A Kernel of Truth

Keenon Werling[*]

Stanford University
keenon@stanford.edu

Abstract

This paper first proposes a (to the author's limited knowledge) novel extension to the OpenIE paradigm, to allow the expression of recursive relations, and presents a fully implemented extension to the EXEMPLAR system, uncreatively referred to as Recursive-EXEMPLAR, to extract recursive n-ary relations automatically and with very high precision. It then explores the possibility of a "concept kernel" over the output of Recursive-EXEMPLAR to allow arbitrary learning over the ideas expressed in raw text inputs, and discusses computational considerations.

I. Introduction

Open Information Extraction (Banko and Etzioni, 2008) is an attractive paradigm, because it frees us from the burden of manually describing every relation type we want to extract with large quantities of labelled training data. At the same time, it is notoriously difficult to use as input to higher level ML tasks, because current state-of-the-art systems ignore many important limitations on the quality of their extractions.
Primarily, this is because related extractions are not linked. For example, "the prime minister proposed that women ascend the throne" would be extracted traditionally as two seperate relations: *proposed*(**prime minister**), *ascend*(**women**, **throne**). This repsentation suggests that "women ascend the throne" is already a fact, and doesn't give us any information on what the prime minister proposed. We could instead extract the recursive relation *proposed*(**prime minister**, *ascend*(**women**, **throne**)). This recursive representation has the benefit that limitations on truth that go beyond the complexity of basic prepositional relations are easily represented, and objects that are more complex than single phrases can still be linked without loss of information. The relative reliability of a speaker can be factored into the quality of an extraction if the speaker is known, and temporal and spatial limitations are still easily extracted from prepositional attachments at any point in the recursive relation. A method will be proposed in the second half ot the paper theoretically capable of learning these relationships automatically, though computational limits prevent a full experiment.

II. Related Work

OpenIE has seen an explosion of methods in the last few years, with varying degrees of accuracy and computational cost. Briefly summarized, they fall into 3 classes: ReVerb (Fader et al. 2011) and SONEX (Merhav et al. 2012), using only shallow syntactic information like POS tags for their extractions. The syntactic parse OpenIE systems, PATTY (Nakashole et al., 2012), OLLIE (Mausam et al. 2012) use regular expressions over parse trees. TreeKernel (Xu et al., 2013) uses an SVM with a kernel over sub-tree-similarity to classify whether or not a relation is present between two named entities. Extremely computationally expensive methods using a semantic parse as input also exist, and achieve competitive results, but are prohibitively time consuming.

[*]Many thanks to Gabor Angeli, and the Stanford NLP Lab in general, for data and advice

III. Failed Methods

Prior to considering a rule-based approach, several methods for the automated training of a recursive OpenIE systems were attempted. Hill-climbing over the regex rules used to extract fully formed relations was implemented, an improvement on the methods in OLLIE, but could not get above 30% precision *on the available dataset*. Application of an SVM to the role labeling step in *Recursive-EXEMPLAR*, in a similar way to the work in TreeKernel (Xu et al., 2013), using the efficient tree-kernels presented in (Moschitti, 2006) found that altough it could get some things right on test data, its errors were easily corrected by the application of a rule based system. The working hypothesis is that inability to get competitive results with machine learning systems is due to the inability to *grammatically consistently* hand-label enough data to train the systems. At the beginning of the project, 100 sentences were carefully hand-labeled, which took 5 hours, and it was discovered that a baseline system that used every pattern it found in the hand-labeled data to make new extractions achieved a 30% precision, 10% recall on unseen data. The underlying structure just wasn't dense enough to be learned from such a small amount of data, so (for now) rule based systems rule the waves.

IV. Original *EXEMPLAR*

A recent paper (Mequita et. al, 2013) did a fair comparison of several leading methods, and proposed an elegant and relatively computationally efficient method for OpenIE with the highest accuracy of any of the measured systems, *EXEMPLAR*. The two major innovations represented in the design of *EXEMPLAR* are

1. The ability to extract relation phrases independently of their arguments, like a semantic-parse-based system.

2. The return to a rule-based approach as a solution to a general lack of sufficient labeled data.

The system presented in this paper builds directly upon *EXEMPLAR*'s innovations to achieve a system for extracting n-ary, recursive relations with state-of-the-art accuracy.

V. Design of *Recursive-EXEMPLAR*

The *Recursive-EXEMPLAR* extraction system works in a series of deterministic steps, repeatedly applied until no new information can be gained from data. Then post-processing is applied across the extracted relations, and the results are returned. The general algorithm is as follows:

```
**function** ExtractRelations(S)
    $P \leftarrow \text{StanfordParser}(S)$
    $E \leftarrow \text{NamedEntities}(P)$
    $R \leftarrow []$
    **while** true **do**
        $T \leftarrow \text{DetectTriggers}(P, E, R)$
        $R_{new} \leftarrow \text{DetectRoles}(P, E, R, T)$
        $R_{new} \leftarrow \text{FilterRels}(P, R_{new})$
        **if** $|R_{new}| == 0$ **then**
            **return** $\text{PostProcess}(R)$
        **else**
            $R \leftarrow \text{Merge}(R, R_{new})$
        **end if**
    **end while**
**end function**
```

Named entities are detected using the Stanford NER system. DetectTriggers(), DetectRoles(), FilterRels(), and PostProcess() will be explained in subsequent sections. The major change in the design of *Recursive-EXEMPLAR* over *EXEMPLAR* is the use of a loop to detect new relations given old relations, instead of a single pass to detect stand-alone relations all at once, and a final PostProcess() step to use parse data to make relations as easy to use as possible for other applications. The necessity for the loop design will become clear in the discussion of DetectRoles().

VI. Detecting Triggers

The brilliant leap in the design of the original *EXEMPLAR* was to detect "triggers", defined to be words that indicate the presence of a relation, seperately from their arguments. *EXEMPLAR* identifies 3 kinds of triggers, **Verb**, **Copula+Noun**, and **Verb+Noun**. *Recursive-EXEMPLAR* adds the recursive trigger, **Conjunction**. Their relative frequencies, automatically collected from 3247 random sentences from Wikipedia, and 55173 sentences from NYT are as follows:

Table 1: *Wikipedia Trigger Frequencies*

Trigger Type	Freq.	Avg./Sentence
Verb	53.8%	1.32
Verb+noun	30.2%	0.74
Copula+noun	11.0%	0.27
Conjunction	5.0%	0.12

Table 2: *NYT Trigger Frequencies*

Trigger Type	Freq.	Avg./Sentence
Verb	53.2%	1.37
Verb+noun	32.3%	0.83
Copula+noun	12.9%	0.32
Conjunction	5.0%	0.13

The striking distributional similarity between the two sources suggests that the types of extractions made by *Recursive-EXEMPLAR* across varied English sources will be balanced between different tree structures.

VII. Detecting Roles

Once a list of triggers is collected, the next step assigns arguments to the triggers, which become preliminary n-ary relations. Each argument is given a weight of preference, and only the single highest preferenced subject and direct object relations are allowed. The classification process is entirely rule-based. Rules take the form of simple dependency patterns from a trigger, with restrictions about what trigger type is applicable (Verb, Verb+Noun, etc), and what trigger POS is applicable.

Recursive-EXEMPLAR adds one more feature to rules. Rules may be marked "recursive" or "non-recursive". A rule marked recursive is only applicable to words that are already within an argument or trigger for an n-ary relation extracted in a previous round, and then the entire n-ary relation is taken as the argument. Rules marked non-recursive can only be applied to words that are not yet contained in any argument or trigger.

VIII. Post-processing

For the convenience of users of the output of *Recursive-EXEMPLAR*, a few post-processing steps are done once the algorithm has returned.

1. An attempt is made to collapse "that" relations. If the subject of the "that" relation has no direct object, then the direct object of the "that" relation is made in the direct object of the subject of the "that", and the original "that" is deleted from the output. To clarify that horrible sentence, using our running example, "the prime minister proposed that women ascend the throne" extracts *that*(*proposed*(**Prime Minister**),*ascend*(**women**,**throne**)), which is reduced to *proposed*(**Prime Minisert**,*ascend*(**women**,**throne**)).

2. Coreference is applied to all arguments that contain pronouns, because extractions containing "he" and "she" are totally useless for some higher level applications. Coreference is *not* applied in general, because it tends to reduce the accuracy of the extractions, since any coreference system is imperfect.

IX. Results

Results are measured against Wikipedia articles and NYT seperately. For each corpus I hand labeled 100 flat extractions, and 100 recursive extractions as correct or incorrect. I measured these two groups seperately because

I am primarily interested in the recursive extractions, and since they are far less frequent, measuring together would mean that I would have a label many more flat extractions in order to get a reasonable sample of recursive extractions. The relative frequency of recursive extractions, as a percentage of total extractions, is noted in the first numeric column.

Table 3: *Hand Labeled Precision Scores*

Corpus	Rec. Freq.	Flat Prec.	Rec. Prec.
Wikipedia	10.9%	71.3%	74.1%
NYT	12.7%	73.1%	81.2%

These numbers are limited to precision only (no recall values) because it proved impossible to label *consistent* n-ary extractions from a body of sentences in the time provided for this project. A reasonable lower bound on recall is possible, however, since the original *EXEMPLAR* paper measured recall at 30%, and the additional rules can only have increased recall (while potentially lowering precision). Empirically, precision was not lowered, but increased. Having measured precision on both NYT and Wikipedia, the recursive extraction rules presented here are highly precise.

Below are some examples of the recursive extractions produced by *Recursive-EXEMPLAR* on the test set during the labeling run to generate the precision numbers you see above:

1. Attorneys for the plaintiffs contended that Exxon bore responsibility for the accident because the company "put a drunk in charge of a tanker in Prince William Sound."

   because(
       bore(**Exxon**, **responsibility**),
       put(**company**, **drunk**)
           [*IN_OBJECT* **charge**]
)
   contended(
       Attorneys,
       bore(**Exxon**, **responsibility**)
)

2. The EDSAC's memory consisted of 1024 locations, though only 512 locations were initially implemented.

   though(
       consisted(**EDSAC memory**)
           [*OF_OBJECT* **1024 locations**],
       implemented(**PASSIVE**,
           only 512 locations)
)

3. Popular opinion holds that longer scale length contributes to greater amplitude.

   holds(
       Popular opinion,
       contributes(**longer scale length**)
           [*TO_OBJECT* **greater amplitude**]
)

4. The economy of the city of Cordova, Alaska was adversely affected after the spill damaged stocks of salmon and herring in the area.

   after(
       affected(**PASSIVE**, **economy**),
       damaged(**spill**, **stocks**)
           [*IN_OBJECT* **area**]
)

X. The Promise of Concept Kernels

What good can recursive first order logic do for machine learning? One possibility is the opening up of knowledge locked away in text as input to learning algorithms. As a concrete example, imagine a system that reads the news for structured extractions, and then applies a non-linear classifier over its extractions to find a relationship between ideas in the news and stock prices. This is deeper than just a word counter, since algorithms can be run over structured representations of ideas, instead of bags of tokens. This is theoretically possible with *Recursive-EXEMPLAR*, though it is very computationally intense. The recursive n-ary relation extractions my system produces can be seen as trees, with labeled nodes. Many different "kernels" (not Mercer kernels, really just similarity measures) exist for measuring the similarity between recursive structures without labels (Moschitti, 2006), and with slight modifications they can take into account semantic similarity information between nodes.

The Major Unsolved Problem:

The simple concept kernels this paper explored used 50 dimensional word representations to measure semantic similarity. In practice, this led to a very large represenational capacity, because a small extraction with 3 nodes has roughly the same representational capacity as an 150 dimensional real numbered vector. Thus, an SVM required a lot of data to collect enough support vectors to make reasonable estimates. On the other hand, an SVM is very computationally expensive to use, and even more so because the kernel computations are quadratic in the size of the concept trees, with large constant factors. An SVM was run using a concept kernel on a dataset of extractions from a month of NYT articles, and compared against fluctuations in 3-year T-bills, with a binary up-down classification over the next time period.

Table 4: *Computation v. Accuracy*

# extractions	Prec.	Hours to converge
90,000	0.45	2
450,000	0.47	14

Extractions improved with sample size, but computational limits prevented increasing data size. More data is available. *Recursive-EXEMPLAR* was able to mine 40 million extractions from a 2 year newswire corpus. Future research will explore a large scale run, and find if concept kernels work in practice to allow machines to learn patterns in arbitrary human ideas.

References

[Dan Klein and Chistopher D. Manning, 2003] Accurate unlexicalized parsing. *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.

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

[Anthony Fader et al., 2011] Identifying Relations for Open Information Extraction. *Proceedings of Coreference on Empirical Methods in Natural Language Processing*, EMNLP-CoNLL'12.

[Mausam et al., 2012] Open language learning for information extraction. *Proceedings of Coreference on Empirical Methods in Natural Language Processing and Computaitonal Natural Language Learning*, EMNLP-CoNLL'12.

[Moschitti, 2006] Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees *ECML 2006*, ECML 2006, pages 318-329

[Ndapandula Nakshole et al., 2012] Patty: a taxonomy of relational patterns with semantic types. *Proceedings of Coreference on Empirical Methods in Natural Language Processing and Computaitonal Natural Language Learning*, EMNLP-CoNLL'12, pages 1135-1145, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Ying Xu et al., 2013] Open information extraction with tree kernels. *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.