation (e.g., "acquired", "acquisition of"). In this scenario, a user may ask SONEX to choose vectors with the highest l^2-norm instead.

2. **Apply HAC.** SONEX 2.0 applies HAC over the sample of vectors and produce a number of clusters, according to the clustering similarity threshold.

Description	Pattern	Example
close apposition	$A?\ N\ ,?$	Apple new CEO Cook
apposition+preposition	$,\ A?\ N\ P\ A?$	Cook, new CEO of Apple
possessive+noun	$,\ C?\ S\ A?\ N\ P?\ A?\ ,?$	Apple's new CEO, Cook
verb	$(,\ W)?\ A?\ V\ A?$	Cook now leads Apple
verb+preposition	$(,\ W)?\ A?\ V\ A?\ P\ A?$	Apple is led by Cook
verb+noun+preposition	$(,\ W)?\ A?\ V\ A?\ N\ A?\ P\ A?$	Cook is the CEO of Apple

Table 3.16: POS patterns employed by SONEX 2.0. The patterns can contain the following symbols: *A* (sequence with any adjective, adverb, particle, modal and determiner), *N* (sequence of nouns), *V* (sequence of verbs), *P* (preposition), *S* (possessive pronoun, possessive wh-pronoun or possessive ending), *C* (coordinating conjunction) and *W* (Wh-pronoun).

This process is identical of how SONEX 1.0 produces clusters, except that it is applied to a sample as opposed to all vectors.

3. **Assign remaining vectors to one or more clusters**. This step consists in assign each remaining vector (those not in the sample) to one of the clusters produced in the previous step. The original buckshot approach adopts the Assign-To-Nearest method, where the vector is assigned to the cluster whose centroid is the most similar to the vector. In order to allow for an entity pair to have multiple relations, SONEX also allow the user to opt for assigning a vector to every cluster whose similarity between vector and the cluster's centroid is above a threshold.

3.9.2 Improving Feature Extraction

SONEX 2.0 also improves the quality of the features extracted from individual sentences. As our experiments show, unigrams and bigrams can be quite effective for clustering similar context vectors; however, they are poor choices for describing long, more specific relations (e.g., "is chief operational officer of", "is honorary vice chairman for").

Our system employs an improved set of POS patterns to extract relational terms for an entity pair. These terms are then used as features for the clustering. Our improved patterns are much more selective about the terms they can extract, eliminating noisy features and improving the quality of the cluster labels.

Table 5.1 shows six POS patterns used by SONEX 2.0 to extract relational terms for an entity pair. Given all entity pairs within a window of X tokens ($X = 10$ in our current implementation), our method tries to match one of the POS patterns with the text in between them. If our method is able to find a match, it concatenates verbs, nouns and prepositions to produce a term.

Finally, SONEX 2.0 applies 4 filtering rules to discard non-relational terms. A term is discarded when:

1. The entity after the term is followed by a noun or possessive ending ('s).
2. The entity after the term is preceded by the preposition "that".
3. The text between the entities contain the word "say".
4. The entity before the term is preceded by the preposition "of".

Hearst patterns. SONEX 2.0 also improves the extraction of features by removing Hearst patterns, a set of patterns discovered by Hearst [44] that describe entity types. For an example, consider the sentence:

> "**Blackberry** is losing market share to *tech companies such as **Apple** and **Google***".

where the phrase in italics matches one of the Hearst patterns: entity type + "such as" + list of entities. Phrases like this one can introduce noisy terms (e.g., "tech companies") in between the entities of a relation instance (e.g., "Blackberry" and "Google"). To eliminate this problem, SONEX 2.0 transforms every sentence containing one of these phrases as follows. Each sentence is duplicated into several sentences, one for each listed entity, and the phrase matching a Hearst pattern is replaced by one of the listed entities. For instance, the following sentences would be generated from our example:

> "**Blackberry** is losing market share to **Apple**",
> "**Blackberry** is losing market share to **Google**".

SONEX 2.0 applies the same transformation for sentences that do not present Hearst patterns but contain a phrase listing three or more entities. For example, consider the sentence:

"**Blackberry** is losing market share to ***Apple, Google*** *and* ***Microsoft***",

where the text in italics lists three named entities. This sentence would be transformed into three sentences:

"**Blackberry** is losing market share to **Apple**",
"**Blackberry** is losing market share to **Google**",
"**Blackberry** is losing market share to **Microsoft**".

3.9.3 Experimental Results

We discuss the efficiency and effectiveness of SONEX 2.0 by evaluating the new clustering algorithm and the improved feature extraction separately. Chapter 6 discusses an experiment with several ORE methods and evaluates a new method using SONEX 2.0's improved patterns to extract relation instances (i.e., without clustering). Our experiment shows that our improved patterns are competitive, outperforming ReVerb in both precision and recall. This method is also more effective and efficient than some methods based on dependency parsing and SRL.

We evaluated the effectiveness of the buckshot algorithm by participating of the Text Analysis Conference (TAC)[14], which is promoted by NIST [67]. Particularly, we submitted SONEX 2.0 to the slot-filling task [67]. This task comprises the identification of values for certain attributes, or *slots*, about an entity. Each named entity is given as a query and the answers are called slot fillers. The evaluation consisted of 100 queries, being 50 people and 50 organizations. Examples of slots are `per:date_of_birth` (a person's date of birth) and `org:subsidiaries` (the child companies of a company). Each slot allows either a single answer (e.g., `per:date_of_birth`) or a list of answers (e.g., `org:subsidiaries`).

SONEX 2.0 extracted and clustered more than three million vectors from the TAC source collection, a number well beyond the capacity of SONEX 1.0. To find cluster centroids, we used a sample of 10,000 entity pairs, which is higher than the suggested 1732 pairs ($\sqrt{3,000,000}$) but still feasible. For this submission, we used

[14]`http://www.nist.gov/tac/`

Slot	Precision
ORG:Number of Employees	0.95
PER:Employee of	0.92
PER:Member of	0.92
ORG:Headquarters	0.90
PER:Place of birth	0.88
...	
PER:Children	0.66
ORG:Subsidiaries	0.65
ORG:Alternate Names	0.56
ORG:Parents	0.55
ORG:Shareholders	0.50
Average	0.79

Table 3.17: The slots in descendants order of precision as measured in the preliminary experiment. For brevity, only the top and bottom five are shown.

unigrams as features and set the clustering threshold to 0.001, since it achieved the best results in our previous experiments with blog posts. We adopted the Assign-To-Nearest method [24], where the assigned cluster is the one that maximizes the similarity between the vector and the cluster centroid. We labelled each cluster with the token in the cluster centroid with the highest weight.

We use the cluster label that SONEX provides to map a cluster to its appropriate slot (if any). To do so, we manually construct a list of valid labels for each slot by observing which clusters generated from the training data correspond to which slots. For example, we defined that a cluster with the label "husband" should be mapped to the `per:spouse` slot.

Preliminary results. We conducted a preliminary experiment to estimate the precision of the clusters produced by SONEX. We randomly selected 50 entity pairs from clusters mapped to slots. We asked four students to verify whether the entity pairs of each slot were supported by any document in source collection. Entity pairs with a supporting document were deemed as correct. Table 3.17 shows the precision for ten slots, being five with highest score and five with the lowest. The average precision for all slots is 0.79.

TAC results. Our submission achieved 36.84% precision and 5.18% recall, while the median values among all submissions were 10.31% and 16.5%, respectively. One could expect a low recall score from SONEX 2.0, given it works better for entity pairs with high support; however, we were surprised by the low precision score in our TAC submission (in light of our preliminary experiment). In order to understand the discrepancy between the results from TAC and our experiment, we looked carefully at the answers not judged as correct. Out of those 84 non-correct answers, 62 were incorrect, 15 were redundant and 7 were inexact. The majority of incorrect answers are from the slots `per:employee_of` (21) and `per:top_members_employees` (13). Analyzing the slot fillers for these two slots, we have found that 38% of the incorrect answer in these slots are due to mistakes when handling entities. For example, the coreference resolver used by SONEX, OrthoMatcher [10], incorrectly determined that "U.N." (United Nations) and "US Navy" refer to the same entity (possibly because the initials of "US Navy" match "U.N."). This coreference decision resulted in a wrong extraction from the text portion "U.N. nuclear watchdog chief Mohamed ElBaradei". In particular, SONEX answered "US Navy" (instead of "U.N.") for the slot `per:employee_of` of the entity "Mohamed ElBaradei". Another major reason for our low results is that most of the evaluated entity pairs had very low support, appearing in only 1 or 2 sentences. Extracting relations for pairs with such a low support is a challenge for any clustering-based method.

3.10 Conclusion

We presented SONEX, an ORE system for extracting information networks from the blogosphere that works by identifying named entities from the text, clustering the sentences that connect those entities, and extracting prominent terms from the clusters to use as labels. We improved effectiveness by introducing the Domain Frequency (df) score, a new term-weighting score designed for identifying relational terms within different domains, and showed that it substantially increases the accuracy of our system in every test we performed. We believe that df can be utilized in various of applications, with the advantage that in practice, for many such applications, the list of terms and scores can be used off-the-shelf with no further effort. Also, the *df* score computation is based on probability (we do not consider the NER to be part of it), and as such, it can be utilized in other languages with similar structure to English.

SONEX 2.0, the new version of our system, addresses two shortcomings of using hierarchical agglomerative clustering for ORE: its quadratic complexity and inability to identify more than one relation per entity pair. By applying the buckshot algorithm, we can cluster entity pairs in linear time and assign an entity pair to multiple clusters. SONEX 2.0 also employs an improved method to extract the context of an entity pair. This method leverages part-of-speech patterns to find a sequence of words that describe a relation.

We reported the first results on large-scale experiments with clustering-based ORE systems on social media text, studying the effect of several parameters for such an approach. We also discussed an automatic way of obtaining high-quality test data from an online curated database. Our experimental evaluation showed textual clustering techniques to be a viable option for building an ORE system. More importantly, our results shed some light on the accuracy of state-the-art extraction tools in this setting, as well as on ways of tuning such systems for higher accuracy.

Chapter 4

Meta-CRF: Extracting Nested Relations

So far, we have discussed SONEX, a system that produces flat information networks. In such networks, named entities are represented by nodes and relations are represented by edges. Despite their effectiveness, flat information networks are somewhat primitive, as they are unable to represent certain interesting, more subtle relations expressed in the text. For example, consider the following sentence:

"Associated Press reported on the Russian attack to Georgia."

Observe that flat information networks cannot naturally express that the Associated Press (AP) *reported on* the attack, since the attack is represented by an instance. Thus, the relation between AP (which is a node in the network) and the attack (which isn't) cannot be directly represented. A relation that accepts instances (in addition to entities) as arguments are referred to as *nested relations* in the Information Extraction literature [18, 19]. This chapter presents an ORE method that extract both flat and nested relations. The work presented here appeared at the AAAI International Conference on Weblogs and Social Media (ICWSM'11) [64].

4.1 Overview

As discussed in Chapter 1, we extend information networks by representing both entities and instances as nodes. In the terminology of knowledge representation, representing an instance as a node is called *reification* [91, 96]. Hence, we refer to these networks as *reified networks*. Figure 1.2(b) in Page 8 shows an example of such a reified network.

Problem Statement. The problem addressed by Meta-CRF is that of accurately extracting a reified network from a text corpus.

Definition 2. *A reified network consists of a set of unique named entities E, a set of predicates P and a set of instances I. An instance $i \in I$ is a triple: (a_1, p, a_2), where $p \in P$ is a predicate and $a_i \in E \cup I$ is an argument, which is either a named entity or an instance. A relation is a set of instances that have the same predicate.*

For an example, consider the sentence "Google's acquisition of YouTube was finalized by the FTC". Instances described in this sentence are:

i_1 = ("Google", "acquisition of", "YouTube"),

i_2 = (i_1, "was finalized by", "FTC").

An instance can be *flat* or *nested*. A flat instance presents two entities as arguments (e.g., i_1) and a nested instance presents one or two instances as arguments (e.g., i_2). A *flat relation* is a set of flat instances that have the same predicate. A *nested relation* is a set of both flat and nested instances that have the same predicate.

Challenges. Extracting reified networks from text presents many challenges. Correctly extracting relation instances described in text is difficult, given the complexity of the English language and the diversity of writing styles [82]. In particular, even with well-written text, as illustrated in detail later, the nested structure of some instances brings problems not found in traditional, "flat" relation extraction. For instance, one must determine whether a relation instance expressed in a sentence concerns an entity or another instance (expressed in the same sentence) containing such entity. This ambiguity undermines a classifier's ability to differentiate between flat and nested instances.

In our work, we follow the seminal approach in TextRunner [6], a state-of-the-art ORE system. Namely, we rely on a supervised method for handling the actual text—we use Conditional Random Fields (CRF), and exploit syntactic features found in parse and dependency trees obtained for the sentences. It should be noted that our goal here is to extend the use of CRF in order to handle both flat and nested instances. In particular, we extend TextRunner's Open-CRF method [6]. If successful, our model could be used within the larger framework in TextRunner, which encompasses both the self-supervision scheme for training the CRF, as well as the post-processing module that further checks the plausibility of the extracted instances.

Contributions. We propose a relation extraction method that applies Conditional Random Fields to extract flat and nested instances seamlessly. We show that the original CRF model in TextRunner, Open-CRF, lacks discerning power to accurately handle both kinds of instances, and provide an explanation of why this is the case. We show the need for relying on more sophisticated syntactic features, and propose a new CRF model, Meta-CRF, that is powerful enough to differentiate between flat and nested instances.

We evaluate Meta-CRF on a sample of the ICWSM Spinn3r dataset [13], a corpus with 25 million weblogs in English. In quantitative terms, our experiments show that Meta-CRF presents substantial improvements over Open-CRF (when both models are trained with the exact same training examples). More precisely, Meta-CRF outperforms Open-CRF considerably in terms of recall, and substantially in terms of accuracy (over 20%). On the other hand, a small loss (3%) is observed in terms of avoiding false-negatives.

4.2 Extracting Flat and Nested Instances

Pre-processing. We process each document from a corpus separately, as follows. First, we identify sentence boundaries using LingPipe[1] and convert each such sentence into plain (ASCII) text for easier manipulation. (In the process, HTML tags and entities referring to special characters and punctuation marks are dealt with);

[1]`http://alias-i.com/lingpipe`

this is accomplished with the Apache Commons library[2] and Unicode characters are converted into ASCII using the LVG component of the SPECIALIST library[3]).

Next, we identify entities in each sentence, using the LBJ Tagger[4], a state-of-the-art named entity recognition (NER) system [76]. LBJ assigns one of four categories (`PER`, `ORG`, `LOC` or `MISC`) to each entity it identifies. The final step is to identify names that refer to the same real-world entity. This is accomplished using a *coreference* resolution tool to group these names together. We used Orthomatcher from the GATE framework[5], which has been shown experimentally to yield very high precision (0.96) and recall (0.93) on news stories [10]. Observe that the coreference resolution is performed for entities within a document only.

Once we process all documents as described above, each sentence is then split into tokens using the OpenNLP library[6]. We explain our approach using the following sentence tokens (separated by white spaces, named entities in bold):

> "The **U.S.** is likely to punish **Moscow** in response to **Russia**'s conflict with **Georgia** ."

In this example, the instances are:

i_1: ("U.S.", "to punish", "Moscow"),

i_2: ("Russia", "conflict with", "Georgia"),

i_3: (i_1, "response to", i_2).

4.2.1 The Algorithm

Our algorithm (Figure 4.1) operates at the argument level, seamlessly considering both entity arguments and instance arguments. This is achieved as follows. On a first pass over the sequence of tokens given as input, we first identify all *mentions* to named entities and add them the set A which keeps a list of candidate arguments found in the text (line 2). In our running example, this first step would result in

[2]`http://commons.apache.org/lang/`
[3]`http://lexsrv3.nlm.nih.gov/SPECIALIST/`
[4]`http://l2r.cs.uiuc.edu/~cogcomp/software.php`
[5]`http://gate.ac.uk/`
[6]`http://opennlp.sourceforge.net`

```
input: Sequence of tokens $T$
output: Set of instances $I$

1  $I \leftarrow \emptyset$;
2  $A \leftarrow$ all entities mentioned in $T$;
3  $C \leftarrow A \times A$ ;              // We consider every pair of candidate arguments
4  foreach $\langle a_1, a_2 \rangle \in C$ do
5      if $a_1$ precedes $a_2$ in $T$ then
6          $p \leftarrow$ Meta-CRF ($T$, $a_1$, $a_2$);
7          if $p$ is defined then
8              $I \leftarrow I \cup \{(a_1, p, a_2)\}$;
9              $a' \leftarrow$ sequence of tokens in $T$ containing $p$, $a_1$ and $a_2$;
               /* Remember the newly found candidate argument              */
10             $C \leftarrow P \cup (\{a'\} \times A) \cup (A \times \{a'\})$;
11             $A \leftarrow A \cup \{a'\}$;
12         end
13     end
14 end
15 return $I$
```

Figure 4.1: Algorithm for finding statements and meta statements.

A ={"U.S.", "Moscow", "Russia", "Georgia"}

The algorithm attempts to find all possible instances involving arguments that appear together in a single sentence (loop 4–14). Thus, for every pair (a_1, a_2) of candidate arguments, such that a_1 precedes a_2 in a sentence we attempt to detect whether they are involved in a relation instance (as detailed in the next section). Meta-CRF returns a predicate p if such a relation instance exists (line 6). If a predicate is returned, the algorithm then (1) adds an instance (a_1, p, a_2) to the set of instances (line 8), and (2) creates a new argument that corresponds to this instance, for future consideration with other arguments already identified (lines 10, 11).

In our example, the first instances to be extracted are:

i_1: ("U.S.", "to punish", "Moscow")

i_2: ("Russia", "conflict with", "Georgia")

Once they are added to both A and P, we have then:

A ={"U.S.", "Moscow", "Russia", "Georgia", "U.S. is likely to punish Moscow", "Russia's conflict with Georgia"}

Thus, the algorithm will eventually attempt to identify relation instances involving other instances, thus producing i_3 above. It can be shown that the algorithm will never consider the same pair of arguments more than once, and thus always terminates.

4.2.2 The Meta-CRF Model

In this section we discuss how to detect a predicate for a pair of arguments in a sentence.

We model predicate detection as a sequence labeling problem — given a input sequence of tokens $\mathbf{x} = x_1, \ldots, x_n$, produce an output sequence of labels $\mathbf{y} = y_1, \ldots, y_n$ from a set of labels. In particular, we consider tokens *in between* two arguments and labels indicating whether a token belongs to a predicate or not. We adopt the BIO encoding, a widely-used technique in natural language processing [47]. This encoding marks the Beginning, Inside and Outside of a phrase; therefore, each token is labeled as B-REL, I-REL or O-REL. Figure 4.2 illustrates the tokens appearing in between "U.S." and "Moscow" and their respective labels. Tokens that should be labelled as B-REL or I-REL are called *relational* tokens.

Our method, called Meta-CRF, is based on conditional random fields (CRF) [51]. CRF is a graphical model that estimates a conditional probability distribution, denoted $p(\mathbf{y}|\mathbf{x})$, over label sequence $\mathbf{y}$ given the token sequence $\mathbf{x}$. The probability of a label be assigned to the i-th token is defined by a vector $\mathbf{f} = \{f^1, f^2, \ldots, f^K\}$ of real-valued *feature functions* of the form $f^k(y_i, y_{i-1}, \mathbf{x}, i)$. Therefore, a feature function can be defined over the current label y_i, the previous label y_{i-1} or any token in $\mathbf{x}$. Examples of feature functions are:

$$\begin{array}{rcl} f^1(y_i, y_{i-1}, \mathbf{x}, i) & = & [[x_i \text{ is an adverb}]].[[y_i = \text{O-REL}]] \\ f^2(y_i, y_{i-1}, \mathbf{x}, i) & = & [[x_i \text{ is a verb}]].[[y_i = \text{B-REL}]]. \\ & & [[y_{i-1} = \text{O-REL}]] \end{array}$$

where the indicator function $[[condition]] = 1$ if *condition* is true and zero otherwise. Each feature function f^k is associated with a weight W_k; therefore, there is

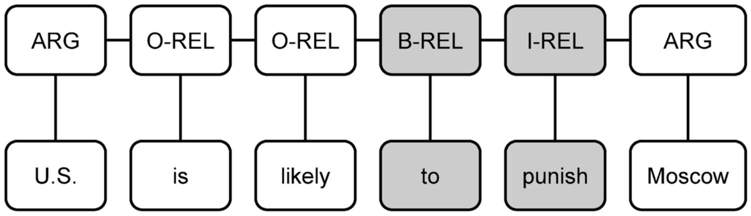


Figure 4.2: A CRF model used to recognize the predicate "to punish" for the instance containing "U.S." and "Moscow".

a weight vector $\mathbf{W} = W_1, \ldots, W_K$ corresponding to $\mathbf{f}$. Finally, we can define the CRF model as follows:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} e^{\mathbf{W}.\mathbf{F}(\mathbf{x},\mathbf{y})} \tag{4.1}$$

where $\mathbf{F}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{|\mathbf{x}|} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}, i)$ and $Z(\mathbf{x})$ is a normalizing constant equal to $\sum_{\mathbf{y}}' e^{\mathbf{W}.\mathbf{F}(\mathbf{x},\mathbf{y}')}$.

Training Meta-CRF consists in learning the weight vector $\mathbf{W}$. This vector defines the likelihood of associating a label to a individual token as well as transitioning from label to label. Meta-CRF uses the CRF implementation provided by the MALLET library [60].

4.2.3 Features

The set of features used by Meta-CRF is similar to those used by state-of-the-art relation extraction systems [47]. We use tokens appearing between arguments, their part of speech, the argument types (statement or entity) and syntactic features from the parse and dependency tree.

Tokens. Following Open-CRF [6], we include function words as features (e.g. prepositions and determiners) but not content words such as verbs or nouns. For example, the tokens used from the sentence "AP reported Russia's conflict with Georgia" are "'s" and "with" only. This is because our method is designed to extract instances of any relation, not a specific one.

Part of speech. Every token is associated with its part of speech. Intuitively, we expect that predicates in English follow a limited number of part-of-speech patterns.

Banko and Etzioni [6] show that 95% the predicates in their dataset follow eight simple part-of-speech patterns. An example is "settlement with", which follows the pattern: noun+preposition. Figure 4.3 presents the tokens (in bold) from the sentence "AP reported Russia's conflict with Georgia" along their part-of-speech tags (in italics).

Argument Type. While Open-CRF assigns the label "ARG" to both arguments, we assign a label that corresponds to the type of the argument ("ENTITY" or "INSTANCE").

Parse tree. Our method uses the path length between a token and each argument in a full parse tree. Intuitively, we expect that the paths between relational tokens and their arguments to be relatively short. The node representing an argument is the lowest common ancestor of all tokens in that argument. Figure 4.3 gives an example parse tree. The arguments "AP", "Russia" and "Russia's conflict with Georgia" are represented by the nodes NP_2, NP_5 and NP_4, respectively. Observe that the path between NP_2 and NP_5 (NP_2–S_1–VP_3–NP_4–NP_5) is longer than the path between NP_2 and NP_4 (NP_2–S_1–VP_3–NP_4), indicating that "AP" and "Russia's conflict with Georgia" are more likely to form an instance than "AP" and "Russia" alone. Our method generates a parse tree for each sentence by using the tools available from OpenNLP[7].

Dependency tree. We also use the path length between a token and each argument in a dependency tree. Intuitively, shorter paths are likely to indicate stronger dependency between the tokens and the arguments. Figure 4.4 illustrates an example dependency tree. An argument is represented by the root of the minimal subtree containing all its tokens. For example, "Russia's conflict with Georgia" is represented by "conflict". Observe that the path between "AP" and "Russia" (AP–reported–conflict–Russia) is longer than the path between "AP" and "Russia's conflict with Georgia" (AP–reported–conflict). Our method produces a dependency tree for each sentence by applying the algorithm from Xia and Palmer [94].

[7]`http://opennlp.sourceforge.net`

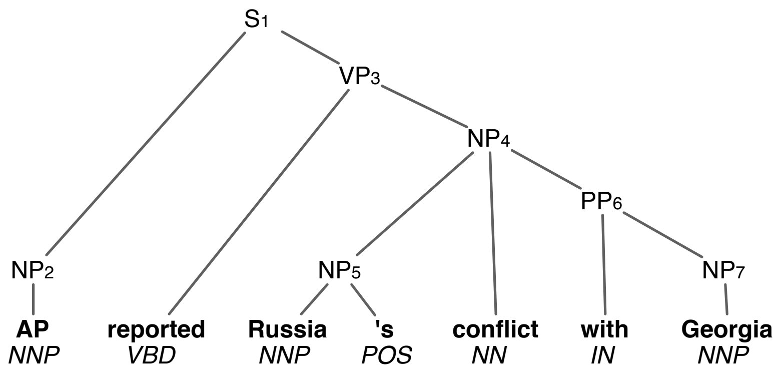


Figure 4.3: Parse tree for the sentence "AP reported Russia's conflict with Georgia" following the Penn TreeBank terminology [7]. Non terminals are numbered for easy identification. Tokens are highlighted in bold and part-of-speech tags are in italic.

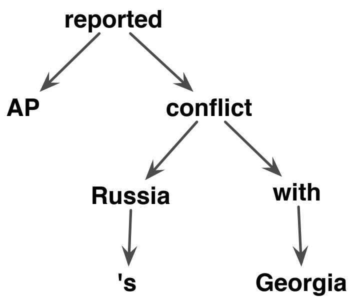


Figure 4.4: Dependency tree for the sentence "AP reported Russia's conflict with Georgia".

4.2.4 The need for syntactic features

State-of-the-art relation extraction systems based on CRF, such as TextRunner's Open-CRF, often rely almost exclusively on tokens and their part-of-speech tags. One problem with this approach is that these features are often insufficient for detecting nested instances in text. To see this, consider the sentence

"The **A.P.** $\underline{\text{reported}}$ **Russia's conflict with Georgia**"

and its parse tree illustrated in Figure 4.3. Observe that "A.P." and "Russia's conflict with Georgia" form an instance of the relation "reported". Furthermore, observe that "A.P." and "Russia" have the word "reported" between them, but there is no instance involving these entities.

In both cases, the part of speech sequence is the same: ARGUMENT $\rightarrow$ VBD $\rightarrow$ ARGUMENT. Therefore, a CRF model has no choice but to assign the same label to "reported" in both cases. No matter the label assigned by the model, this label will be incorrect for at least one of the above argument pairs. This lose-lose situation is very common when dealing with meta statements, since statement arguments will always contain entity arguments.

Our solution for the above problem is to rely on the syntactic structure of a sentence. Parse and dependency trees often provide useful hints to determine whether a sentence describes a relation instance involving two arguments or not. As discussed in Section 4.2.3, we observe that the path between "A.P." and "Russia" is longer than the path between "A.P." and "Russia's conflict with Georgia" in both parse and dependency trees. Our observations are in agreement with a recent study that claims that relations can be extracted almost exclusively from the path between arguments in a dependency tree [12].

4.2.5 Limitations of Meta-CRF

Our method focuses in relation instances whose predicate is explicitly expressed in the text, and not implied by punctuation or structural clues. In addition, our method extracts predicates that appear in the text *between* arguments only. Banko and Etzioni [6] have showed that more than 80% of predicates are found in the text window between arguments, as opposed to the windows before and after the pair of arguments. Moreover, our method focuses on relation instances expressed within a sentence and is not able to extract instances whose arguments cross sentence boundaries. Finally, our method focuses on extracting binary relations (flat or nested) and is not capable of extracting n-ary relations.

4.3 Experiments

In this section we present the results of an experimental evaluation of Meta-CRF. Our method uses all features described in Section 4.2.3. We use as baseline a CRF model that relies on the features used by TextRunner's Open-CRF (tokens and their part of speech).

Setup. Our experiments use sentences from the ICWSM Spinn3r blog dataset [13]. The ICWSM dataset contains 25 million English posts published between August 1st and October 1st, 2008. Popular topics include the 2008 U.S. Election, the Russian conflict with Georgia, the Olympics and the economic crisis. We manually collected a hundred sentences from blog posts containing popular entities in politics (e.g., Barack Obama, John McCain), sports (e.g., Michael Phelps), entertainment (e.g., Paris Hilton) and entities involved in the conflict in Georgia (e.g., Russia, U.S.).

Each collected sentence was used to produce positive (those describing instances) and negative examples (those describing no instances). We produce an example for each pair of arguments in a sentence. For example, the examples produced from the sentence "U.S. condemned Russia's conflict with Georgia" are:

> "**U.S.** condemned **Russia's conflict with Georgia**."
>
> "U.S. condemned **Russia** 's conflict with **Georgia**."
>
> "**U.S.** condemned **Russia** 's conflict with Georgia."
>
> "**U.S.** condemned Russia's conflict with **Georgia**."

where underlined tokens form a predicate and tokens in bold are arguments. Observe that the first example describes a nested instance, the second example describes a flat instance and the last two describe no instances. Producing examples in this way may result in many negative examples, since there are usually many pairs of arguments that do not form an instance. To limit the number of negative examples, we prune out every argument pair where the arguments are separated by more than 5 tokens.

We use both positive and negative examples to evaluate our method. Table 4.1 provides more information about these examples. Our experiments rely on ten-fold cross validation by splitting the examples into ten partitions. In each round, a partition is used for testing while the nine remaining are used for training.

Metrics. The quality of the extraction is measured by the number of tokens correctly labeled. The extraction accuracy is defined as follows.

Unit	Quantity
Original Sentences	100
Examples	496
Nested instances	107
Flat instances	111
No instances	278
Tokens	1245
Relational tokens	364
Non relational tokens	881

Table 4.1: Details about the examples used in experiments. "Original sentences" indicates the sentence collected from the ICWSM dataset, "Examples" are sentences annotated with arguments and predicates (describing nested instances, flat instances and no instances). "Tokens" indicates the number of tokens in all examples. "Relational tokens" indicate tokens labeled as predicates (B-REL, I-REL) and "Non relational tokens" indicate tokens labeled as O-REL.

$$\text{Accuracy} = \frac{\text{Number of Correct Labels}}{\text{Number of Tokens}} \tag{4.2}$$

4.3.1 Comparison between Open-CRF and Meta-CRF

We use Open-CRF as our baseline for comparison as it is the state-of-the-art of CRF-based relation extraction methods. It should be noted that while Open-CRF is not a method for extracting nested instances (nor their authors claim so), this comparison is valuable in that it provides an objective way to assess the impact of using syntactic features when extracting nested instances, as opposed to relying almost completely on part-of-speech tags for this task.

Table 4.2 presents the accuracy of Open-CRF and Meta-CRF in each experimental round. Observe that Meta-CRF improves Open-CRF performance by over 20% on average. In addition, our method consistently outperforms Open-CRF in every round with a minimum improvement of 11.6% and maximum improvement of 34.1%.

Table 4.3 details the performance of Meta-CRF and Open-CRF by reporting their results on examples that describe nested instances, flat instances and no instances in separate. Observe that our method almost tripled the results obtained by Open-CRF when extracting nested instances. Table 4.3 also shows that our method almost

Round	Open-CRF	Meta-CRF	Improvement
1	0.78	0.89	14.4%
2	0.75	0.89	17.7%
3	0.77	0.89	14.6%
4	0.72	0.91	25.6%
5	0.69	0.85	22.7%
6	0.71	0.83	16.3%
7	0.70	0.79	11.6%
8	0.67	0.89	34.1%
9	0.63	0.77	21.5%
10	0.68	0.85	25.0%
Average	0.71	0.86	20.1%

Table 4.2: Results for Open-CRF and Meta-CRF in each round of a tenfold cross-validation evaluation. "Improvement" indicates the relative gain in performance by Meta-CRF over Open-CRF.

doubled Open-CRF performance on examples containing flat instances. This result can be explained by our method's ability to better differentiate direct and nested instances by using structural information as explained in Section 4.2.4. The lack of syntactic information led Open-CRF to label most relational tokens as non relational. An in-depth investigation revealed that Open-CRF was able to correctly label relation tokens only 21% of the time (a metric known as recall), while our method reported 78% at the same task. This is because many examples present the same part-of-speech tag sequence but different labels (recall Section 4.2.3). Open-CRF's inclination to label tokens as O-REL also explains why our method was unable to improve Open-CRF performance at labelling non relational tokens when compared to our method (3.2% drop). Since O-REL comprises the majority of labels in our example set, the Meta-CRF overall improvement (20.1%) was substantially below the improvement in examples containing nested instances (189.7%) and flat instances (82.4%).

4.3.2 Contribution of Individual Features

In this experiment our goal is to study the contribution of individual features to our method's overall performance. Table 4.4 shows the results for our baseline extended with the following features: argument types, dependency tree and parse tree.

	Open-CRF	Meta-CRF	Improvement
Nested instances	0.271	0.785	189.7%
Flat instances	0.4392	0.801	82.4%
No instances	0.9259	0.8965	-3.2%
All examples	0.71	0.86	20.1%

Table 4.3: The performance of Open-CRF and Meta-CRF on average for examples describing nested instances, flat instances and no instances. "Improvement" indicates the relative gain in performance by Meta-CRF over Open-CRF.

Method	Accuracy	Improvement
Open-CRF	0.71	–
+ Argument types	0.82	14.8%
+ Dependency	0.81	14.2%
+ Parse Tree	0.80	12.2%
All Features	0.86	20.1%

Table 4.4: Impact of extending Open-CRF with individual features. "+ Feature" indicates the model Open-CRF extended with "Feature".

Observe that all features combined outperformed individual features. Furthermore, the addition of each individual features produces better accuracy than our baseline.

Another interesting result is that relying on dependency trees yields results as good as those obtained considering argument types alone, which explicitly provide weather an argument is an entity or an instance. This result shows the discriminative power of a dependency tree to differentiate between flat and nested instances.

4.4 Summary

We discussed a method for extracting a reified information network from a corpus. Unlike previous work, we focus on the simultaneous extraction of flat instances as well as nested instances. We proposed Meta-CRF, a CRF-based model that extracts both flat and nested instances seamlessly. Our model extends TextRunner's Open-CRF model by also incorporating syntactic features as found in parse and dependency trees. We showed the need for these syntactic features when dealing with nested instances. Finally, our evaluation reported that Meta-CRF outperforms Open-CRF by as much as 20%.

Discussion. Overall, our Meta-CRF method was able to extract nested instances with 0.86 accuracy. Also, Meta-CRF improved the state-of-the-art by over 20% when extracting both flat and nested instances. These results indicate, in a sense, the limitation of relying mainly on part-of-speech tags to extract nested relations. The root of this limitation is that, with Open-CRF, one cannot avoid positive and negative examples which have the exact same features (recall our example on Section 4.2.3).

It is worth mentioning that this confusion introduced into Open-CRF is unavoidable, and not an artifact of the way in which we train the models. In fact, our examples were produced automatically from pairs of arguments in each sentence. Also, we tried to achieve the standard 50/50 split between positive (218) and negative examples (278) by automatically pruning some of the negative examples. Since negative examples are necessary to properly train a CRF model, it is thus hard to see a way of avoiding this confusion with Open-CRF.

Our results indicate that Meta-CRF often outperformed Open-CRF even when extracting flat instances only. This happened, for instance, on sentences such as "The A.P. reported Russia's conflict with Georgia", where we observed that a method needs to, at least, detect the nested instance involving "A.P." and the conflict. By doing so, the method avoids extracting spurious relations, such as: ("A.P.", "reported", "Russia"). This improvement over Open-CRF indicates that our model might be useful in an industry-strength information extraction system such as TextRunner. It would be interesting, for instance, to investigate whether the self-supervised training method used in TextRunner can be applied to our model.

Our method's improvement over Open-CRF comes at expense of processing time. This is because parse and dependency trees require heavyweight full parsing techniques. Processing time is a real concern when dealing with large amounts of text. The remaining of this thesis investigates the effectiveness-efficiency trade-off of using a range of natural language processing tools, such as shallow parsing, dependency parsing and semantic role labeling.

Chapter 5

EXEMPLAR: Extracting N-ary Relations

One limitation regarding the effectiveness of most ORE methods, including SONEX and Meta-CRF, is that they ignore n-ary relations ($n > 2$). This chapter discusses EXEMPLAR, an ORE method that extracts n-ary relations by using rules over dependency parsing trees. The work presented in this chapter appeared at the ACL Conference on Empirical Methods in Natural Language Processing (EMNLP) [66].

5.1 Overview

Problem statement. The problem addressed by EXEMPLAR is the extraction of n-ary relation instances from a corpus. More specifically, EXEMPLAR takes a list of dependency trees automatically generated from individual sentences and outputs n-ary instances. Figure 5.1 shows an example of dependency tree. An n-ary instance is a tuple $(p, a_1, \ldots, a_n)$, where p is a predicate and a_i is an argument.

Argument roles. Each argument a_i of a relation instance r is associated to a role $\rho(r, i)$. In ORE, the roles for each relation are not provided and must be recognized from the text. We use the following roles: `subject`, `direct_object` and `prep_object`. An argument has a role `prep_object` when its connected to the predicate by a preposition. The roles of prepositional objects consist of their preposition and the suffix "_object", indicating that each preposition corresponds to a different role. In the sentence "Arthur Blank is the owner ***of*** the Falcons", "Falcons"

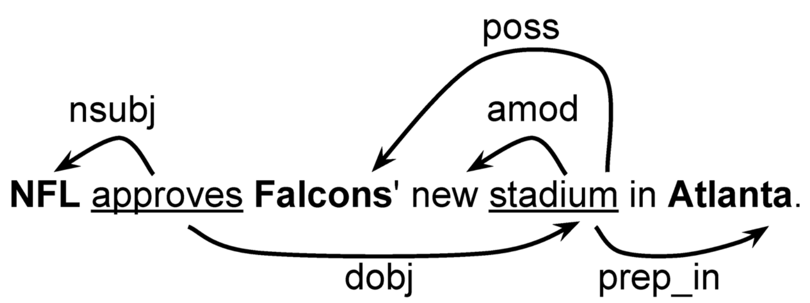


Figure 5.1: A dependency tree generated from an input sentence. EXEMPLAR's pre-processing step automatically identifies sentence boundaries, generates a dependency tree for each sentence and extracts named entities. Entities are in bold, triggers are underline and arrows represent dependencies.

is an object of the preposition "of" and has the role `of_object`. For an example, the instance described in Figure 5.1 is represented by the tuple:

$r =$("approve stadium", "NFL", "Falcons", "Atlanta")

and roles $\rho(r, 1) =$`subject`, $\rho(r, 2) =$`of_object` and $\rho(r, 3) =$`in_object`. Observe that multiple entities can play the same role in a relation instance. For instance, in the sentence "The Falcons and the Saints played well", both the "Falcons" and the "Saints" play the `subject` role. Furthermore, some predicate types accept less roles than others. Verb relations accept all three roles, while copula+noun and verb+noun relations accept `subject` and `prep_object` only.

Our roles are different from those used in SRL. SRL roles carry semantic information across different relations. This information is unavailable for ORE methods, and for this reason, we rely on syntactic roles. An open problem is to determine which syntactic roles correspond to the same semantic role across different relations [16]. However, this problem is out of the scope of this work.

5.2 Predicates and Predicate Types

Before designing extraction rules for EXEMPLAR, we investigated the structure of predicates for n-ary relations. We based our investigation in a similar study conducted over binary relations by Banko and Etzioni [6]. They claim that nearly 95% of predicates are expressed using eight syntactic patterns, the most popular being verb, verb+preposition and noun+preposition. We observed that these patterns are not suitable for n-ary instances, particularly those patterns containing a preposition. This is because such a preposition connects the verb or noun in the predicate to one of the arguments. Following this approach, the predicate of a n-ary instance may need to contain multiple prepositions. To see this, consider the sentence

> "The **Saints** were <u>beaten</u> by the **Falcons** in **New Orleans**."

which describes a relation instance involving three named entities: "Saints", "Falcons" and "New Orleans". If we were to add every preposition connecting "beaten" to an argument, the predicate for this instance would be "beaten by in". Producing predicates in this way is problematic since instances describing the same type of event (e.g., "beaten") may be assigned different predicates. For an example, consider the following instances:

> ("beaten by in", "Saints", "Falcons", "New Orleans"),
> ("beaten by", "Packers", "Bears")

These instances should describe two instances of the relation "beaten", but instead they belong to different relations because they do not have the same predicate. To avoid this problem, EXEMPLAR does not include prepositions in predicates.

To investigate the structure of predicates for n-ary relations, we collected 100 sentences describing an n-ary relation instance from a random sample of sentences in the New York Times Corpus [81]. For this, we used the Stanford NER system [31] to automatically recognize named entities and filter out those that contain two or less entities. Then, we manually reviewed approximately 500 sentences until we found 100 sentences containing an n-ary instance. We categorized the predicate found in these sentences according to their syntactic structure. As a result, we found that 86% of predicates are expressed using one of the patterns shown in Table 5.1.

Observe that some patterns listed in Table 5.1 are *redundant*. A pattern is redundant when every predicate expressed using this pattern can be expressed using a different pattern. For instance, the following patterns are redundant: verb ("beat"), passive verb ("was beaten") and nominalized verb ("beating"). We only need one of these patterns to represent all predicates composed by a single verb. In addition, the

Pattern	Frequency	Examples
Verb	30%	Atlanta Falcons beat ...
Apposition+noun	19%	Arthur Blank, the owner of ...
Passive verb	14%	The Saints were beaten by ...
Verb+noun	14%	NFL approves new stadium ...
Copula+noun	5%	Arthur Blank is the owner of ...
Nominalized verb	4%	The Saints' overtime loss to ...

Table 5.1: Patterns representing 86% of the relation instances with three or more arguments. Frequencies collected from 100 relations from the New York Times Corpus. Predicates are underlined.

Predicate Type	Freq.	Example	Variations
Verb	48%	beat	Pass. verb, nom. verb
Copula+noun	24%	is owner	Apposition+noun
Verb+noun	14%	approves stadium	–

Table 5.2: Predicate types recognized by EXEMPLAR.

pattern apposition+noun is redundant, since predicates expressed in this pattern can also be expressed using the pattern copula+noun (by replacing the apposition for the verb "be"). Therefore, most predicates can be expressed using one of following patterns: *verb*, *verb+noun* and *copula+noun*.

Predicate types. We classify predicates by the pattern used to express them. For instance, a predicate expressed via the pattern verb+noun is called a *verb+noun predicate*. Table 5.2 presents a list of predicate types, each one associated with a non-redundant pattern. We designed EXEMPLAR to specifically recognize predicates of these types, including predicates expressed using one of their redundant pattern variations. We define each predicate type as follows.

Definition 3. *A verb predicate is composed by a single non-copular verb.*

Definition 4. *A copula+noun predicate is composed by a copular verb and a noun acting as the copula's complement.*

Definition 5. *A verb+noun predicate is composed by a non-copular verb and a noun acting as the direct object of the verb.*

5.3 The EXEMPLAR Method

5.3.1 Preprocessing

Given a document, EXEMPLAR extracts its syntactic structure by applying a pipeline of NLP tools provided by the Stanford Parser [50]. Our method converts the original text into sentences, each containing a list of tokens. Each token is tagged with a lemma and part-of-speech tag. When describing EXEMPLAR's rules, we often refer to verbs and nouns. A verb is a token presenting one of the following tags: VB, VBD, VBG, VBN, VBP and VBZ. In addition, a noun is a token presenting one of the following tags: NN and NNS[1].

EXEMPLAR parses every sentence and produces a dependency tree. EXEMPLAR can work with any dependency parser whose output tree follows the Stanford dependency manual [15]. We have tested the performance of Exemplar using both the Stanford parser [50]. and the Malt parser [73].

Figure 5.1 illustrates an example of sentence where each word is a token and arrows represent dependencies among tokens. In this example, "stadium" depends on "approves" and the arrow connecting them can be read as "the direct object of *approves* is *stadium*".

Extracting Named Entities. EXEMPLAR employs the Stanford NER [31] to recognize named entities. We consider these types of entities: people, organization, location and miscellaneous. Figure 5.1 shows entities highlighted in bold.

5.3.2 Detecting triggers

After recognizing entities, EXEMPLAR detects *triggers*. A trigger is a single token that indicates the presence of a predicate. A predicate may have one or two triggers. For instance, the predicate in our running example (see Figure 5.1) has two triggers. In addition, the text of a predicate is generated from its triggers. A trigger can be any noun or verb that was not tagged as being part of a named entity. Other requirements are enforced for each predicate type as follows.

[1]For a list with all part-of-speech tags and their description, please refer to: `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`

Verb predicates. A verb predicate is triggered by any verb that does not present one of the following dependencies: direct object (dobj), passive subject (nsubjpass) or copula (cop). This is because a verb presenting one of these dependencies is a trigger of a different type of predicate, as discussed later. A verb predicate can also be triggered by a noun, as long as this noun is a nominalized verb. To identify nominalized verbs, EXEMPLAR checks whether a noun is filed under the type "event" in Wordnet's Morphosemantic Database[2]. This list contains nouns that (1) describe an event and (2) are derived from a verb. An example of such a noun is "beating", which is derived from the verb "to beat".

EXEMPLAR defines the text of a verb predicate by leveraging its triggers. If the trigger is a verb, EXEMPLAR uses the trigger's lemma as the predicate. If the trigger is a noun, the predicate is the lemma of the noun's original verb (before nominalization). For instance, the predicate for the trigger "beating" is "beat".

Copula+noun predicate. The trigger of a copula+noun predicate is a single noun that presents either: (1) a copula dependency (cop) with a verb or (2) an apposition dependency (appos) with a noun. When the trigger has a copula dependency with a verb, EXEMPLAR generates the predicate text by concatenating this verb's lemma and the trigger's lemma. For instance, consider the sentence "Arthur Blank became the owner of the Atlanta Falcons". EXEMPLAR takes the copula "become" and the trigger "owner" to generate the predicate "become owner". Conversely, when the trigger has an apposition dependency with a noun, EXEMPLAR generates the the predicate text by concatenating the copula "be" with the trigger's lemma.

Verb+noun predicate. To recognize verb+noun predicates, EXEMPLAR looks for a pair of triggers: a verb and a noun acting as its direct object (e.g., "approves stadium") or passive subject (e.g., "stadium was approved"). This pair of triggers must present one of the following dependency types: direct object (dobj) or passive subject (nsubjpass).

EXEMPLAR generates the predicate text by concatenating the verb's lemma with

[2]`http://wordnetcode.princeton.edu/standoff-files/morphosemantic-links.xls`

Dependency	Example
nsubj (Subject)	**Romeo** <u>loves</u> Juliet
dobj (Direct Object)	The **Prince** <u>exiles</u> Romeo
nsubjpass (Pass. Subj.)	**Romeo** was <u>seen</u> in Verona
agent (Pass. Voice Obj.)	Juliet is <u>loved</u> by **Romeo**
iobj (Indirect Object)	Romeo <u>gave</u> **Juliet** a <u>kiss</u>
poss (Possessive)	**Romeo**'s <u>father</u> Montague
appos (Apposition)	**Capulet**, Juliet's <u>father</u>,
amod (Adj. Modifier)	The **Italian** <u>city</u> of Verona
nn (Noun Comp. Mod.)	Romeo's <u>cousin</u> **Benvolio**
prep_* (Prep. Modifier)	Romeo <u>lived</u> in **Verona**
partmod (Participal Mod.)	**Romeo**, <u>born</u> in Italy
rcmod (Rel. Clause Mod.)	**Juliet**, who <u>loved</u> Romeo

Table 5.3: Dependencies that connect arguments and triggers. Arguments are in bold and triggers are underlined.

the noun's lemma. In the example of Figure 5.1, the lemmas of triggers "approves" and "stadium" are concatenated to form the predicate "approve stadium".

5.3.3 Detecting arguments and roles

After triggers are identified, EXEMPLAR proceeds to detect arguments. For this, EXEMPLAR looks at the dependency between an entity and a trigger separately. EXEMPLAR relies on two observations: (1) an argument is often adjacent to a trigger in the dependency graph, and (2) the type of the dependency can accurately predict the role of the argument.

Table 5.3 enumerates 12 types of dependencies (from a total of 53) that often connect arguments and triggers. EXEMPLAR identifies every entity connected to a trigger via one of these dependency types as a *candidate argument*.

For instance, consider our running example illustrated in figure 5.1. The entities "NFL" and "Atlanta" depend on the trigger "approves" and "Falcons" depends on the trigger "stadium". Since their dependency types are listed in Table 5.3, these entities are marked as candidate arguments.

Assigning roles. Given an argument and a trigger, EXEMPLAR determines the role of the argument by taking into account the trigger's POS tag (noun or verb), the

Trigger	Dependency	Role
Verb	>nsubj	`subject`
Verb	>agent	`subject`
Verb	<partmod	`subject`
Verb	<rcmod	`subject`
Verb	>dobj	`direct_object`
Verb	>subjpass	`direct_object`
Verb	>iobj	`to_object`
Verb	>prep_*	`prep_object`
Noun	>prep_by	`subject`
Noun	>amod	`subject`
Noun	>nn	`subject`
Noun	>poss	`subject`
Noun	>prep_of	`direct_object`
Noun	>prep_*	`prep_object`

(a) Rules for verb predicates.

Trigger	Dependency	Role
Verb	>nsubj	`subject`
Verb	>agent	`subject`
Verb	<partmod	`subject`
Verb	<rcmod	`subject`
Verb	>iobj	`to_object`
Verb	>prep_*	`prep_object`
Noun	>amod	`of_object`
Noun	>nn	`of_object`
Noun	>poss	`of_object`
Noun	>prep_*	`prep_object`

(b) Rules for verb+noun predicates.

Trigger	Dependency	Role
Noun	>nsubj	`subject`
Noun	>appos	`subject`
Noun	<appos	`subject`
Noun	<partmod	`subject`
Noun	<rcmod	`subject`
Noun	>amod	`of_object`
Noun	>nn	`of_object`
Noun	>poss	`of_object`
Noun	>prep_*	`prep_object`

(c) Rules for copula+noun predicates.

Figure 5.2: Rules for assigning roles to arguments.

type of dependency and the direction of the dependency. For simplicity, we define the direction of a dependency by prefixing its type with ">" when the argument depends on the trigger and "<" when the trigger depends on the argument.

Figure 5.2 shows EXEMPLAR's rules that assign roles to arguments. Each predicate type has a different set of rules. Figure 5.2(a) applies to verb predicates, Figure 5.2(b) applies to verb+noun predicates and Figure 5.2(c) applies to copula+noun predicates. Rules are represented by triples of the form (t, d, r), where t is a POS tag, d is a dependency type and r is a role. Given an argument–trigger pair, a rule is applied as follows:

> If trigger's POS tag = t and dependency type = d then assign role r to the argument.

For example, the first rule in Figure 5.2(a) specifies that, given an argument–trigger pair, the argument must be assigned the role `subject` if the trigger is a *verb* and the dependency type between argument and trigger is *>nsubj*.

The rules with dependency type >prep_* in Figure 5.2 are applied to dependency types with ">prep_" followed by any preposition (e.g., >prep_in, >prep_to), excluding those prepositions that have a specific rule. For instance, Figure 5.2(a) have a specific rule for preposition "by" (dependency >prep_by). Therefore, the last rule of Figure 5.2(a) (whose dependency is >prep_*) applies to all prepositions but "by". In addition, rules with dependency >prep_* assign a role that matches the preposition in a given sentence. For example, if the prepositional modifier is >prep_in, then the preposition is "in" and the assigned role is `in_object`.

Exceptions. There are three exceptions for the rules above. The first exception concerns arguments of verb predicates whose dependency type is <partmod (participial clause modifier) or <rcmod (relative clause modifier). EXEMPLAR chooses the role of `direct_object` (as oppose to `subject`) for these arguments *when the verb trigger is in passive form*. For instance, in the sentence "Barbie, (which was) invented by Handler", "Barbie" has the role `direct_object` because "invented" is in passive form.

The second exception is for nominalized verb triggers followed by the preposition "by", such as in "Georgian invasion by Russia". An argument connected to this trigger via one of the dependency types >nn, >amod or >poss is assigned the role `direct_object`.

Finallly, there is an exception for copula+noun predicates expressed via close appositions of the form: "determiner entity noun", such as "the Greenwood Heights section of Brooklyn". In this case, EXEMPLAR assigns the `subject` role to the entity between the determiner and the noun.

5.3.4 Filtering incomplete instances

The final step in EXEMPLAR is to remove incomplete relation instances. EXEMPLAR removes instances with less than two arguments and instances that present `prep_object` only.

5.4 Conclusion

We presented EXEMPLAR, an ORE method capable of extracting n-ary relations. EXEMPLAR leverages rules over dependency trees to detect relation instances. EXEMPLAR's rules are applied to each candidate argument individually as opposed to all candidate arguments of an instance. This makes EXEMPLAR's rules simpler and easier to be designed.

The next chapter discusses an evaluation of EXEMPLAR and other eight ORE methods over 5 datasets. As shown in the next chapter, EXEMPLAR shows promising results indicating that rule-based methods may still be very competitive. From a pragmatic point of view, EXEMPLAR is also preferable because it does not require training data.

An interesting research question is whether machine learning can be used to learn more rules for EXEMPLAR in order to improve recall without loss in precision. Rules could be learned from both dependency parsing and shallow parsing, or just shallow parsing if computing time is extremely limited. In addition, EXEMPLAR's rules may be used as features in machine learning methods, potentially improving their accuracy.

Chapter 6

The Effectiveness–Efficiency Trade-off

A large number of ORE approaches have been proposed, covering a wide range of NLP machinery, from "shallow" NLP (e.g., part-of-speech tagging) to "deep" NLP (e.g., semantic role labeling–SRL). A natural question then is what is the trade-off between NLP depth (and associated computational cost) versus effectiveness. This chapter presents a fair and objective experimental comparison of 9 state-of-the-art approaches over 5 different datasets, and sheds some light on the issue. The work presented in this chapter appeared the ACL Conference on Empirical Methods in Natural Language Processing (EMNLP) [66]. Our benchmarks have been published by the LDC[1].

Broadly speaking, existing ORE approaches can be grouped according to the level of sophistication of the NLP techniques they rely upon: (1) shallow parsing, (2) dependency parsing and (3) semantic role labelling (SRL). Shallow methods annotate the sentences with part-of-speech (POS) tags and the ORE approaches in this category, such as ReVerb [28] and SONEX [62], identify relation instances by matching patterns over such tags. Dependency parsing gives unambiguous syntactic relationships among each word in the sentence in the form of a dependency tree, and the ORE approaches in this category such as PATTY [71], OLLIE [59], Meta-CRF [64] and TreeKernel [95] identify subtrees connecting the relation trigger and its arguments. Finally, semantic annotators, such as Lund [45] and SwiRL [86],

[1]https://catalog.ldc.upenn.edu/LDC2014T27

add roles to each node in a parse tree, enabling ORE approaches that identify the precise connection between each argument and the relation trigger, independently.

Out of five datasets used in our evaluation, four were annotated manually, covering both well-formed sentences, from the New York Times (NYT) and the Penn Treebank, as well as mixed-quality sentences from a popular Web corpus. We also include a much larger dataset with 12,000 sentences from NYT, which were automatically annotated. Another experiment focuses on n-ary relation instances separately. The results show, as expected, that the three broad classes above are separated by orders of magnitude when it comes to throughput. In the same period of time, shallow methods handle ten times more sentences than dependency parsing methods, which in turn handle ten times more sentences than semantic parsing methods. Nevertheless, the cost-benefit trade-off is not as simple; and the higher computation cost of dependency or semantic parsing does not always pays off with higher effectiveness.

6.1 Compared ORE Methods

Shallow ORE. We evaluate two shallow methods: SONEX-p and ReVerb [28]. SONEX-p is a method based on SONEX 2.0's extraction patterns to detect predicates in a sentence as discussed in Section 3.9.2. SONEX-p does not employ the clustering module of SONEX. This is because this module recognizes and merges synonym predicates (e.g., "acquisition of", "acquired") and our benchmarks is not meant to evaluate the resolution of synonyms, as explained later. SONEX-p relies on Stanford's CoreNLP library for POS tagging and NER[2]. The binaries for SONEX-p is available at the SONEX project website[3]. To evaluate ReVerb, we use the source code available at the authors' website[4]. ReVerb relies on OpenNLP's POS tagger and phrase chunker[5]. We do not include TextRunner [6] in our evaluation because its authors proposed ReVerb and have shown that ReVerb outperforms TextRunner in their own evaluation [28].

[2]`http://nlp.stanford.edu/software/corenlp.shtml`
[3]`https://sites.google.com/a/ualberta.ca/sonex/`
[4]`https://github.com/knowitall/reverb`
[5]`https://opennlp.apache.org/`

Dependency-based ORE. We include five ORE methods based on dependency parsing in our evaluation: EXEMPLAR[S], EXEMPLAR[M], Meta-CRF, OLLIE, PATTY-p and TreeKernel. EXEMPLAR[S] and EXEMPLAR[M] are variants of our EXEMPLAR method that use the Stanford parser [50] and the Malt parser [73], respectively. Both variants rely the Stanford's CoreNLP library for POS tagging and NER. The code for EXEMPLAR is available for download online[6]. Meta-CRF uses Stanford's NER for entity recognition and the OpenNLP library[7] for POS tagging and dependency parsing. For OLLIE [59], we use the code available at the author's website[8]. The authors of TreeKernel [95] and PATTY [71] kindly provided us with the code of their methods. Since we only use PATTY's pattern-based method that extracts relation instances from a sentence (recall Section 2.3.2), we refer to this method as PATTY-p. As for the underlying NLP tools used by these methods, PATTY-p and OLLIE rely on Stanford's CoreNLP library for POS tagging, NER and dependency parsing. OLLIE relies on the OpenNLP library for POS tagging and Malt parser for dependency parsing.

SRL-based ORE. To represent methods based on SRL, we implemented two methods: Lund-IE and SwiRL-IE. Both methods are based on the SRL-IE approach [19], which converts SRL annotations into relation instances. Lund-IE is based on the Lund system [45], while SwiRL-IE is based on the SwiRL system [86]. SwiRL is trained on PropBank and only recognizes verb triggers. On the other hand, Lund is trained on both PropBank and NomBank, making it able to extract instances triggered by either a verb or a noun. Since not all SRL arguments are named entities, we ignore any argument whose head word is not an entity. For this, we use the Stanford NER[9] to recognize named entities and the heuristics proposed by SwiRL's authors to find the *content* head word of an argument [86].

Dataset	Source	# Sentences	# Instances
WEB-500	Search Snippets	500	461
NYT-500	New York Times	500	150
PENN-100	Penn Treebank	100	51

Table 6.1: Binary relation datasets.

6.2 Experimental Study

This section compares the effectiveness and efficiency of several ORE methods. We start by evaluating the extraction of binary relations using manual annotations. Then, we move to evaluate the extraction of n-ary relations using manual annotations. Finally, we discuss a method to produce automatic annotations for the evaluation of binary relations.

6.2.1 Binary relations – Setup

We start by evaluating the extraction of binary relations. Table 6.1 shows our experimental datasets. WEB-500 is a commonly used dataset, developed for the TextRunner experiments [6]. This dataset contains 500 sentences extracted from search engine snippets. These sentences are often incomplete and grammatically unsound, representing the challenges of dealing with web text. NYT-500 represents the other end of the spectrum with individual sentences from formal, well written new stories from the New York Times Corpus [81]. PENN-100 contains sentences from the Penn Treebank recently used in an evaluation of the TreeKernel method [95]. We manually annotated the instances for WEB-500 and NYT-500 and use the PENN-100 annotations provided by TreeKernel's authors [95].

The task. We manually annotate each sentence as follows. For each sentence, we annotate two entities and an optional *trigger* (a single token indicating the presence of a predicate—see Section 5.3.2) for the predicate describing the relationship be-

[6]https://github.com/U-Alberta/exemplar
[7]http://opennlp.sourceforge.net
[8]https://github.com/knowitall/ollie
[9]http://nlp.stanford.edu/software/CRF-NER.shtml

tween them. If such a relationship does not exist, we annotate the two entities only. In addition, we specify a *window* of tokens allowed to be in a predicate, including modifiers of the trigger and prepositions connecting triggers to their arguments.

Given a sentence, a method being evaluated must extract every relation instance described in this sentence. Our evaluation ignores all extracted instances but one involving the annotated pair of entities. This instance is deemed correct if its predicate contains the trigger and allowed tokens only.

Annotations. Figure 6.3 in Page 111 shows several examples of annotated sentences. In these sentences, entities are enclosed in triple square brackets, triggers are enclosed in triple curly brackets and the window of allowed tokens for a predicate is defined by arrows ("`--->`" and "`<---`"), as in this example:

```
I've got a media call on [[[Google]]] ---> 's {{{acqui-
sition}}} of <--- [[[YouTube]]] ---> today <---.
```

where "Google" and "YouTube" are entities, "acquisition" is the trigger and the allowed tokens are "acquisition", "'s" and "of". We include time and location modifiers (e.g., "today", "here") in the list of allowed tokens since OLLIE extracts them as part of the predicate. OLLIE's predicates may also include auxiliary verbs and prepositions that are not present in the original sentence. To be fair with OLLIE, we remove auxiliary verbs and prepositions from OLLIE's predicates.

Ensuring entities are recognized properly. Since every method uses a different tool to recognize entities, we try to ensure every method is able to recognize the entities marked by our annotators. We replace the original entities by a single word, preventing any method from recognizing only part of an entity. Entities are replaced by the tokens "Europe" and "Asia", since we empirically found that, for 99.7% of the sentences in our experiment, all methods were able to recognize "Europe" and "Asia" as entities (or nouns, for methods that do not use a NER tool). In addition, we did not find any occurrence of "Europe" and "Asia" in the original sentences that could conflict with our entity placeholders.

For methods that consider any noun phrase as an entity (ReVerb, OLLIE, SwiRL-IE and Lund-IE), there is the additional task of identifying whether a noun phrase containing additional words surrounding “Europe” and “Asia” is still a reference to the annotated entity. For example, “the beautiful Europe” refers to the entity, while “Europe’s enemy” does not. In our evaluation, we ignore noun phrases that do not reference the annotated entity. For SwiRL-IE and Lund-IE, we ignore any noun phrase that do not present “Europe” or “Asia” as its head word. For ReVerb and OLLIE, we ignore noun phrases that do not contain these words in the end of the phrase.

Dealing with n-ary instances. For EXEMPLAR, Lund-IE and SwiRL-IE, which extract n-ary instances, our evaluation needs to convert n-ary instances into binary ones. This is done by selecting all pairs of arguments from a n-ary instance and creating a new (binary) instance for each of them. We remove binary instances containing two arguments with the same role. We also remove instances containing two prepositional objects, regardless of the preposition.

Metrics. Our evaluation focuses on the extraction of relation instances at a sentence level. For this, we assume there is only one instance involving a pair of entities in a sentence. The number of entity pairs involved with more than one instance was insignificant in our datasets (less than 0.5%).

The metrics used in this analysis are precision (P), recall (R) and f-measure (F), defined as usual:

$$P = \frac{\text{\# correctly extracted instances}}{\text{\# extracted instances}}, R = \frac{\text{\# correctly extracted instances}}{\text{\# annotated instances}}, \quad (6.1)$$

$$F = \frac{2PR}{P+R} \quad (6.2)$$

We also measure the total computing time of each method, excluding initialization or loading any libraries or models in memory. To ensure a fair comparison, we make sure each method runs in a single-threaded mode, thus utilizing a single computing core at all times.

6.2.2 Limitations and fairness

Our benchmarks are meant to evaluate *lexical-syntactic methods*; that is, methods that analyze the lexical-syntactic structure of individual sentences to extract predicates as described in this sentence.

We claim that our benchmarks are fair to all lexical-syntactic methods since our annotations allow all possible variations of a predicate (e.g., “acquired”, “acquired part of”, “has just acquired part of”) as long they contain the relation’s trigger word (“acquired”) and are within a certain window of allowed words. Our benchmarks would be unfair if a variation was favoured over another, penalizing a method for including or ignoring non-essential words in the predicate.

On the other hand, our benchmarks are not meant to evaluate *semantic methods*; that is, methods that leverage multiple sentences to recognize and merge predicates that are synonyms. Semantic methods include PATTY, SONEX or any other method that generates a canonical predicate to represent several synonym predicates. Our benchmarks are also not meant to evaluate methods that produce relations by inference.

One way to expand our benchmarks to evaluate semantic methods would be to allow synonyms for each trigger word. For this, one could map each trigger to a corresponding WordNet[10] sense.

6.2.3 Binary relations – Results

Table 6.2 presents the results for our experiment with binary relations. WEB-500 turned out to contain the easiest sentences as evidenced by the precision of all methods in this dataset. This is because WEB-500 sentences were collected by querying a search engine with known relation instances. As a result, 92% of the sentences in WEB-500 describe a relation instance. The other two datasets, on the other hand, contain randomly chosen sentences. NYT-500 presents only 30% of sentences describing an instance, while PENN-500 presents 52%. These findings indicate that although WEB-500 is a popular dataset, it perhaps does not represent the challenges

[10] `http://wordnet.princeton.edu/`

Method	NYT-500				WEB-500				PENN-100			
	Time	P	R	F	Time	P	R	F	Time	P	R	F
ReVerb	**0.02**	0.70	0.11	0.18	**0.01**	0.92	0.29	0.44	**0.02**	0.78	0.14	0.23
SONEX-p	0.04	0.77	0.22	0.34	0.02	**0.98**	0.30	0.45	0.04	0.92	0.43	0.59
OLLIE	0.05	0.62	0.27	0.38	0.04	0.81	0.29	0.43	0.14	0.81	0.43	0.56
EXEMPLAR[M]	0.08	0.70	**0.39**	0.50	0.06	0.95	0.44	0.61	0.16	0.83	0.49	**0.62**
Meta-CRF	0.13	0.59	0.17	0.27	0.08	0.89	0.36	0.51	0.12	**1.00**	0.13	0.24
EXEMPLAR[S]	1.03	0.73	**0.39**	**0.51**	0.47	0.96	**0.46**	**0.62**	0.62	0.79	**0.51**	**0.62**
PATTY-p	1.18	0.49	0.23	0.32	0.48	0.71	0.48	0.57	0.66	0.46	0.24	0.31
SwiRL-IE	2.96	0.63	0.10	0.17	1.73	0.97	0.34	0.50	2.17	0.89	0.16	0.27
Lund-IE	11.40	**0.78**	0.24	0.37	2.69	0.91	0.37	0.52	5.21	0.86	0.35	0.50
TreeKernel	–	–	–	–	–	–	–	–	0.85	0.85	0.33	0.48

Table 6.2: Results for the task of extracting binary relations. Methods are ordered by computing time per sentence (in seconds). Best results for each column are underlined and highlighted in bold.

found in web text.

We were unable to run TreeKernel for NYT-500 and WEB-500 for lack of training data. We ran TreeKernel, as trained by its authors, on the same test set used in their paper [95].

Comparing methods based on effectiveness (f-measure) or efficiency (computational cost) alone can be misleading. Instead, we compare methods in terms of *dominance*. We say method A dominates method B if A is: (1) *more effective* and as efficient as B; (2) *more efficient* and as effective as B; or (3) both more effective and more efficient than B. The methods that are not dominated by any other form the state-of-the-art.

Figure 6.1 plots the effectiveness and efficiency of all methods, averaged over all datasets (TreeKernel was not included due to missing results). The lines in the plot identify the state-of-the-art before (dashed) and after (solid) EXEMPLAR. (Note: Although not clear in the figure, OLLIE and Lund-IE are dominated by SONEX-p as they tie in f-measure.)

In terms of efficiency, there is a clear separation of approximately one order of magnitude among methods based on shallow parsing (ReVerb and SONEX-p), dependency parsing (OLLIE, EXEMPLAR[M], EXEMPLAR[S], and PATTY-p) and semantic parsing (SwiRL-IE and Lund-IE). ReVerb is fastest method since it uses shallow parsing and does not rely on a NER system. SONEX-p is more effective than ReVerb since it is able to recognize predicates triggered by a noun. However,

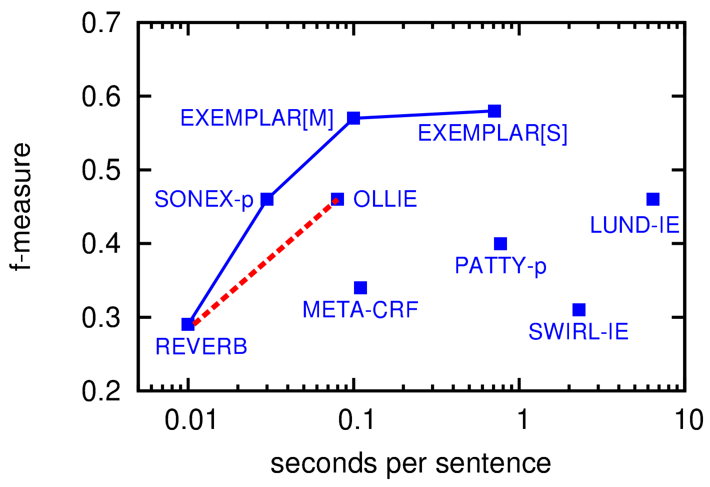


Figure 6.1: Average f-measure vs average time for NYT-500, WEB-500 and PEN-100.

SONEX-p is less efficient then ReVerb because of the overhead introduced by its NER.

EXEMPLAR[M] and EXEMPLAR[S] closely match OLLIE and PATTY, respectively, since they use the same dependency parsers. As for effectiveness, EXEMPLAR outperforms both methods (OLLIE and PATTY). EXEMPLAR is more effective than these methods mainly because it can recognize more correct instances (i.e., higher recall), particularly those with verb+noun predicates. This is because EXEMPLAR allows multiple triggers, which is essential to detect verb+noun predicates. Moreover, EXEMPLAR analyzes the path between an argument and a trigger separately, as opposed to the whole subtree connecting two arguments. As it turns out, this design choice greatly simplifies the task of designing good patterns for predicates with multiple triggers.

Unlike EXEMPLAR, OLLIE considers only one trigger per predicate. As a consequence, OLLIE cannot extract an argument between a verb and noun forming a verb+predicate. For instance, consider the following sentence:

```
Although [[[Cuba]]] ---> has announced their intention of
{{{joining}}} the [[[North Korean]]] boycott of the Olympics
<--- , the Cuban players have told their American counterparts
during their current tour of the United States that they think
their country will change its mind.
```

In this example, the argument “North Korean” is in between a verb and noun form-

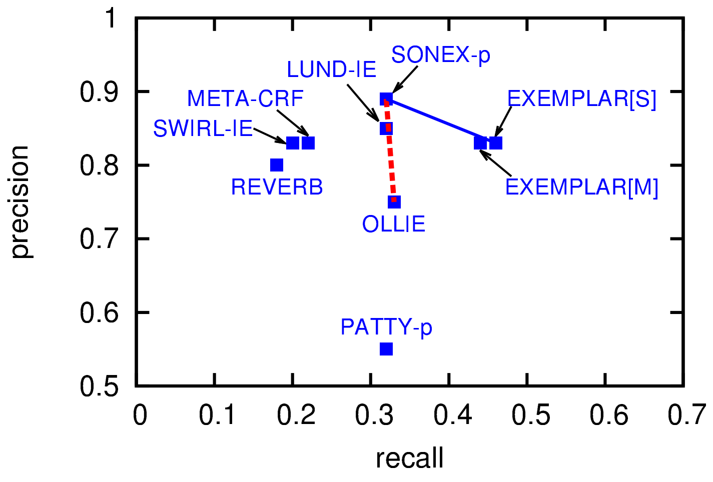


Figure 6.2: Average precision vs average recall for NYT-500, WEB-500 and PEN-100.

ing the predicate "joined boycott". This pattern can be seen in over 30% of the instances in the NYT-500 dataset. OLLIE's lower recall can be explained by its inability to detect these instances.

The poor performance of PATTY-p and Meta-CRF can be explained as follows. PATTY-p applies rather permissive extraction patterns, which detect most dependency paths between arguments as a predicate. The original method, PATTY, relies on redundancy to normalize predicates in order to recover from mistakes done in the sentence-level. Meta-CRF has achieved high precision but low recall. Such a low recall is likely due to insufficient training examples, since Meta-CRF was trained with a few hundred examples only.

Figure 6.2 illustrates the dominance relationship differently, using precision versus recall. Again, the dashed line shows the previous state-of-the-art, and the solid line shows the current situation. SONEX-p dominates PATTY-p and Lund-IE, since they tied in recall. OLLIE, however, achieved greater recall than SONEX-p.

Somewhat surprisingly, EXEMPLAR presents 44% more recall than the more sophisticated Lund-IE, at a close level of precision. This can be explained by Lund-IE's dependency on SRL training data. Particularly, Lund-IE is unable to extract verb+noun predicates since SRL predicates are single words only. In addition, Lund-IE was unable to detect predicates triggered by nouns not included in the training data, such as “psychologist” and “company”.

The importance of noun triggers is illustrated by the higher recall of SONEX-p and Lund-IE when compared, respectively, to ReVerb and SwiRL-IE, similar methods that handle verb triggers only.

6.2.4 Binary relations – Discussion

Differences in annotation. It is worth noting some differences between our annotations (WEB-500 and NYT-500) and the annotations from PEN-100. The first difference concerns the definition of an entity. Consider the following sentence from PEN-100:

> “… says Leslie Quick Jr., chairman of the Quick & Reilly discount brokerage firm.”

Unlike our annotation style, the original annotation defines that “Leslie Quick Jr.” is the chairman of “the Quick & Reilly discount brokerage firm”, as opposed to “Quick & Reilly”. While we consider the words surrounding “Quick & Reilly” as apposition, the original consider them as part of the entity.

Another difference concerns the definition of the RE task. We assume that RE methods are responsible for resolving co-references when necessary to identify an instance. For example, consider the sentence:

> “It also marks P&G’s growing concern that its Japanese rivals, such as Kao Corp., may bring their superconcentrates to the U.S.”

According to our annotation style, there is an instance of relation “rivals” involving “P&G” and “Kao Corp.” in this sentence. Conversely, according to the annotation style adopted by TreeKernel’s authors [95], the instance of “rivals” involves “Kap Corp.” and the pronoun “it”, leaving the task of resolving the coreference between “P&G” and “it” as a posterior step.

These differences in annotation illustrate the challenges of producing a benchmark for open relation extraction.

Differences in evaluation methodology. A *sentence-level* evaluation like ours focuses on each sentence, separately. On the other hand, the evaluations of SONEX,

Method	NYT n-ary			
	Time	P	R	F
EXEMPLAR[M]	**0.11**	0.94	**0.40**	**0.56**
OLLIE	0.12	0.87	0.14	0.25
EXEMPLAR[S]	0.88	0.92	0.39	0.55
SwiRL-IE	2.90	0.94	0.30	0.45
Lund-IE	9.20	**0.95**	0.36	0.53

Table 6.3: Results for n-ary relations.

ReVerb, PATTY, TreeKernel and OLLIE are performed at *the corpus level*. Corpus-level evaluations consider an extracted instance as correct regardless of whether a method was able to identify one or all sentences that describe this relation instance.

Creating a ground truth for corpus-level evaluations is extremely hard, since one has to identify and curate all relations described in a corpus. This often involves detecting synonym predicates and co-referential entities. As a consequence, corpus-level evaluations perform only an inspection of a method's extracted instances. This inspection measures a method's precision, but is unable to measure recall.

Other differences in methodology are as follows. PATTY's evaluation concerns predicates (e.g., "wrote hits for") and their argument types (e.g. Musician–Musician), as opposed to relation instances. The evaluations of ReVerb and OLLIE consider any noun phrase as a potential argument, while the evaluations of TreeKernel and SONEX consider named entities only.

Due to the lack of a ground truth and differences in evaluation methodology, results from different papers are usually not comparable. This work tries to alleviate this problem by providing reusable annotations that are flexible and can be used to evaluate a wide range of methods.

6.2.5 N-ary relations

The goal of this experiment is to evaluate the accuracy and performance of our method when extracting n-ary relations ($n > 2$). For this experiment, we manually tagged 222 sentences with n-ary instances from the New York Times. Every sentence is annotated with a single trigger and its arguments.

Method	NYT 12K			
	Time	P	R	F
ReVerb	0.01	0.84	0.11	0.19
OLLIE	0.02	0.85	0.22	0.35
SONEX	0.03	**0.87**	0.20	0.32
EXEMPLAR[M]	0.05	**0.87**	0.26	0.40
EXEMPLAR[S]	1.20	0.86	**0.29**	**0.43**
PATTY	1.29	0.86	0.18	0.30
SwiRL-IE	3.58	**0.87**	0.16	0.27
Lund-IE	11.28	0.86	0.21	0.33

Table 6.4: Results for binary relations automatically annotated using Freebase and WordNet.

This experiment measures precision and recall over the extracted arguments. For each sentence, a method is asked to extract a relation instance of the form $(p, a_1, a_2, \ldots, a_n)$, where p is the predicate and a_i is an argument. If multiple instances are extracted, we only use the extracted instance whose predicate contains the annotated trigger, if one exists. We evaluate the correctness by looking at individual arguments separately. An extracted argument is deemed correct if it is annotated in the sentence; otherwise, it is deemed incorrect.

Precision and recall are now defined as follows:

$$P = \frac{\text{\# correctly extracted arguments}}{\text{\# extracted arguments}}, R = \frac{\text{\# correctly extracted arguments}}{\text{\# annotated arguments}}. \tag{6.3}$$

There are 765 annotated arguments in total. Table 6.3 reports the results for our experiment with n-ary relations. EXEMPLAR[M] shows a 6% increase in f-measure over Lund-IE, the second best method, while being almost two orders of magnitude faster.

6.2.6 Automatically annotated sentences

The creation of datasets for open RE is an extremely time-consuming task. In this section we investigate whether external data sources such as Freebase[11] and Word-

[11]http://www.freebase.com

Net[12] can be used to automatically annotate a dataset, leading to a useful benchmark.

Our *automatic annotator* annotates an sentence by highlighting a pair of entities and a single token triggering an instance involving them. It does so by first trying to link all entities in a sentence to Wikipedia (and consequently to Freebase, since Freebase is linked to Wikipedia) by using the method proposed by [22]. For each two entities appearing within 10 tokens of each other in a sentence, our annotator checks whether Freebase has this entity pair as an instance of a relation. If such an instance exists, the annotator tries to find a trigger of this instance in the sentence. A trigger must be a synonym for the Freebase's relation name (according to WordNet) and its distance to the nearest entity cannot be more than 5 tokens.

We applied this method for the New York Times and were able to annotate over 60,000 sentences containing over 13,000 distinct entity pairs. For our experiments, we randomly selected one sentence for each entity pair and separated a thousand for development and over 12,000 for test.

Comparing with human annotators. Although we expect our automatic annotator to be less accurate than a human annotator, we are interested in measuring the difference in accuracy between them. To do so, two authors of this paper looked at our development set and marked each sentence as correct or incorrect. The agreement (that is, the percentage of matching answers) between the humans was 82%. On other hand, the agreement between our automatic annotator and each human was 71% and 72%. This shows that our annotator's accuracy is not too far below human's level of accuracy.

Table 6.4 shows the results for the test sentences. Both EXEMPLAR[S] and EXEMPLAR[M] outperformed all methods in recall, while keeping the same level of precision.

[12]`http://wordnet.princeton.edu/`

6.3 Conclusion

We presented new benchmarks for ORE enabling a fair and objective evaluation of several methods. Our benchmarks are fair since they do not penalize an ORE method for including or excluding non-essential tokens in a predicate. This is achieved by annotating a trigger and a list of allowed tokens for each predicate. An extracted predicate is deemed as correct when it contains the trigger and any number of allowed tokens.

Our evaluation shed some light on the trade-off between effectiveness and efficiency of ORE methods. For this, we defined that the state of the art in ORE is composed by methods that are not dominated by any other methods, i.e., no other method is both more efficient and more effective than these methods. In our experiments, the state-of-the-art methods for binary relation extraction are: ReVerb, SONEX-p, EXEMPLAR[M] (which uses the Malt parser) and EXEMPLAR[S] (which uses the Stanford parser). ReVerb is the fastest method. SONEX-p outperforms ReVerb in effectiveness, but its NER's overhead make it less efficient. The EXEMPLAR variations present similar effectiveness, but differ greatly in efficiency. This is because the Malt parser is almost one order of magnitude faster than the popular Stanford parser, while producing dependency trees that are almost as good.

An interesting observation is that the state-of-the-art is composed exclusively by methods that rely on hand-crafted rules. These methods dominate 3 methods based on machine learning: OLLIE, Lund-IE and SwiRL-IE. However, these methods' underperformance is not due to their statistical learning approach. Instead, this underperformance is a statement of the importance of detecting multiple triggers for a predicate. Over 30% of the instances described in one of our datasets required the detection of two triggers. Since these methods consider only one trigger, they were not able to extract these instances. Furthermore, we discussed some issues with SRL training data that make it inadequate for ORE.

Our experiments indicate that fast yet accurate dependency parsers is a promising direction for scalable ORE. This is because dependency parsing have a significant effect in effectiveness when compared to POS tagging. Conversely, SRL's improvement over dependency parsing is insignificant.

A caveat with our results is that the compared methods differ in many factors, not only in their underlying NLP machinery. In the next chapter, we present additional experiments to confirm whether our observations are due to NLP tools used by these methods or a different factor. For this, we evaluate one method with different NLP tools.

```
1.  ---> finally <--- [[[google]]] ---> bought <---
    [[[youtube]]]
2.  [[[Google]]] ---> confirms <--- [[[YouTube]]] --->
    {{{aquisition}}} <--- - BBC News
3.  On January 31 , 2006 , [[[Viacom]]] ---> completed
    its {{{acquisition}}} of <--- [[[DreamWorks]]] LLC (
    _ ) , a producer of live - action motion pictures
    , television programming and home entertainment
    products .
4.  [[[Gershwin]]] ( 1898 - 1937 ) , ---> was {{{born}}}
    in <--- Brooklyn , [[[New York]]] , the son of
5.  Fascinating facts about [[[Ruth Handler]]] --->
    {{{inventor}}} of <--- [[[Barbie]]] in 1959 .
```

(a) Examples of annotated sentences from the WEB-500 dataset. Typos, missing punctuations and any other linguistic issues have been transcribed exactly as found in the source text.

```
6.  But they will also see the work of the woman
    who inspired Mr.  Martin 's tale in the first
    place :  the artist Allyson Hollingsworth , who
    created the photographs and drawings attributed
    to [[[Ms.  Danes]]] ---> 's {{{character}}} <--- ,
    [[[Mirabelle Buttersfield]]] , and who also served
    as a consultant on the film .
7.  Another character , the [[[Robber Daughter]]] ,
    and her mother ---> plot to `` {{{kill}}} '' <---
    [[[Gerda]]] ( a light moment with dark undertones ,
    which may make some children shudder ) .
8.  He shot and missed a fallaway 10-footer ,
    [[[Bramlett]]] missed a tip-in and [[[Jefferson]]]
    put up an 8-foot air ball as the buzzer sounded .
```

(b) Examples of annotated sentences from the NYT-500 dataset. Sentence #8 is an example of sentence without annotated trigger.

```
9.   That 's when [[[George L. Ball]]] , --->
     {{{chairman}}} of the <--- [[[Prudential Insurance
     Co.  of America]]] ---> unit <--- , took to the
     internal intercom system to declare that the plunge
     was only `` mechanical .  ''
10.  A group of Arby 's franchisees said they formed
     an association to oppose [[[Miami Beach]]] --->
     {{{financier}}} <--- [[[Victor Posner]]] 's control
     of the restaurant chain .
```

(c) Examples of annotated sentences from the PENN-100 dataset. This dataset contains many predicates triggered by a noun.

Figure 6.3: Examples of annotated sentences from WEB-500, NYT-500 and PENN-100.

Chapter 7

Efficiens: Allocating Computational Resources for Efficient ORE

Current ORE methods apply a fixed set of NLP tools for every sentence. For instance, ReVerb applies POS tagging for every sentence, while EXEMPLAR applies POS tagging and dependency parsing. These methods do not provide a solution for cases where the user is willing to allocate more computational resources (e.g., use more expensive NLP tools) in order to improve the quality of the extracted relations. This chapter discusses Efficiens, an ORE method that can apply different NLP tools for each sentence. Efficiens applies POS tagging for all sentences and allows the user to choose the percentage of sentences that can be processed by dependency parsing and SRL.

By applying more expensive NLP tools for a sentence, Efficiens can improve effectiveness. This is because the additional information about a sentence can help Efficiens to extract more correct instances and less incorrect ones. However, we have found that improvements like these occur rather infrequently. In particular, we applied POS tagging, dependency parsing and SRL for our development dataset containing 300 sentences from the The New York Times. Then, we compared the instances produced using POS tagging only and those produced using deep NLP tools (dependency parsing and SRL). We have found that only 7% of the sentences in this dataset produced better instances with deep NLP tools. In other words, an extractor that applies deep NLP for all sentences will be wasting computational resources for 93% of the sentences in this specific dataset. Therefore, it is desirable for Efficiens (and any ORE method, for that matter) to prioritize sentences according their likelihood of improving effectiveness with the additional information provided by deep NLP tools. In this way, Efficiens would be able to better allocate computational resources and avoid wasting these resources in sentences that are unlikely to improve effectiveness.

7.1 Overview

Efficiens takes a budget in the form of two parameters: α, β in the range $[0, 1]$. These parameters define the number of sentences that can be processed by the more expensive NLP tools: dependency parsing and SRL. Given a set of N sentences, α defines that $\alpha \times N$ should be processed by dependency parsing, while β defines that $\alpha \times \beta \times N$ should be processed by SRL. These refer to these subsets as α subset and β subset, respectively. Each of the N sentences is processed by POS tagging.

Efficiens has a module for each NLP tool. The Efficiens[POS] module relies on POS tagging, while the Efficiens[DEP] and Efficiens[SRL] rely on dependency parsing and SRL, respectively. Each module can be used as a stand-alone ORE method.

Problem statement. Given a set of sentences, the problem addressed by each Efficiens module is to extract relation instances of the form $(p, a_1, \ldots, a_n)$, where p is a predicate and a_i is an argument. Each argument a_i must be associated with one of the following roles: `subject`, `direct_object` or `prep_object`, where `prep_object` folds into many roles, one for each preposition.

The problem addressed by Efficiens is to choose which sentences should be processed by each module in order to maximize effectiveness. Given a corpus and the budget parameters α and β, Efficiens selects sentences for the α subset and β subset as to maximize the number of correctly extracted instances.

To choose the α and β subsets, Efficiens assigns a score to each sentence trying to assign higher scores to sentences that are more likely to improve the quality of extracted instances. The α and β subsets are composed of the top $\alpha \times N$ highest scored sentences and top $\beta \times N$ highest scored sentences, respectively.

7.2 The Efficiens Method

Efficiens reads every document sequentially and extracts instances in a single pass through the corpus. For each document, the Stanford's CoreNLP library [88] is used to extract tokens and sentences. Sentences are then processed by a pipeline of modules, as discussed next.

Efficiens pipeline. Figure 7.1 illustrates the Efficiens pipeline. All sentences are first processed by the Efficiens[POS] module, which extracts instances using the information provided by the Stanford POS tagger [88]. Next, Efficiens scores every sentence and sends the top $\alpha \times N$ sentences (the α subset) to Efficiens[DEP]. Instances extracted from the *chosen* sentences are discarded and instances extracted from the *non chosen* sentences are returned. Efficiens[DEP] applies the Malt dependency parser [73] over the α subset and produces new instances. This module also exploits the POS tags generated in the previous module. Next, Efficiens chooses the top $\alpha \times \beta \times N$ sentences (the β subset) from the α subset and sends them to the final module: Efficiens[SRL]. As before, the instances extracted from non chosen sentences are returned. Efficiens[SRL] produces instances by leveraging a new version[1] of Lund [8], the SRL system from Mate Tools[2]. For differentiation, we call this system Lund2. Lund2 uses an internal dependency parser [9], whose output is not compatible with the Malt parser. Therefore, Efficiens[SRL] uses features from SRL and dependency parsing from Lund2 in addition to the features produced by the two previous modules.

7.2.1 Extracting relation instances with an Efficiens module

Figure 7.2 illustrates the architecture of an Efficiens module. All modules follow the same architecture; they differ only over the NLP tool applied to a sentence. Given a sentence, the first step is run one of the following NLP tools: POS tagging, dependency parsing or SRL, depending on the module.

[1]Note that we have evaluated an older version of Lund in Chapter 6.

[2]`https://code.google.com/p/mate-tools/`

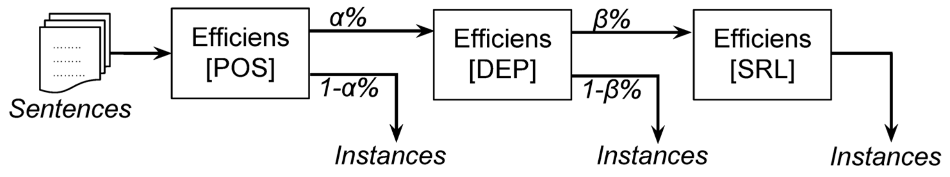


Figure 7.1: The Efficiens Pipeline. Sentences are buffered before the Efficiens[POS] modules until the buffer is full. All buffered sentences are then processed by Efficiens[POS]. The α subset of sentences goes through Efficiens[DEP]. Finally, the β subset of sentences goes through Efficiens[SRL].

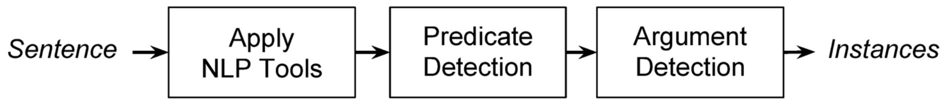


Figure 7.2: The architecture of a module. Each module employs a set of NLP tools, such as POS tagging, dependency parsing or SRL. Using the information provided by these tools, this module detects predicates and arguments of this predicate.

Detecting predicates. The next step is to detect predicates. For this, an Efficiens module generates a list of candidate predicates. A candidate is composed by one trigger (a verb *or* noun) or two triggers (a verb *and* noun). In order to be considered a valid predicate, a candidate must fall into one of the possible predicate types.

Figure 7.3 shows the types of predicates recognized by Efficiens. Our module tries to classify each candidate into one of five classes: verb+noun, verb+prep+noun, copula+noun, possessive+noun or "none", for candidates that do not belong to any of the predicate types. Our module uses a logistic regression classifier and follows the "one vs. all" approach to multiclass classification. Our module employs 5

Predicate Type	Triggers	Example Sentences
Verb	1	"X acquired Y", "X was seen with Y"
Verb+noun	2	"X made a deal with Y", "X invited Y's leaders"
Verb+prep+noun	2	"X gave Y up for adoption", "X will stay on Y's side"
Copula+noun	1 or 2	"X is the president of Y", "X, a operations director at Y"
Possessive+noun	1	"X's acquisition of Y", "X's headquarters at Y"

Figure 7.3: Predicate types recognized by Efficiens with example sentences. Triggers are underlined in examples. The column "Triggers" shows the number of triggers accepted for each type. Copula+noun accepts two triggers when the copula is explicitly mentioned. It also accepts one trigger when the copula is implicitly mentioned via apposition.

classifiers, one for each class. Given a candidate, these classifiers return a number between 0 and 1 indicating the likelihood of the candidate belonging to a particular class. A candidate whose predicted class is different from "none" is sent to the next step for argument detection.

Detecting arguments. For each predicate, the argument detection step generates a list of candidate arguments and classify each candidate according to its role: `subject`, `direct_object`, `prep_object`, or "none" (for candidates that are not arguments). A candidate argument is represented by a single noun, which is the head word of the argument. There is flexibility over which nouns are used as candidates. The user may choose to use any noun (including proper and common nouns), proper nouns only or nouns marked as entities by a NER. Therefore, Efficiens does not require a NER. As in the previous step, each type is recognized by an individual logistic regression classifier.

The advantage of using logistic regression classifiers is that their output can be interpreted as probabilities, allowing Efficiens to assign a confidence score to each classification, particularly to the classification of predicates and arguments as described in this section. Another reason for using logistic regression is that it has been successfully applied for argument detection in both ORE [28] and SRL [45].

7.3 Learning

Training data for ORE is scarce. Most methods circumvent this problem by producing training data automatically via heuristics [6, 59] or manually annotating small datasets [28, 64]. The problem with these circumventions is that they have a negative impact on a classifier's accuracy. This is because automatic annotations are noisy and small datasets seldom provide enough statistical support.

We produce training data for Efficiens by leveraging SRL annotations. While SRL annotations are quite different from the output of ORE methods, they can be automatically converted into training data for ORE. One advantages of leveraging SRL annotations to produce ORE training data is that these annotations are handcrafted by experts, which reduces the changes of introducing noise into the training data. Another advantage is that these annotations are available in large quantities through initiatives like PropBank and NomBank. In particular, we use the dataset published by the CoNLL-2009 Shared Task [40], that integrates annotations from several initiatives into a single dataset. Each sentence in this dataset is annotated with POS tags, a dependency tree and SRL annotations.

To produce training data for Efficiens, we designed a rule-based method similar to EXEMPLAR that leverages a sentence's dependency tree and SRL annotations to produce relation instances. We call this method Trainer and describe its rules in Appendix A. We use the instances extracted by Trainer to train Efficiens' logistic regression classifiers. This process is described as follows. First, we apply Trainer to extract relation instances from a sentence. These instances are provided to Efficiens to be used as ground truth. Efficiens generates all candidate predicates and arguments from a sentence as described in Section 7.2.1 and tries to find these candidates in the ground truth instances. A candidate found in the ground truth is labelled with a class according to its type as defined in the ground truth instance. A candidate not found in the ground truth is labelled "none".

Efficiens uses the machine learning library LIBLINEAR [29] in conjunction with Weka framework [41]. The features used for detecting predicates and arguments are no different from most unlexicalized ORE systems (recall Section 2.3.2). Appendix B discusses the features used by Efficiens.

7.4 Choosing sentences for the α and β subsets

Now, we discuss how Efficiens chooses sentences for the α and β subsets. After extracting instances from a sentence, each Efficiens module calculates an *improvement score*. A sentence with a high improvement score is expected to describe instances that can be improved with the additional information provided by deep NLP tools. Therefore, the α and β subsets are composed by sentences with the highest scores.

Efficiens chooses the α and β subsets from many small, in-memory lists of sentences, which is much cheaper than working with a large list of sentences that do not fit memory. For this, Efficiens uses a buffer of fixed size N. Before sentences are sent to Efficiens[POS], they are stored in this buffer. When the buffer is full, the buffered sentences are sent to Efficiens[POS]. Once Efficiens[POS] processes all buffered sentences, it calculates the improvement score for each sentence. The top $\alpha \times N$ sentences are chosen for the α subset. Once Efficiens[DEP] processes these sentences, it calculates a new improvement score for them. Finally, the top $\beta \times N$ sentences are chosen for the β subset and sent to Efficiens[SRL].

One measure used in our improvement score is the number of extracted instances with exactly 1 argument. This measure comes from the observation that Efficiens sometimes filters out instances missing an argument. With the additional information provided by deep NLP tools, Efficiens is often able to recognize the missing argument.

Another measure in our improvement score is the number of low-confidence classifications when detecting predicates and arguments. We define the confidence of a classification as the difference between the probability for the best class (e.g., verb+noun) and the probability of the second best class (e.g., copula+noun), as returned by the logistic regression classifiers. To calculate this measure, we use the classifications for both predicate types and argument types as discussed in Section 7.2.1.

The improvement score for a sentence s_i is defined as follows:

$$\textit{improvement}(s_i) = w_1 \cdot I(s_i) + w_2 \cdot D(s_i, c) \tag{7.1}$$

where w_j is a weight, $I(s_i)$ is the number of instances extracted from s_i with exactly 1 argument and $D(s)$ is the number of classifications for s_i with confidence equal to or lower than c. In our experiments, we empirically choose $w_1 = 0.9$, $w_2 = 0.1$ and $c = 0.4$ since these values performed the best in our development dataset. We tested 9 combinations for w_1 and w_2 by adding increments of 0.1 to w_1 until 0.9 and setting w_2 to $1 - w_1$. We performed a similar test to choose the value of c. Other possible features such as sentence length, total number of extracted instances and average confidence score for all classifications were tested but failed to increase the performance of the improvement score.

In the next section, we evaluate Efficiens and our improvement score. We will show that our score consistently outperforms random selection.

7.5 Experiments

7.5.1 Comparison with ORE methods

Setup. To evaluate Efficiens, we use three datasets introduced in Chapter 6: NYT-500, WEB-500 and PENN-100. Recall that each sentence in a dataset is annotated with two entities and, if this sentence describes an instance involving them, a predicate is also annotated. We also use the concept of dominance. To recall, we say that method A dominates method B if A is: (1) *more effective* and as efficient as B; (2) *more efficient* and as effective as B; or (3) both more effective and more efficient than B. The methods that are not dominated by any other form the state-of-the-art.

As before, the metrics used in this analysis are precision (P), recall (R) and f-measure (F):

$$P = \frac{\text{\# correctly extracted instances}}{\text{\# extracted instances}}, R = \frac{\text{\# correctly extracted instances}}{\text{\# annotated instances}}, \quad (7.2)$$

$$F = \frac{2PR}{P+R} \quad (7.3)$$

We also measure the total computing time of each method, excluding initialization or loading any libraries or models in memory. To ensure a fair comparison, we make sure each method runs in a single-threaded mode, thus utilizing a single computing core at all times.

Methods. In Chapter 6, we evaluated the f-measure and computational time of existing ORE methods. We repeat this evaluation here, now including two new methods: Efficiens and Lund2-IE. Lund2-IE is a new method based on the SRL-IE approach that relies on Lund2. Lund2 is also the SRL system used by Efficiens (recall Section 7.2).

Results. Figure 7.4 compares Efficiens with other ORE methods. Unlike previous methods, the parameters α and β can change the effectiveness and efficiency of

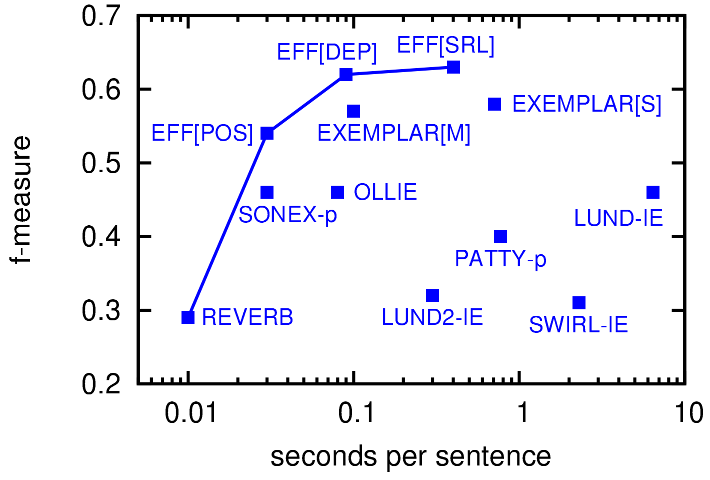


Figure 7.4: Average f-measure vs average time for NYT-500, WEB-500 and PEN-100. EFF[POS], EFF[DEP] and EFF[SRL] represent the performance of each Efficiens module when applied for all sentences.

our method. Therefore, Efficiens cannot be represented by a single plot. In this experiment, we show three plots for Efficiens, one for each module. Efficiens[POS] applies POS tagging only ($\alpha = 0.0, \beta = 0.0$). Efficiens[DEP] applies dependency parsing for all sentences ($\alpha = 1.0, \beta = 0.0$), and Efficiens[SRL] applies semantic role labeling for all sentences ($\alpha = 1.0, \beta = 1.0$).

The lines in the plot identify the state-of-the-art methods. ReVerb is the fastest method and therefore is not dominated by any other method. Efficiens[POS] dominates SONEX-p, OLLIE, PATTY-p, SwiRL-IE and Lund-IE. Finally, Efficiens[DEP] dominates EXEMPLAR[M], while Efficiens[SRL] dominates EXEMPLAR[S].

Discussion. Efficiens shows a gain in effectiveness over methods using similar features. In particular, Efficiens[POS] is more effective than SONEX-p (both use Stanford POS tagger) and Efficiens[DEP] is more effective than EXEMPLAR[M] (both use the Malt dependency parser). These gains in effectiveness show the power of machine learning over manually crafted rules.

Efficiens is also more efficient than rule-based methods using the same features. This is because Efficiens does not require a NER system and the time required to apply classifiers is lower than the time of running the Stanford NER. In addition, the

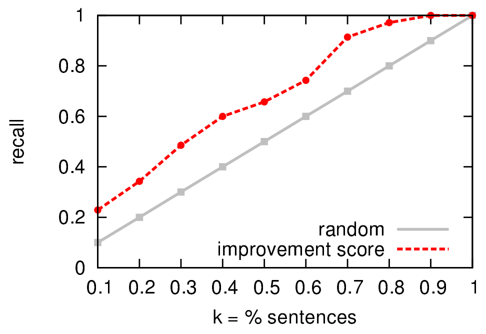
SRL system used by Efficiens (Lund2) is much faster than the other SRL systems in our comparison.

When compared with its previous version (Lund-IE), Lund2-IE presents higher efficiency but lower effectiveness. Analyzing the output of this method, we found that Lund2-IE extracted very few predicates triggered by a noun. We believe that the root cause of this problem in the SRL training data. NomBank often annotates the same noun as both the predicate and the argument. For instance, the example for the predicate “administrator” in NomBank documentation is “Connecticut’s chief court administrator”, where “administrator” is both the predicate and the argument representing the job holder. This is the case for most nouns classified as ACTREL (relational nouns with beneficiaries) and DEFREL (relational nouns for personal relationships). While this annotation style is common in NomBank, it is not useful for ORE. Despite using the same training data, Lund-IE seems to avoid tagging the same noun as both predicate and argument. This seems to be the reason for its higher effectiveness.

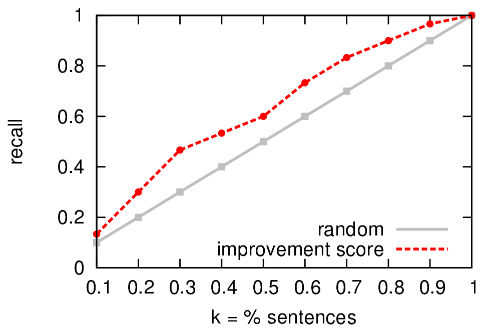
Our experiments indicate that dependency parsing and SRL can contribute to improving the quality of extracted instances. However, SRL’s contribution is much lower than the contribution of dependency parsing. Regarding efficiency, running a SRL system on top of a dependency parser is much more cheaper than we expected. To see this, recall that Lund2 is not compatible with the Malt parser used in Efficiens[DEP]. Therefore, Efficiens[SRL] employs Lund2’s dependency parser. As it turns out, over 90% of the computational cost of Efficiens[SRL] can be attributed to Lund2’s dependency parser. Hence, Lund2’s SRL module requires only a fraction of the time needed to generate the dependency trees they use as input.

7.5.2 Evaluation of our improvement score

Now, we evaluate our improvement score on the task of prioritizing sentences to be processed by Efficiens[DEP] (the α subset) and the sentences to be processed by Efficiens[SRL] (the β subset).



(a) Recall of the ranking used to choose the α subset.



(b) Recall of the ranking used to choose the β subset.

Figure 7.5: Recall of the rankings used to choose the α and β subsets (dashed line). Each k represents a percentage of sentences at the top of the ranking. We also plot the expected recall fora random ranking (solid line).

Setup. The goal of this experiment is to evaluate the quality of the ranking defined by our improvement score. For this, we applied every module of the Efficiens pipeline to the NYT-500 dataset and identified the set of sentences containing improved instances. In particular, we identified the set G_1 of sentences containing instances that were improved when comparing the output of Efficiens[POS] and Efficiens[DEP]. Similarly, we identified the set G_2 of sentences containing instances that were improved when comparing the output of Efficiens[DEP] and Efficiens[SRL]. Observe that G_1 and G_2 are the ground truth for the α and β subsets, respectively. That is, a perfect α subset (or β subset) is equal to G_1 (or G_2).

We evaluate two rankings in separate: (1) the ranking used to choose the α subset, whose ground truth is G_1 and (2) the ranking used to choose the β subset, whose found truth is G_2. A perfect ranking lists every sentence in G_i before listing any other sentence.

The quality of a ranking is measured by recall at k [61]:

$$\text{Recall at } k = \frac{\text{number of sentences in the top } k \text{ that belong to } G_i}{\text{number of sentences in } G_i} \tag{7.4}$$

Results. Figure 7.5 shows the recall of our rankings for various values of k (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1). This figure also shows the expected recall of a random ranking (i.e., the recall value averaged over a large number of random rankings). We observe that our rankings outperform random rankings by a large margin. The recall improvement of using our rankings over random rankings for the values of $k < 1$ (the recall value for $k = 1$ is 1 for any ranking) is on average 48% for the choosing the $alpha$ subset and 28% for choosing the β subset.

Discussion. The results shown in Figure 7.5 indicate that using our improvement score for selecting the α and β subsets is often a better choice than randomly choosing these subsets. Next section explores the overall impact of our improvement score to the effectiveness and efficiency of our method.

7.5.3 Performance of Efficiens with our improvement score

Setup. This experiment measures the effectiveness and efficiency of Efficiens for 31 different budgets. Each budget is a combination of values for α and β, selected from following list: 0, 0.2, 0.4, 0.6, 0.8, 1. We compare Efficiens' performance when using our improvement score and its performance when using a random score. Given the results shown in Figure 7.5, we expect Efficiens to produce better relation instances when using our improvement score as opposed to a random score for a given budget.

Results. Table 7.1 shows the f-measure (effectiveness) and time per sentence (efficiency) of Efficiens for 31 budgets (combinations of values for α, β) when using our improvement score and a random score. For each budget, the value of f-measure and time is averaged over 10 runs for both scores. In total, we computed 310 runs per score. We highlight in bold the highest f-measure value for budget when comparing our score with the random score. Observe that this table shows only one budget with $\alpha = 0$, since we can only run SRL over sentences that have been through dependency parsing. In addition, the f-measure value presented by both scores must to be equal for the following budgets: $(0, 0)$, $(1, 0)$ and $(1, 1)$. This is because these combinations require all sentences to be processed by Efficiens[POS], Efficiens[DEP] and Efficiens[SRL], respectively. Since all sentences are processed, the ordering of the sentences does not affect the result.

α	β	random		improvement score	
		f-measure	time	f-measure (gain)	time
0	0	0.461	0.017	0.461 (0%)	0.017
0.2	0	0.484	0.030	**0.515** (6%)	0.033
0.2	0.2	0.489	0.041	**0.521** (6%)	0.048
0.2	0.4	0.478	0.044	**0.523** (9%)	0.057
0.2	0.6	0.492	0.052	**0.524** (7%)	0.069
0.2	0.8	0.491	0.058	**0.522** (6%)	0.083
0.2	1	0.493	0.064	**0.537** (9%)	0.094
0.4	0	0.515	0.035	**0.558** (8%)	0.038
0.4	0.2	0.518	0.050	**0.567** (9%)	0.063
0.4	0.4	0.515	0.063	**0.570** (11%)	0.080
0.4	0.6	0.521	0.076	**0.576** (11%)	0.097
0.4	0.8	0.520	0.089	**0.576** (11%)	0.115
0.4	1	0.535	0.098	**0.587** (10%)	0.130
0.6	0	0.544	0.039	**0.557** (3%)	0.043
0.6	0.2	0.546	0.060	**0.568** (4%)	0.074
0.6	0.4	0.536	0.081	**0.581** (8%)	0.097
0.6	0.6	0.564	0.096	**0.592** (5%)	0.121
0.6	0.8	0.562	0.114	**0.611** (9%)	0.138
0.6	1	0.567	0.133	**0.624** (10%)	0.161
0.8	0	0.566	0.043	**0.589** (4%)	0.046
0.8	0.2	0.573	0.070	**0.598** (5%)	0.089
0.8	0.4	0.581	0.094	**0.629** (8%)	0.115
0.8	0.6	0.581	0.118	**0.639** (10%)	0.143
0.8	0.8	0.590	0.140	**0.639** (8%)	0.164
0.8	1	0.595	0.166	**0.636** (7%)	0.183
1	0	0.589	0.047	0.589 (0%)	0.047
1	0.2	0.585	0.081	**0.603** (3%)	0.098
1	0.4	0.605	0.111	**0.631** (4%)	0.134
1	0.6	0.615	0.142	**0.631** (3%)	0.156
1	0.8	0.628	0.170	**0.638** (2%)	0.184
1	1	0.627	0.198	0.627 (0%)	0.200

Table 7.1: F-measure and time (seconds per sentence) for different values of α and β.

Observe that our improvement score achieves greater f-measure than random selection for all budgets where the order of the sentences can affect Efficiens' performance (i.e., excluding the tree budgets mentioned earlier). However, the sentences selected with our score often take more time to process than those chosen randomly. This is because our score tends to assign higher scores to longer sentences and these sentences require more time to be parsed by NLP tools.

To understand the actual impact of our improvement score to Efficiens' performance, we will now discuss a different way to present the values shown in Table 7.1. For this, we selected several values of f-measure as shown in Table 7.2. For each

f-measure	random	improvement score	increase
0.510	28.6	30.3	6.1%
0.515	25.6	26.3	2.6%
0.520	25.6	26.3	2.6%
0.525	25.6	26.3	2.6%
0.530	25.6	26.3	2.6%
0.535	25.6	26.3	2.6%
0.540	25.6	26.3	2.6%
0.545	23.3	26.3	13.2%
0.550	23.3	26.3	13.2%
0.555	23.3	26.3	13.2%
0.560	23.3	21.7	-6.5%
0.565	23.3	21.7	-6.5%
0.570	21.3	21.7	2.2%
0.575	21.3	21.7	2.2%
0.580	21.3	21.7	2.2%
0.585	21.3	21.7	2.2%
0.590	9.0	11.2	24.7%
0.595	9.0	11.2	24.7%
0.600	9.0	10.2	13.3%
0.605	9.0	8.7	-3.5%
0.610	7.0	8.7	23.5%
0.615	7.0	8.7	23.5%
0.620	5.9	8.7	47.8%
0.625	5.9	8.7	47.8%
average			10.8%

Table 7.2: Throughput (i.e., the number of sentences processed per second) for several levels of f-measure.

f-measure value and choice of score, we found the minimum time required to reach (or surpass) this value in Table 7.1. For example, the minimum time required to reach or surpass a f-measure value of 0.6 when using the random score is 0.111, which can be found in the row of the budget $(\alpha = 1, \beta = 0.4)$. Then, we calculated Efficiens' throughput (i.e., number of sentences processed per second) for each value of f-measure, where throughput = $\frac{1}{\text{time}}$.

Discussion. Table 7.2 shows the increase in Efficiens' throughput when using our score as opposed to the random score. On average, our score increases Efficiens' throughput by 10.8% while keeping the same level of effectiveness. This increase is more accentuated for higher level of effectiveness (f-measure > 0.59). Due to our score, Efficiens was able to output 47.8% more sentences at the highest level of f-measure (0.625). These results indicate that carefully selecting sentences to be processed by more expensive NLP tools is a promising research direction.

7.6 Conclusion

We presented Efficiens, an ORE method that takes as input a budget, which limits its ability to run sophisticated (and expensive) NLP tools over all sentences from a corpus. Efficiens respects the user budget by leveraging a modular architecture, where each module exploit one of the following tools: POS tagging, dependency parsing and SRL. These modules work in a pipeline, where a sentence can go throughout all modules or stop after any module. This allows Efficiens to allocate more or less time to a sentence according to this sentence's likelihood of improving the final set of extracted instances. Our experiments show that our intelligent selection of sentences is better than random selection. Our selection increased the throughput of Efficiens by 11% on average while keeping the same level of effectiveness as the random selection. This result shows that allocation of computational resources is a promising solution for more efficient ORE methods.

Our experiments show that Efficiens dominates every ORE method but ReVerb, which is faster but not as accurate. These results show that leveraging readily available training data for SRL is a solution for the lack of training data for ORE.

By combining POS tagging, dependency parsing and SRL within a single framework, Efficiens allows one to measure the real contribution of these features for ORE. In our experiments, dependency parsing shows an improvement of 15% (0.62 up from 0.54) in f-measure over POS tagging. On the other hand, SRL improves f-measure by only 2% when compared to dependency parsing.

A reason for the low contribution of SRL features is as follows. SRL systems use much of the same features as Efficiens, that is, the features extracted from POS tagging and dependency parsing. Therefore, including the SRL output as a feature do not provide much additional information. A similar phenomenon is known to happen in NER. Adding POS tags as features results in little improvement since NER and POS tagging uses much of the same features [38].

Chapter 8

Conclusion

This thesis explored the trade-off between effectiveness and efficiency for open relation extraction (ORE), the task of extracting unknown relations from large corpora. For this, we presented new benchmarks for ORE enabling a fair and objective evaluation of a wide range of methods. In particular, we tested 11 methods, which can be grouped by their underlying NLP tools: part-of-speech (POS) tagger, dependency parser and semantic role labeler (SRL). Our benchmarks are fair since they do not penalize a method for including or excluding non-essential tokens in a predicate. In addition, we introduced four ORE methods: SONEX, Meta-CRF, EXEMPLAR and Efficiens.

Our experiments indicate that dependency parsing offers a significant contribution to increasing ORE's effectiveness when compared to POS tagging alone. Our experiments also indicate that SRL's contribution to ORE's effectiveness is limited or even insignificant when compared to POS tagging and dependency parsing together. We conjecture that this is because SRL offers little additional information about a sentence's structure since SRL uses the same features (POS tagging and dependency parsing) as ORE.

We expect the relative effectiveness achieved by our experimental methods to be consistent for other corpora. This is in part because when comparing two methods using the *same underlying NLP tool* (e.g., EXEMPLAR[M] vs. Efficiens[DEP]), the method with higher effectiveness (e.g., Efficiens[DEP]) often contains the instances correctly extracted by the method with lower effectiveness (e.g., EXEMPLAR[M]). Moreover, this observation also applies to methods using the *same implementation* *but different features* (e.g., Efficiens[POS] vs. Efficiens[DEP]); that is, the method with higher effectiveness (e.g., Efficiens[DEP]) often contains the instances correctly extracted by the method with lower effectiveness (e.g., EXEMPLAR[M]). In both cases, higher effectiveness is achieved by recognizing additional instances presenting a more complex structure in text without missing the extraction of instances with simpler structures. Therefore, we expect our experimental methods' *relative effectiveness* to be consistent for corpora presenting various levels of text complexity.

Another conclusion from this work is that the computational cost of an ORE method is dominated by its underlying NLP machinery. This can be seen in our experiments, where methods using the same NLP tool present similar cost. This held even for methods whose approaches are vastly different (e.g., Efficiens[POS] and SONEX-p). This means that cost differences among our experimental methods are due to the use of different NLP tools, as opposed to the quality of a method's implementation. This observation helps corroborate the validity of our experiments.

We have discovered a promising direction to increase ORE's efficiency. By comparing the instances extracted by Efficiens[POS] and Efficiens[SRL], we discovered that most sentences (93% in our corpus) do not benefit from dependency parsing and SRL. That is, for 93% of the sentences, the instances extracted from a sentence using POS tagging were not improved after applying dependency parsing and SRL over this same sentence. This happens when a sentence: (1) does not describe any instances, (2) describes instances that were not correctly extracted with either Efficiens[POS] or Efficiens[SRL], and (3) describes instances that were correctly extracted by both methods. This means that running Efficiens[SRL] for all sentences of a corpus would result in a great waste of resources. To address this problem, we proposed an intelligent method as part of Efficiens (the improvement score) for choosing what features to extract from each sentence.

8.1 Method-specific observations

SONEX relies on a clustering to group entity pairs into different relations. This clustering-based approach is one of the fastest solutions for ORE, since it can work by applying sentence splitting and tokenization only. SONEX can optionally leverage POS tagging and work at a higher effectiveness level with an additional computation cost. As shown in our experiments, clustering works well for pair of entities occurring together in many sentences. On the other hand, clustering methods are limited in two aspects: (1) they may not work as well for pair of entities occurring in few sentences, since features tend to be scarce in this case; and (2) clustering is a poor choice for extracting n-ary and nested relations. Despite the limitations of clustering, SONEX's patterns for the extraction of predicates turned out to be quite effective as a stand-alone method called SONEX-p.

SONEX-p and our remaining methods (Meta-CRF, EXEMPLAR and Efficiens) adopt the single-pass, sentence-level extraction introduced by the seminal work of TextRunner [6]. This framework does not share the aforementioned limitations of clustering-based methods; however, it relies on NLP tools that require additional computational time. Meta-CRF exploits full parsing to extract nested relations. In our experiments, full parsing improves not only the quality of nested relations, but it also improves the quality of flat, binary relations (by way of reducing false positives).

EXEMPLAR addresses the problem of extracting n-ary relations by using handcrafted rules over dependency trees. EXEMPLAR loosely follows the approach of SRL methods by detecting the particular role of each argument in a relation instance. We argue that this approach is superior to detecting all arguments of an instance at once. This is because our approach allows for fewer rules, which are simpler and can be applied to a greater number of sentences.

Efficiens incorporates the strengths of the aforementioned methods in one framework capable of exploiting POS tagging, dependency parsing and SRL. While today's ORE methods apply the same steps for every sentence, Efficiens decides which NLP tool to use for each individual sentence. Efficiens allows the users to provide a budget that constraints the use of dependency parsing and SRL to a fraction of the sentences only. By carefully selecting the subset of sentences to be processed by these tools, Efficiens is able to process on average 11% more sentences per second than if the sentences were randomly chosen while keeping the same level of effectiveness.

8.2 Directions for future work

8.2.1 Improving Efficiency

From a practical point of view, most organizations dealing with relation extraction cannot afford running expensive extractors for all sentences. Therefore, the problem of maximizing effectiveness within computational constraints is a promising research direction for ORE. While this thesis has made some contributions to increase the effectiveness and efficiency of ORE, much work is needed to advance this research area. We propose directions for future work as follows.

Sentence selection for further processing. A promising direction to increase the efficiency of ORE is to find the small subset of sentences (7% in our corpus) that lead to increased effectiveness when processed by more expensive NLP tools (dependency parsing and SRL). A method addressing this problem must not introduce significant overhead to the process of extracting instances. Otherwise, this method's cost may detract from its potential impact on the overall effectiveness. A potentially better solution than our improvement score (recall Chapter 7.4) is to use a supervised learning approach. This approach can leverage features already extracted for relation extraction (see Appendix B) to estimate a sentence's likelihood to benefit from further processing. The training examples for this approach require no human annotation, since examples can be obtained from running two Efficiens modules (e.g., those based on POS tagging and dependency parsing, respectively) over a sentence and automatically checking whether its extracted instances are the same. A sentence whose extracted instances are the same for both modules should rank lower than a sentence whose extracted instances are different.

Text simplification. Another promising direction to increase ORE's efficiency is text simplification [83]. ORE methods often struggle to efficiently extract relations from long, complex sentences. An approach to solve this problem is to re-structure a complex sentence and replacing it by several others that, together, convey the same meaning and are more amenable for ORE. This approach has the potential to reduce computational time (while maintaining effectiveness) by feeding shorter, simpler sentences to deep NLP tools. This is because these tools are likely to spend less time parsing parse several short, simple sentences than the original, complex one [83].

Efficient NLP tools. Given that the cost of extracting features using NLP tools dominates the cost of most ORE methods, one way to improve the efficiency of these methods is to improve the efficiency of their underlying NLP tools.

8.2.2 Improving Effectiveness

Other directions for future work concerning ORE's effectiveness are as follows.

Additional predicate types. Our methods have focused on extracting predicates triggered by verbs and nouns. However, many predicates are triggered by adjectives. For an example, consider the sentence:

> "Georgia is <u>close</u> to Russia".

where "close" is an adjective and one of the triggers of the copula+adjective predicate "is close". Recognizing predicates of this type is a way to increase recall of current ORE methods.

Extraction across sentences. Like most work in this area, we have focused on extracting instances whose arguments appear in a single sentence. There is a need for methods that can detect instances whose arguments are scattered across several sentences. One way to detect an argument appearing in a different sentence than the predicate is to leverage anaphora resolution and detect a pronoun referring to this argument in the predicate's sentence. There is also a need for ORE benchmarks that annotate instances whose arguments cross sentence boundaries.

Cross document extraction. A challenge for full-fledge ORE systems is to recognize and merge instances that refer to the same event or relationship but were extracted from different documents. By merging duplicates, these systems improve the quality of the extracted instances by combining arguments scattered across multiple documents. This process involves the problems of recognizing synonym predicates (e.g., "is subsidiary" and "acquired") and determining which syntactic roles (e.g., `subject`, `direct_object`, `prep_object`) correspond to the same semantic role (e.g., "parent company") across different instances [16].

Extraction of facts. Today's ORE methods cannot differentiate among instances describing facts, opinions, rumours, and others. Being able to separate fact from fiction is essential to many applications such as knowledge base population and question answering. Recent work has studied tools like T-Verifier [52], which search the web for evidence of truthfulness. Another direction is to measure truthfulness from the writing style, source authority and other signals intrinsic to the textual source. Wu et al. [93] propose a framework for computational fact checking by leveraging databases as opposed to text. This framework models a claim as a query over a database and measures truthfulness by find counterarguments, that is, alternative queries that weaken the original claim. This framework could be used to verify relation instances extracted from text by converting them into queries over existing online databases.

ORE evaluation. Finally, there is a need for benchmarks capable of measuring precision and recall of a full-fledged ORE system *in large scale*. Large scale is essential since many ORE systems (e.g., those based on clustering) rely on redundancy, that is, having a relation instance being described in many documents. Since manually annotating every instance described in millions of documents is virtually impossible, alternative methods are required. This thesis introduced some alternatives; however, we have only scratched the surface of this problem.

Bibliography

[1] Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting Relations from Large Plain-Text Collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries (DL 2000)*. ACM Press, New York, NY, 85–94. DOI:http://dx.doi.org/10.1145/336597.336644

[2] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. 2009. A Comparison of Extrinsic Clustering Evaluation Metrics Based on Formal Constraints. *Inf. Retr.* 12, 4 (August 2009), 461–486. DOI:http://dx.doi.org/10.1007/s10791-008-9066-8

[3] Gabor Angeli and Christopher D. Manning. 2014. NaturalLI: Natural Logic Inference for Common Sense Reasoning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP '14)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 534 –545.

[4] Amit Bagga and Breck Baldwin. 1998. Entity-based Cross-Document Coreferencing Using the Vector Space Model. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1 (ACL '98)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 79–85. DOI: http://dx.doi.org/10.3115/980845.980859

[5] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open Information Extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artifical intelligence (IJCAI '07)*. AAAI Press, Menlo Park, CA, USA, 2670–2676.

[6] Michele Banko and Oren Etzioni. 2008. The Tradeoffs Between Open and Traditional Relation Extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, 28–36.

[7] Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing Guidelines for Treebank II Style Penn Treebank Project. (Jan. 1995). http://languagelog.ldc.upenn.edu/myl/PennTreebank1995.pdf

[8] Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A High-performance Syntactic and Semantic Dependency Parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations (COLING '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 33–36.

[9] Bernd Bohnet. 2010. Very High Accuracy and Fast Dependency Parsing is Not a Contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 89–97.

[10] Kalina Bontcheva, Marin Dimitrov, Diana Maynard, Valentin Tablan, and Hamish Cunningham. 2002. Shallow Methods for Named Entity Coreference Resolution. In *Chaines de References et Resolveurs d'Anaphores, Workshop (TALN '02)*. ATALA, Paris, France.

[11] Sergey Brin. 1998. Extracting Patterns and Relations from the World Wide Web. In *Selected Papers from the International Workshop on The World Wide Web and Databases (WebDB '98)*. ACM Press, New York, NY, 172–183.

[12] Razvan C. Bunescu and Raymond J. Mooney. 2005. A Shortest Path Dependency Kernel for Relation Extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, Raymond J. Mooney (Ed.). Association for Computational Linguistics, Stroudsburg, PA, USA, 724–731.

[13] K. Burton, A. Java, and I. Soboroff. 2009. The ICWSM 2009 Spinn3r Dataset. In *Proceedings of the Third Annual Conference on Weblogs and Social Media (ICWSM '09)*. AAAI Press, Menlo Park, CA, USA.

[14] David Carmel, Haggai Roitman, and Naama Zwerdling. 2009. Enhancing Cluster Labeling Using Wikipedia. In *Proceedings of the 32nd international ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)*. ACM Press, New York, NY, USA, 139–146.

[15] Marie Catherine de Marneffe and Christopher D. Manning. 2008. *Stanford typed dependencies manual*. Technical Report. Stanford University, The Stanford Natural Language Processing Group.

[16] Nathanael Chambers and Dan Jurafsky. 2011. Template-Based Information Extraction without the Templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1 (HLT '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 976–986.

[17] Tao Cheng, Xifeng Yan, and Kevin Chen-Chuan Chang. 2007. EntityRank: Searching Entities Directly and Holistically. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB '07)*. VLDB Endowment, San Jose, CA, USA, 387–398.

[18] Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2010. Semantic Role Labeling for Open Information Extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading (FAM-LbR '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 52–60.

[19] Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An Analysis of Open Information Extraction based on Semantic Role Labeling. In *Proceedings of the sixth international conference on Knowledge capture*. ACM Press, New York, NY, USA, 113–120. DOI:http://dx.doi.org/10.1145/1999676.1999697

[20] CNN. 2008. McCain ad compares Obama to Britney Spears, Paris Hilton. `http://www.cnn.com/2008/POLITICS/07/30/mccain.ad`. (2008). [Online; accessed 16-August-2010].

[21] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. 2000. Learning to Construct Knowledge Bases from the World Wide Web. *Artif. Intell.* 118, 1-2 (2000), 69–113.

[22] Silviu Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *The 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 708–716.

[23] Aron Culotta and Jeffrey S. Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL '04)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 423–429.

[24] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. 1992. Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '92)*. ACM Press, New York, NY, USA, 318–329. `DOI:http://dx.doi.org/10.1145/133160.133214`

[25] Nilesh Dalvi, Ashwin Machanavajjhala, and Bo Pang. 2012. An Analysis of Structured Data on the Web. *Proc. VLDB Endow.* 5, 7 (2012), 680–691.

[26] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-Scale Information Extraction in Knowitall: (Preliminary Results). In *Proceedings of the 13th International Conference on World Wide Web (WWW '04)*. ACM Press, New York, NY, USA, 100–110. `DOI:http://dx.doi.org/10.1145/988672.988687`

[27] Richard Evans. 2003. A Framework for Named Entity Recognition in the Open Domain. In *Proceedings of the Recent Advances in Natural Language Processing (RANLP '03)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 137–144.

[28] Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying Relations for Open Information Extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1535–1545.

[29] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research* 9 (2008), 1871–1874.

[30] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building Watson: An overview of the DeepQA project. *AI Magazine* 31, 3 (2010), 59–79.

[31] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-Local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 363–370. DOI:http://dx.doi.org/10.3115/1219840.1219885

[32] David Fisher, Stephen Soderland, Fangfang Feng, and Wendy Lehnert. 1995. Description of the UMass System as Used for MUC-6. In *Proceedings of the 6th conference on Message understanding (MUC6 '95)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 127–140. DOI:http://dx.doi.org/10.3115/1072399.1072412

[33] Joseph L. Fleiss, Bruce Levin, and Myunghee Cho Paik. 2003. *Statistical Methods for Rates and Proportions* (third ed.). John Wiley & Sons, New York.

[34] Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI '07)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1606–1611.

[35] Eric J. Glover, Kostas Tsioutsiouliklis, Steve Lawrence, David M. Pennock, and Gary W. Flake. 2002. Using Web Structure for Classifying and Describing Web Pages. In *Proceedings of the 11th International Conference on World Wide Web (WWW '02)*. ACM Press, New York, NY, USA, 562–569. DOI:http://dx.doi.org/10.1145/511446.511520

[36] David A. Grossman and Ophir Frieder. 2004. *Information Retrieval: Algorithms and Heuristics* (2. ed.). Springer, New York, NY, USA.

[37] Barbara J. Grosz, Douglas E. Appelt, Paul A. Martin, and Fernando C. N. Pereira. 1987. TEAM: An Experiment in the Design of Transportable Natural-language Interfaces. *Artif. Intell.* 32, 2 (May 1987), 173–243. DOI:http://dx.doi.org/10.1016/0004-3702(87)90011-7

[38] The Stanford Natural Language Processing Group. 2010. Stanford NER CRF FAQ. http://nlp.stanford.edu/software/crf-faq.shtml. (2010). Accessed: 2014-07-24.

[39] Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring Various Knowledge in Relation Extraction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 427–434. DOI:http://dx.doi.org/10.3115/1219840.1219893

[40] Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task (CoNLL '09)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1–18.

[41] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.* 11, 1 (Nov. 2009), 10–18. DOI:http://dx.doi.org/10.1145/1656274.1656278

[42] Robert Hanneman and Mark Riddle. 2005. *Introduction to Social Network Methods*. University of California, Riverside, Riverside, CA. http://faculty.ucr.edu/~hanneman/nettext/

[43] Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering Relations among Named Entities from Large Corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, 415. DOI:http://dx.doi.org/10.3115/1218955.1219008

[44] Marti A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2 (COLING '92)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 539–545. DOI:http://dx.doi.org/10.3115/992133.992154

[45] Richard Johansson and Pierre Nugues. 2008. Dependency-Based Semantic Role Labeling of PropBank. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 69–78.

[46] Karen Sparck Jones and Peter Willett (Ed.). 1997. *Readings in Information Retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[47] Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing* (2 ed.). Prentice Hall, Upper Saddle River, NJ, USA.

[48] Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics* (2nd edition ed.). Prentice-Hall, Upper Saddle River, NJ, USA.

[49] Nanda Kambhatla. 2004. Combining Lexical, Syntactic and Semantic Features with Maximum Entropy Models. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, Article 22, 4 pages.

[50] Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1 (ACL '03)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 423–430. DOI:http://dx.doi.org/10.3115/1075096.1075150

[51] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the International Conference on Machine Learning (ICML)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 282–289.

[52] Xian Li, Weiyi Meng, and Clement Yu. 2011. T-Verifier: Verifying Truthfulness of Fact Statements. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering (ICDE '11)*. IEEE Computer Society, Washington, DC, USA, 63–74. DOI:http://dx.doi.org/10.1109/ICDE.2011.5767859

[53] Dekang Lin and Patrick Pantel. 2001. DIRT–Discovery of Inference Rules from Text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '01)*. ACM, New York, NY, USA, 323–328. DOI:http://dx.doi.org/10.1145/502512.502559

[54] Thomas Lin, Mausam, and Oren Etzioni. 2010. Identifying Functional Relations in Web Text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1266–1276. http://dl.acm.org/citation.cfm?id=1870658.1870781

[55] Philadelphia Linguistic Data Consortium. 2007. Message Understanding Conference (MUC) 7. http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2001T02. (2007).

[56] Oded Maimon and Lior Rokach. 2005. *The Data Mining and Knowledge Discovery Handbook.* Springer, New York, NY, USA.

[57] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval* (1 ed.). Cambridge University Press, Cambridge, England.

[58] Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the Workshop on Human Language Technology (HLT '94)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 114–119. DOI:http://dx.doi.org/10.3115/1075812.1075835

[59] Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open Language Learning for Information Extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CONLL '12)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 523–534.

[60] Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. (2002). http://mallet.cs.umass.edu

[61] Frank McSherry and Marc Najork. 2008. Computing Information Retrieval Performance Measures Efficiently in the Presence of Tied Scores. In *Proceedings of the IR Research, 30th European Conference on Advances in Information Retrieval (ECIR'08)*. Springer-Verlag, Berlin, Heidelberg, 414–421. http://dl.acm.org/citation.cfm?id=1793274.1793325

[62] Yuval Merhav, Filipe Mesquita, Denilson Barbosa, Wai Gen Yee, and Ophir Frieder. 2012. Extracting Information Networks from the Blogosphere. *ACM Trans. Web* 6, 3 (2012), 11.

[63] Filipe Mesquita. 2012. Clustering Techniques for Open Relation Extraction. In *Proceedings of the on SIGMOD/PODS 2012 PhD Symposium (PhD '12)*. ACM Press, New York, NY, USA, 27–32. DOI:http://dx.doi.org/10.1145/2213598.2213607

[64] Filipe Mesquita and Denilson Barbosa. 2011. Extracting Meta Statements from the Blogosphere. In *Proceedings of the Fifth International Conference on Weblogs and Social Media (ICWSM '11)*. AAAI Press, Menlo Park, CA, USA, 225–232.

[65] Filipe Mesquita, Yuval Merhav, and Denilson Barbosa. 2010. Extracting Information Networks from the Blogosphere: State-of-the-Art and Challenges. In *Proceedings of the Fourth International Conference on Weblogs and Social Media, Data Challenge Workshop (ICWSM '10)*. AAAI Press, Menlo Park, CA, USA, Article 3, 8 pages.

[66] Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and Efficiency of Open Relation Extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, 447–457.

[67] Filipe Mesquita, Ying Xu, Aditya Bhargava, Mirko Bronzi, Denilson Barbosa, and Grzegorz Kondrak. 2011. The Effectiveness of Traditional and Open Relation Extraction for the Slot Filling Task at TAC 2011. In *Proceedings of the Fourth Text Analysis Conference*. NIST, Gaithersburg, MD, USA, Article 66, 7 pages.

[68] Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank Project: An Interim Report. In *Proceedings of the NAACL HLT 2004 Workshop on Frontiers in Corpus Annotation*. Association for Computational Linguistics, Stroudsburg, PA, USA, 24–31.

[69] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant Supervision for Relation Extraction without Labeled Data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 (ACL '09)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1003–1011.

[70] David Nadeau and Satoshi Sekine. 2007. A Survey of Named Entity Recognition and Classification. *Lingvisticae Investigationes* 30, 1 (2007), 3–26. DOI: http://dx.doi.org/10.1075/li.30.1.03nad

[71] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: A Taxonomy of Relational Patterns with Semantic Types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL '12)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1135–1145.

[72] NIST. 2008. Automatic Content Extraction 2008 Evaluation Plan (ACE08). http://www.itl.nist.gov/iad/mig/tests/ace/2008/doc/ace08-evalplan.v1.2d.pdf. (2008).

[73] Joakim Nivre and Jens Nilsson. 2004. Memory-Based Dependency Parsing. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL '04)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 49–56.

[74] Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Comput. Linguist.* 31, 1 (March 2005), 71–106. DOI:http://dx.doi.org/10.1162/0891201053630264

[75] Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task (CONLL Shared Task '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1–27. http://dl.acm.org/citation.cfm?id=2132936.2132937

[76] Lev Ratinov and Dan Roth. 2009. Design Challenges and Misconceptions in Named Entity Recognition. In *Proceedings of the Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Stroudsburg, PA, USA, 147–155.

[77] Ronald L. Rivest. 1992. The MD5 Message-Digest Algorithm (RFC 1321). http://www.ietf.org/rfc/rfc1321.txt?number=1321. (1992).

[78] Stephen Robertson. 2004. Understanding Inverse Document Frequency: On Theoretical Arguments for IDF. *Journal of Documentation* 60 (2004), 2004.

[79] Barbara Rosario and Marti A. Hearst. 2004. Classifying Semantic Relations in Bioscience Texts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, 430–437.

[80] Benjamin Rosenfeld and Ronen Feldman. 2007. Clustering for Unsupervised Relation Identification. In *Proceedings of the ACM Conference on Information and Knowledge Management*. ACM Press, New York, NY, USA, 411–418. DOI:http://dx.doi.org/10.1145/1321440.1321499

[81] Evan Sandhaus. 2008. The New York Times Annotated Corpus. https://catalog.ldc.upenn.edu/LDC2008T19. (2008).

[82] Sunita Sarawagi. 2008. Information Extraction. *Found. Trends databases* 1, 3 (March 2008), 261–377. DOI:http://dx.doi.org/10.1561/1900000003

[83] Jordan Schmidek and Denilson Barbosa. 2014. Improving Open Relation Extraction via Sentence Re-Structuring. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis (Eds.). European Language Resources Association (ELRA), Reykjavik, Iceland. http://www.lrec-conf.org/proceedings/lrec2014/summaries/1038.html

[84] Stefan Schoenmackers, Oren Etzioni, Daniel S. Weld, and Jesse Davis. 2010. Learning First-order Horn Clauses from Web Text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1088–1098. http://dl.acm.org/citation.cfm?id=1870658.1870764

[85] S. Soderland, B. Roof, B. Qin, S. Xu, O. Etzioni, and Others. 2010. Adapting Open Information Extraction to Domain-Specific Relations. *AI Magazine* 31, 3 (2010), 93–102.

[86] Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using Predicate-Argument Structures for Information Extraction. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, 8–15.

[87] Niket Tandon, Gerard de Melo, and Gerhard Weikum. 2014. Acquiring Comparative Commonsense Knowledge from the Web. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI Press, Menlo Park, CA, USA, 166–172.

[88] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1 (NAACL '03)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 173–180.

[89] Pucktada Treeratpituk and Jamie Callan. 2006. Automatically Labeling Hierarchical Clusters. In *Proceedings of the 2006 International Conference on Digital Government Research (DG.O '06)*. ACM Press, New York, NY, USA, 167–176. DOI:http://dx.doi.org/10.1145/1146598.1146650

[90] George Tsatsaronis, Iraklis Varlamis, and Michalis Vazirgiannis. 2010. Text Relatedness Based on a Word Thesaurus. *Journal of Artificial Intelligence Research (JAIR)* 37 (2010), 1–39.

[91] W3C. 2010. RDF Primer. http://www.w3.org/TR/rdf-primer/. (December 2010).

[92] Fei Wu and Daniel S. Weld. 2010. Open Information Extraction Using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 118–127.

[93] You Wu, Pankaj K. Agarwal, Chengkai Li, Jun Yang, and Cong Yu. 2014. Toward Computational Fact-Checking. *Proceedings of the VLDB Endowment* 7, 7 (2014), 589–600.

[94] Fei Xia and Martha Palmer. 2001. Converting Dependency Structures to Phrase Structures. In *Proceedings of the International Conference on Human Language Technology Research (HLT '01)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1–5. DOI:http://dx.doi.org/10.3115/1072133.1072147

[95] Ying Xu, Mi-Young Kim, Kevin Quinn, Randy Goebel, and Denilson Barbosa. 2013. Open Information Extraction with Tree Kernels. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, 868–877.

[96] G. Yang and M. Kifer. 2003. Reasoning about Anonymous Resources and Meta Statements on the Semantic Web. *J. Data Semantics* 2800 (2003), 69–97.

[97] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *J. Mach. Learn. Res.* 3 (2003), 1083–1106.

[98] Min Zhang, Jian Su, Danmei Wang, Guodong Zhou, and Chew Lim Tan. 2005. Discovering Relations Between Named Entities from a Large Raw Corpus Using Tree Similarity-Based Clustering. In *Proceedings of the International Joint Conference on Natural Language Processing*. Springer, New York, NY, USA, 378–389.

[99] Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. 2009. StatSnowball: A Statistical Approach to Extracting Entity Relationships. In *Proceedings of the 18th International Conference on World Wide Web (WWW '09)*. ACM Press, New York, NY, USA, 101–110. DOI:http://dx.doi.org/10.1145/1526709.1526724

Appendices

Appendix A

Converting SRL annotations into ORE annotations

To produce training data, Efficiens converts SRL annotations to ORE annotations by applying a set of rules. The rules for detecting predicates are as follows.

- Verb. A predicate with a verb trigger that is a SRL predicate.
- Verb+Noun. A predicate with two triggers, where the verb trigger is a predicate and the noun trigger is an "A1" argument of the verb. In addition, the dependency from the noun to its parent is of the type "OBJ" (direct object).
- Verb+Prep+Noun. A predicate with two triggers, where the verb is a predicate and the noun is an "Ai" argument ($i > 1$) of the verb. In addition, the dependency path from the verb to the noun contains a preposition.
- Copula+Noun. A predicate with one or two triggers. If the predicate has one trigger, it must be a noun and have an inbound or outbound dependency edge of the type "APPO" (apposition). If the predicate has two triggers, the noun trigger must depend on the verb trigger and the dependency type must be "PRD" (Predicative complement).
- Possessive+Noun. A predicate with one trigger, where the trigger is a noun and one of its children is a proper noun whose dependency type is "NMOD" (Modifier of nominal).

The rules for detecting arguments are as follows.

- Subject. If the trigger is a verb and SRL predicate, the argument must satisfy: (1) the argument's SRL role is "A0" or "A1" and (2) the edges in the path from the trigger to the argument must contain the dependencies "SBJ" (Subject), "NMOD" (Modifier of nominal) or "LGS" (Logical subject). If the predicate is a Possessive+Noun, the argument must be a proper noun modifying the noun trigger. If the predicate is a Copula+Noun with one trigger, this trigger must be a noun and in an apposition with the argument. If the predicate is a Copula+Noun with two triggers, the argument must have a "PRD" or "SBJ" dependency with the verb trigger.

- Direct Object. If the trigger is a verb and SRL predicate, the argument must satisfy: (1) the argument's SRL role is "A0" or "A1" and (2) the edges in the path from the trigger to the argument must contain the dependencies "OBJ" or "APPO".

- Prepositional Object. If the trigger is a verb and SRL predicate, the argument's SRL role must be "Ai", $i > 1$. If the trigger is a noun, the argument must modify the trigger with a preposition or have a dependency on the trigger of type "NMOD".

Appendix B

Features Used In Efficiens

Predicate Detection Features

- TriggerCount+IsHead (POS): The number of triggers for the predicate concatenated with a flag indicating whether the noun trigger (if present) is the head of a noun phrase. All POS features below are concatenated with this feature.
- isNominalizedVerb (POS): The noun trigger (if present) is listed as a nominalized verb in WordNet.
- Apposition (POS): A flag indicating whether there is an apposition between the noun trigger and the noun preceding it.
- PosBefore (POS): Sequence of POS tags for the tokens preceding each trigger, including the POS of the trigger itself. Three sequences are generated for each trigger, starting at one token up to three. The POS of a trigger is surround by squared brackets in order to differentiate from other tokens.
- PosAfter (POS): Sequences of POS tags generated as above, now with tokens succeeding each trigger.
- PosSurround (POS): Concatenation of PosBefore and PosSurround.
- PosBetween (POS): Sequence of POS tags for the tokens in between two triggers.
- ParentPos (DEP): The POS tag of the trigger's parent.

- DepType (DEP): The type of dependency between the trigger and its parent.
- Apposition (DEP): A flag indicating whether there is an apposition dependency between a trigger and its parent and/or a trigger and its children.
- DepPathEdges (DEP): The edges in the shortest path between two triggers of a predicate. Each edge is presented by its direction (up or down) and its dependency type.
- IsPredicate (SRL): A flag indicating whether the trigger is a SRL predicate.
- SrlRole(SRL): The SRL role of the noun trigger considering the verb trigger as its SRL predicate. The role ”none” is used when the has no role.
- ParentSrlRole(SRL): The SRL role of noun trigger's parent considering the verb trigger as its SRL predicate.

All sequences of POS tags are normalized by removing adjectives and adverbs and replacing all tags starting with “NN” to “NN”. In this way, singular nouns, plural nouns and proper nouns are all represented by the same POS tag. In addition, continuous sequences of the same POS tag (e.g. “NN-NN-NN”) is replaced by a single POS tag (e.g., “NN”).

Argument Detection Features

- PredicateType+IsHead (POS): The type of the predicate concatenated with a flag indicating whether the argument is the head of a noun phrase. All POS features below are concatenated with this feature.
- Apposition (POS): Is the argument succeeded by a comma, determiner and the predicate's noun trigger.
- PreceededByPreposition (POS): A flag indicating whether the argument is preceded by a preposition.
- PosBetween (POS): Sequence of POS tags for the tokens in between the argument and the trigger.

- Lexicon+PosBetween (POS): A sequence of POS tags as in ”PosBetween”, where lexicons are appended to functional words, such as preposition.

- DepPathEdges (POS): The edges in the shortest path between an argument and a trigger. Each edge is presented by its direction (up or down) and its dependency type.

- SrlRole(SRL): The SRL role of the argument considering the trigger as its SRL predicate. The role ”none” is used when the argument has no role.

- ParentSrlRole(SRL): The SRL role of the argument’s parent considering the trigger as its SRL predicate.