Effectiveness and Efficiency of Open Relation Extraction

Filipe Mesquita **Jordan Schmidek** **Denilson Barbosa**
Department of Computing Science, University of Alberta, Canada
{mesquita,schmidek,denilson}@ualberta.ca

Abstract

A large number of Open Relation Extraction 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). A natural question then is what is the trade-off between NLP depth (and associated computational cost) versus effectiveness. This paper presents a fair and objective experimental comparison of 8 state-of-the-art approaches over 5 different datasets, and sheds some light on the issue. The paper also describes a novel method, EXEMPLAR, which adapts ideas from SRL to less costly NLP machinery, resulting in substantial gains both in efficiency and effectiveness, over binary and n-ary relation extraction tasks.

1 Introduction

Open Relation Extraction (ORE) (Banko and Etzioni, 2008) has become prevalent over traditional relation extraction methods, especially on the Web, because of the intrinsic difficulty in training individual extractors for every single relation. 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 (Fader et al., 2011) and SONEX (Merhav et al., 2012), identify relations by matching patterns over such tags. Dependency parsing gives unambiguous relations among each word in the sentence, and the ORE approaches in this category such as PATTY (Nakashole et al., 2012), OLLIE (Mausam et al., 2012), and TreeKernel (Xu et al., 2013) identify whole subtrees connecting the relation predicate and its arguments. Finally, semantic annotators, such as Lund (Johansson and Nugues, 2008) and SwiRL (Surdeanu et al., 2003), add roles to each node in a parse tree, enabling ORE approaches that identify the precise connection between each argument and the predicate in a relation, independently.

The first contribution of the paper is an objective and fair experimental comparison of the state-of-the-art in ORE, on 5 datasets with varying degree of "difficulty". Of these, 4 datasets 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. A much larger corpus with 12,000 sentences from NYT, automatically annotated is also used. Another experiment focuses on n-ary relation extractions separately. The results show, as expected, that the three broad classes above are separated by orders of magnitude when it comes to throughput. 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.

The second contribution of the paper is a new ORE method, called EXEMPLAR, which applies a key idea in semantic approaches (namely, to identify the precise connection between the argument and the predicate words in a relation) over a dependency parse tree (i.e., without applying SRL). The goal is to achieve the higher accuracy of the semantic approaches at the lower computational cost of the dependency parsing approaches. EXEMPLAR is a rule-based system derived from a careful study of all dependency types identified by the Stanford parser. (Note, however, that other parsers can be used, as shown later on.) EXEMPLAR works for both binary and n-ary relations, and is evaluated separately in each case. For binary relations, EXEMPLAR outperforms all previous methods in terms of accuracy, losing to the shallow methods only in terms of throughput. As for n-ary relations, EXEMPLAR outperforms the methods that support this kind of extraction.

2 Related Work

Others have pointed out the importance of understanding the trade-off between "shallow" versus "deep" NLP in ORE. One side of the argument favors shallow methods, claiming deep NLP costs orders of magnitude more and provide much less dramatic gains in terms of effectiveness (Christensen et al., 2011). The counterpoint, illustrated with a recent analysis on a industrial-scale Web crawl (Dalvi et al., 2012), 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. This paper sheds more light into the debate by comparing the state-of-the-art from three broad classes of approaches.

Shallow ORE. TextRunner (Banko and Etzioni, 2008) and its successor ReVerb (Fader et al., 2011) are based on the idea that most relations are expressed using few syntactic patterns. ReVerb, for example, detects only three types of relations ("verb", "verb+preposition" and "verb+noun+preposition"). Following a similar approach, SONEX (Merhav et al., 2012) extends ReVerb by detecting patterns with appositions and possessives.

ORE via dependency parsing. PATTY (Nakashole et al., 2012) extracts textual patterns from sentences based on paths in the dependency graph. For all pairs of named entities, PATTY finds the shortest path in the dependency graph that connects the two named entities. They limit the search to only paths that start with one of these dependencies: nsubj, rcmod and partmod.

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

Table 1: Binary relation datasets.

OLLIE (Mausam et al., 2012) also extracts relations between two entities. It applies pattern templates over the dependency subtree containing pairs of entities. Pattern templates are learned automatically from a large training set that is bootstrapped from high confidence extractions from ReVerb. OLLIE merges binary relations that differ only in the preposition and second argument to produce n-ary extractions, as in: (A, "met with", B) and (A, "met in", C) leading to (A, "met", [with B, in C]).

The TreeKernel (Xu et al., 2013) method uses a dependency tree kernel to classify whether candidate tree paths are indeed instances of relations. The shortest path between the two entities along with the shortest path between relational words and an entity are used as input to the tree kernel. An expanded set of syntactic patterns based on those from ReVerb are used to generate relation candidates.

ORE via semantic parsing. Recently, a method based on SRL, called SRL-IE, has shown that the effectiveness of ORE methods can be improved with semantic features (Christensen et al., 2011). We implemented our version of SRL-IE by relying on the output of two SRL systems: Lund (Johansson and Nugues, 2008) and SwiRL (Surdeanu et al., 2003). SwiRL is trained on PropBank and expands upon the syntactic features used in previous work. One of its major limitations is that it is only able to label arguments with verb predicates. Lund, on the other hand, is based on dependency parsing and is trained on both PropBank and NomBank, making it able to extract relations with both verb and noun predicates.

3 Experimental Study

This section compares the effectiveness and efficiency of the following ORE methods: ReVerb,

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	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**
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	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	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	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 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 marked in bold, for clarity.

SONEX, OLLIE, PATTY, TreeKernel, SwiRL, Lund and our two variants of our method, EXEMPLAR, explained in detail in Appendix A: (1) EXEMPLAR[S] uses the Stanford parser (Klein and Manning, 2003) and (2) EXEMPLAR[M] uses the Malt parser (Nivre and Nilsson, 2004).

3.1 Binary Relations – Setup

We start by evaluating the extraction of binary relations. Table 1 shows our experimental datasets. WEB-500 is a commonly used dataset, developed for the TextRunner experiments (Banko and Etzioni, 2008). 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 formal, well written new stories from the New York Times Corpus (Sandhaus, 2008). PENN-100 contains sentences from the Penn Treebank recently used in an evaluation of the TreeKernel method (Xu et al., 2013). We manually annotated the relations for WEB-500 and NYT-500 and use the PENN-100 annotations provided by TreeKernel's authors (Xu et al., 2013).

We annotate each sentence manually as follows. We identify exactly two entities and a *trigger* (a single token indicating a relation–see Section A.3) for the relation between them, if one exists. In addition, we specify a *window* of tokens allowed to be in a relation, including modifiers of the trigger and prepositions connecting triggers to their arguments. For each sentence annotated with two entities, a system must extract a string representing the relation between them. Our evaluation method deems an extraction as correct if it contains the trigger and allowed tokens only.

In our annotated sentences, entities are enclosed in triple square brackets, triggers and enclosed in triple curly brackets and the window of allowed tokens is defined by arrows ("--->" and "<---"), as in this example:

```
I've got a media call about
[[[ORG Google]]] --->'s
{{{acquisition}}} of<--- [[[ORG
YouTube]]] --->today<---.
```

where "Google" and "YouTube" are entities of the type organization, "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 relation. OLLIE's extractions 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 extractions.

Our benchmarks are available upon request.

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 system from recognizing only part of an entity. Entities are replaced by "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 systems 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 extract relations between noun phrases (ReVerb, OLLIE, SwiRL and Lund), 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 and Lund, 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.

Metrics. Our evaluation focuses on sentence-level extractions. Therefore, we only apply the steps of each method that perform this task. Additional steps to merge relations and remove infrequent relations are not applied. In addition, we assume there is only one relation between a pair of entities in a sentence. The number of entity pairs with more than one relation 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:

$$P = \frac{\text{\# correct}}{\text{\# extractions}}, R = \frac{\text{\# correct}}{\text{\# relations}}, F = \frac{2PR}{P+R}$$

where "# correct" is the number of extractions deemed as correct.

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.

3.2 Binary Relations – Results

Table 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. The other two datasets, on the other hand, contain randomly chosen sentences. Although WEB-500 is a popular

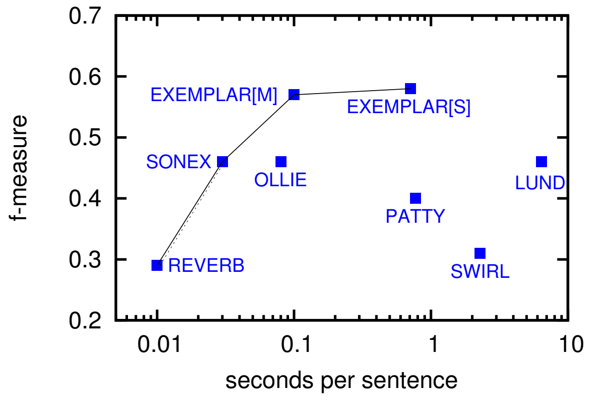

Figure 1: Average f-measure vs average time for NYT-500, WEB-500 and PEN-100.

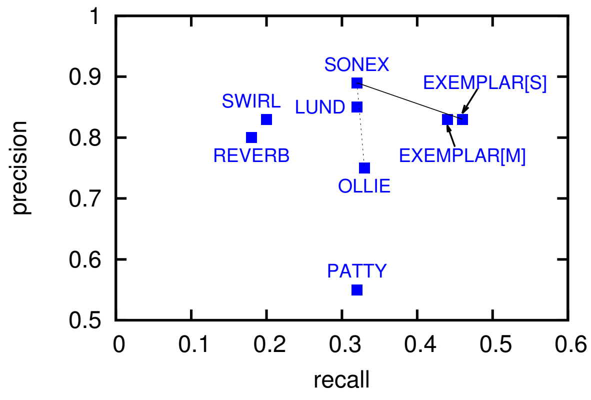

Figure 2: Average precision vs average recall for NYT-500, WEB-500 and PEN-100.

dataset, it perhaps does not represent the challenges 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 (Xu et al., 2013).

Comparing methods based on effectiveness (f-measure) or efficiency (computational cost) alone can be misleading. To do so, 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 1 plots the effectiveness and efficiency of all methods, averaged over all datasets (TreeKernel was not included due to missing results). For efficiency, there is a clear separation of approximately one order of magnitude among methods based on shallow parsing (ReVerb and SONEX), dependency parsing (OLLIE, EXEMPLAR[M], EXEMPLAR[S], and PATTY) and semantic parsing (SwiRL and Lund). 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 are dominated by SONEX as they tie in terms of effectiveness.)

In terms of efficiency, EXEMPLAR[M] and EXEMPLAR[S] closely match OLLIE and PATTY, respectively, since they use the same dependency parsers. EXEMPLAR outperforms both systems by covering a larger number of relational patterns. This is possible because EXEMPLAR looks at each argument 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.

The poor performance of PATTY is due to its rather permissive sentence-level extraction of relations, which looks at the shortest path between arguments. PATTY relies on redundancy of extractions to normalize the produced relations in order to recover from mistakes done in the sentence-level.

Figure 2 illustrates the dominance relation 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 dominates PATTY and Lund, since they tied in recall. OLLIE, however, achieved greater recall than SONEX. Somewhat surprisingly, EXEMPLAR presents 44% more recall than the more sophisticated Lund, at a close level of precision. This can be explained by Lund's dependency on training data, which contains only a subset of all possible predicates and roles. The importance of relations triggered by nouns is illustrated by the higher recall of SONEX and Lund when compared, respectively, to ReVerb and SwiRL, similar methods that handle verb triggers only.

3.3 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 a relation. 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 a relation "rivals" between "P&G" and "Kao Corp." in this sentence. On the other hand, the original annotations for PENN-100 consider only the relation between "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, ReVerb, PATTY, TreeKernel and OLLIE are performed at *the corpus level*. Corpus-level evaluations consider an extracted relation as correct regardless of whether a method was able to identify one or all sentences that describe this relation.

Creating a ground truth for corpus-level evaluations is extremely hard, since one has to identify and curate (e.g., merge near-duplicate relations and co-referential entities) all relations described in a corpus. As a consequence, most corpus-level evaluations perform only a manual inspection of a

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	2.90	0.94	0.30	0.45
Lund	9.20	**0.95**	0.36	0.53

Table 3: Results for n-ary relations.

method's extractions. This manual inspection measures a method's precision, but is unable to measure recall.

Other differences in methodology are as follows. PATTY's evaluation concerns relation patterns (e.g., "wrote hits for") and their type signatures (e.g. Musician–Musician), as opposed to the relation itself, which includes its two arguments. 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.

3.4 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 relations from the New York Times. Every sentence is annotated with a single relation trigger and its arguments.

This experiment measures precision and recall over the extracted arguments. For each sentence, a system can extract a number of relations of the form $r(a_1, a_2, \ldots, a_n)$, where r is the relation name and a_i is an argument. We only use the extracted relation whose name contains the annotated trigger, if it exists. An argument of such relation is deemed a correct extraction if it is annotated in the sentence; otherwise, it is deemed incorrect.

Precision and recall are now defined as follows:

$$P = \frac{\text{\# correct}}{\text{\# extracted args}}, R = \frac{\text{\# correct}}{\text{\# annotated args}}.$$

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	3.58	**0.87**	0.16	0.27
Lund	11.28	0.86	0.21	0.33

Table 4: Results for binary relations automatically annotated using Freebase and WordNet.

where "# correct" is the number of arguments deemed as correct. There are 765 annotated arguments in total. Table 3 reports the results for our experiment with n-ary relations. EXEMPLAR[M] shows a 6% increase in f-measure over Lund, the second best system, while being almost two orders of magnitude faster.

3.5 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[1] and WordNet[2] can be used to automatically annotate a dataset, leading to a useful benchmark.

Our *automatic annotator* identifies pairs of entities and a trigger of the relation between them. It does so by first trying to link all entities to Wikipedia (and consequently to Freebase, since Freebase is linked to Wikipedia) by using the method proposed by (Cucerzan, 2007). Given two entities appearing within 10 tokens of each other in a sentence, our annotator checks whether there is a relation connecting them in Freebase. If such a relation exists, the annotator looks for a trigger in the sentence. A trigger must be a synonym for the Freebase relation (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 with 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.

[1]http://www.freebase.com
[2]http://wordnet.princeton.edu/

Comparing with human annotators. Although we expect our automatic annotator to be less accurate than a human annotator, we are interested to measure 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 4 shows the results for the test sentences. Both EXEMPLAR[S] and EXEMPLAR[M] outperformed all systems in recall, while keeping the same level of precision.

4 Conclusion

This work presents a fair and objective evaluation of several ORE methods, shedding some light on the trade-offs between f-measure and computational as well as precision and recall. Our evaluation is able to assess the effectiveness of different methods by specifying a trigger and a window of allowed tokens for each relation.

EXEMPLAR's promising results indicate that rule-based methods may still be very competitive, especially if rules are applied to each argument separately. Looking at the recall levels of different methods, we conjecture that EXEMPLAR outperforms machine learning methods like Ollie and TreeKernel, because its rules apply in cases not trained by these methods. From a pragmatic point of view, EXEMPLAR is also preferable because it doesn't 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.

The next step for our experimental study is to evaluate corpus-level extractions, where an automatic annotator is essential due to the massive number of annotations required for even one relation, let alone thousands.

Acknowledgements

We thank the anonymous reviewers for useful suggestions to improve the paper, and the Natural Sciences and Engineering Council of Canada, through the NSERC Business Intelligence Network, for financial support.

A EXEMPLAR

ORE methods must recognize relations from the text alone. To do this, each method tries to model how relations are expressed in English. Banko and Etzioni claim that more than 90% of binary relations are expressed through a few syntactic patterns, such as "verb" and "noun+preposition" (Banko and Etzioni, 2008). It is unclear, however, whether n-ary relations follow.

This section presents a study focusing on how n-ary relations ($n > 2$) are expressed in English, based on 100 distinct relations manually annotated from a random sample of 514 sentences in the New York Times Corpus (Sandhaus, 2008).

Table 5 shows six syntactic patterns that cover 86% of our n-ary relations. These patterns are slightly different from those used for binary relations. In binary relations, the pattern implicitly defines the roles of the two arguments. For instance, a relation "met with" indicates that the first argument is the subject and the second one is the object of "with". In order to represent n-ary relations, our patterns do not contain prepositions, possessives or any other word connecting the relation to the argument. For instance, the sentence "Obama met with Putin in Russia" contains the relation "meet" along three arguments: "Obama" (subject), "Putin" (object of preposition with) and "Russia" (object of preposition in).

Relation types. A single relation can be represented in different ways using the patterns shown in Table 5. For instance, the relation "donate" can be expressed as an active verb ("donates"), passive voice ("was donated by") and normalized verb ("donation"). In addition, an apposition+noun relation can be expressed as an copula+noun relation by replacing apposition for the copula verb "be". By merging these patterns, we have that most relations fall into one of the following types: *verb*, *verb+noun*

Pattern	Frequency	Example
Verb	30%	Hingis **beat** Steffi Graf in the Italian Open two weeks ago.
Apposition+noun	19%	Jaden and Willow, the **famous children** of Will Smith, ...
Passive verb	14%	Jumbo the Elephant **was exported** from Sudan to Paris.
Verb+noun	14%	D-League will **move its offices** from Greenville to New York.
Copula+noun	5%	Kimball **was a** Fulbright **scholar** at the University of Heidelberg.
Nominalized verb	4%	Thousands died in Saddam Hussein's **attack** on Halabja in 1988.

Table 5: Patterns representing 86% of the relations with three or more arguments. Frequencies collected from 100 relations from the New York Times Corpus. Relation triggers are highlighted in bold.

Relation Type	Freq.	Example	Variations
Verb	48%	beat	Pass. verb, nom. verb
Copula+noun	24%	is son	Apposition+noun
Verb+noun	14%	sign deal	–

Table 6: Relation types recognized by EXEMPLAR.

and *copula+noun*.

Table 6 present the relation types found through our analysis. We developed EXEMPLAR to specifically recognize these relation types, including their variations.

Argument roles. An *argument role* defines how an argument participates in a relation. In ORE, the roles for each relation are not provided and must also 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 relation 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 "Obama is the president ***of*** the U.S.", "U.S." is an object of the preposition "of" and has the role `of_object`.

Multiple entities can play the same role in a relation instance. For instance, in the sentence "Obama and Putin discuss the Syria conflict", both "Obama" and "Putin" have the `subject` role. Furthermore, some relations 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 systems, 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 (Chambers and Jurafsky, 2011). However, this problem is out of the scope of this work.

`subject:`	"NFL"
`relation:`	"approve new stadium"
`of_object:`	"Falcons"
`in_object:`	"Atlanta"

Figure 3: A relation instance extracted by EXEMPLAR for the sentence "NFL approves Falcons' new stadium in Atlanta".

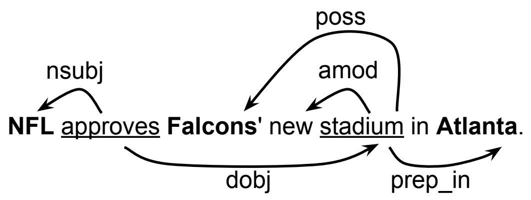

Figure 4: A input sentence after pre-processing. Entities are in bold, triggers are underline and arrows represent dependencies.

A.1 The method

EXEMPLAR takes a stream of textual documents and extracts instances of n-ary relations as illustrated in Figure 3.

A.2 Preprocessing

Given a document, EXEMPLAR extracts its syntactic structure by applying a pipeline of NLP tools provided by the Stanford Parser (Klein and Manning, 2003). Our method converts the original text into sentences, each containing a list of tokens. Each token is tagged with part of speech, lemma and dependencies. EXEMPLAR also works with other dependency parsers based on Stanford's dependencies, such as the Malt parser (Nivre and Nilsson, 2004).

Figure 4 illustrates our running example 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 (Finkel et al., 2005) to recognize named entities. We consider these types of entities: people, organization, location, miscellaneous and date. Figure 4 shows entities highlighted in bold.

A.3 Detecting triggers

After recognizing entities, EXEMPLAR detects *relation triggers*. A trigger is a single token that indicates the presence of a relation. A relation may present one or two triggers. For instance, the relation in our running example has two triggers. EXEMPLAR also uses triggers to determine the relation name, as discussed later.

A trigger can be any noun or verb that was not tagged as being part of an entity mention. Other requirements are enforced for each canonical pattern.

Verb relations. A verb relation is triggered by a verb that does not include a noun as its direct object. The name of the relation is the trigger's lemma.

A noun must be a nominalized verb to be a trigger for verb relations. To identify nominalized verbs, EXEMPLAR checks if a noun is filed under the type "event" in Wordnet's Morphosemantic Database[3]. Doing so may generate false positives; however, EXEMPLAR has a filtering step to eliminate these false positives, as discussed later.

The name of a relation triggered by a nominalized verb is the trigger's original verb (before nominalization). For instance, "donation" triggers the relation "donate".

Copula+noun relations. Only noun triggers are accepted for copula+noun relations. The copula used in the relation name can be a verb with copula dependency to the trigger, or the verb "be" for appositions. The relation name is the concatenation of the copula's lemma and the trigger's lemma along its modifiers. For instance, the relation in the sentence "Jaden and Willow, the famous children of Will Smith" is "be famous child".

Verb+noun relations. EXEMPLAR recognizes two triggers for each verb+noun relation: a verb and a noun acting as its direct object. The relation name is defined by concatenating the verb's lemma with the the noun and its modifiers. In our running example, "approves" and "stadium" trigger the relation "approve new stadium".

A.4 Detecting candidate arguments

After relation triggers are identified, EXEMPLAR proceeds to detect their *candidate arguments*. For this, we look at the dependency between each 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 whether an entity is an argument for the relation or not.

Table 7 enumerates 12 types of dependencies (from a total of 53) that often connect arguments and triggers. EXEMPLAR identifies as a candidate argument every entity that is connected to trigger, as long as their dependency type is listed in Table 7.

Our observations can be seen in our running example. The entities "NFL" and "Atlanta" depends on the trigger "approves" and "Falcons" depend on the trigger "stadium". Since their dependency types are listed in Table 7, these entities are marked as candidate arguments.

A.5 Role Detection

EXEMPLAR determines the role of an argument based on the trigger type (noun or verb), the type of dependency between the trigger and argument and the direction of the dependency. To take into account the dependency direction, we prefix each dependency type with ">" when an entity depends on the trigger and "<" when the trigger depends on the entity.

Table 8 shows EXEMPLAR's rules that assign roles to arguments for each relation type. Rules are triples ($trigger$, $dependency$, $role$) whose meaning is as follows:

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

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

Table 7: Dependencies connecting arguments and triggers. Arguments are in bold and triggers are underlined.

If trigger type = *trigger* and dependency type = *dependency* then assign *role*.

For example, the first rule in Table 8a specifies that an argument must be assigned the role of a *subject* if this argument depends on a *verb* trigger and the dependency type is *>nsubj*.

Rules are ordered by descending priority in Table 8. In case several rules can be applied for an argument, we apply only the rule with higher priority. If none of the rules applies to an argument, EXEMPLAR removes this argument from the relation.

Exceptions. There are three exceptions for the rules above. The first exception concerns arguments of verb relations whose dependency type is <partmod or <rcmod. 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 verbs followed by the preposition "by", such as in "Georgian invasion by Russia". Arguments of this type of trigger with dependency types >nn, >amod or >poss are assigned the role direct_object.

Finallly, there is an exception for copula+noun relations expressed with close appositions of the form: "determiner entity noun". An example is "the Greenwood Heights section of Brooklyn". Here, EXEMPLAR assigns the subject role to the entity between the determiner and the noun.

Trigger Type	Dependency Type	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 relations.

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

(b) Rules for copula+noun relations.

Trigger Type	Dependency Type	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

(c) Rules for verb+noun relations.

Table 8: Rules for assigning roles to arguments.

A.6 Filtering Relations

The final step in EXEMPLAR is to remove incomplete relations. EXEMPLAR removes relations with less than two arguments and relations that do not present subject and direct_object.

For EXEMPLAR, Lund and SwiRL, which extract n-ary relations, our evaluation needs to convert n-ary relations into binary ones. This is done by selecting all pairs of arguments from a n-ary relation and creating a new (binary) relation for each of them. Binary relations containing two prepositional objects (or equivalent for SRL systems) are removed.

References

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*, pages 28–36, Columbus, Ohio, June. Association for Computational Linguistics.

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, pages 976–986, Stroudsburg, PA, USA. Association for Computational Linguistics.

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*, pages 113–120. Association for Computational Linguistics.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL'12, pages 708–716.

Nilesh Dalvi, Ashwin Machanavajjhala, and Bo Pang. 2012. An analysis of structured data on the web. *Proc. VLDB Endow.*, 5(7):680–691.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying Relations for Open Information Extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, EMNLP'12.

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, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of propbank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP'08, pages 69–78, Stroudsburg, PA, USA. Association for Computational Linguistics.

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, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL'12.

Yuval Merhav, Filipe de Sá Mesquita, Denilson Barbosa, Wai Gen Yee, and Ophir Frieder. 2012. Extracting information networks from the blogosphere. *TWEB*, 6(3):11.

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, pages 1135–1145, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joakim Nivre and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of Computational Natural Language Learning*, CoNLL'04.

Evan Sandhaus. 2008. The new york times annotated corpus. `http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2008T19`.

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*, pages 8–15. Association for Computational Linguistics.

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*, pages 868–877, Atlanta, Georgia, June. Association for Computational Linguistics.