An Analysis of Open Information Extraction based on Semantic Role Labeling

Janara Christensen , **Mausam** , **Stephen Soderland** and **Oren Etzioni**
Turing Center
University of Washington
Seattle, WA 98195, USA
{janara,mausam,soderlan,etzioni}@cs.washington.edu

ABSTRACT

Open Information Extraction extracts relations from text without requiring a pre-specified domain or vocabulary. While existing techniques have used only shallow syntactic features, we investigate the use of semantic role labeling techniques for the task of Open IE. Semantic role labeling (SRL) and Open IE, although developed mostly in isolation, are quite related. We compare SRL-based open extractors, which perform computationally expensive, deep syntactic analysis, with TEXTRUNNER, an open extractor, which uses shallow syntactic analysis but is able to analyze many more sentences in a fixed amount of time and thus exploit corpus-level statistics.

Our evaluation answers questions regarding these systems, including, can SRL extractors, which are trained on PropBank, cope with heterogeneous text found on the Web? Which extractor attains better precision, recall, f-measure, or running time? How does extractor performance vary for binary, n-ary and nested relations? How much do we gain by running multiple extractors? How do we select the optimal extractor given amount of data, available time, types of extractions desired?

1. INTRODUCTION

The challenge of Machine Reading and Knowledge Extraction at Web scale [10] requires a scalable system for extracting diverse information from large, heterogeneous corpora. The traditional approaches to information extraction, *e.g.*, [17, 1], seek to learn individual extractors for each relation of interest and hence, cannot scale to the millions of relations found on the Web. In response, the *Open Information Extraction* paradigm [5] attempts to overcome this knowledge acquisition bottleneck by extracting relational tuples without any restrictions of a pre-specified vocabulary, domain or ontology.

TEXTRUNNER[6], a state-of-the-art open extractor, is a relation classifier based primarily on shallow syntactic features. In this paper, we study the applicability of semantic role labeling (SRL) for the task of Open IE.

Our first observation is that SRL and Open IE, although developed in isolation, are related tasks: semantically labeled arguments correspond to the arguments in Open IE extractions, and verbs often match up with Open IE relations. We construct a scheme for Open IE based on SRL, and create novel extractors based on two state-of-the-art SRL systems, one developed at UIUC and the other at LUND [15, 12]. These systems represent the top ranking systems at CoNLL 2005 and 2008, respectively. We study the trade-offs between TEXTRUNNER and SRL-based extractors across a broad range of metrics and experimental conditions, both qualitative and quantitative.

Given the distinct perspectives from which Open IE and SRL have been developed, we expect TEXTRUNNER and SRL extractors to be quite different. For example, we expect the SRL extractors to have lower recalls on Web text due to out-of-vocabulary verbs and diverse writing styles, since they are trained on a more homogeneous corpus (PropBank). On the other hand, their deep processing in comparison to TEXTRUNNER's shallow syntactic features may result in a much higher precision. We also believe *a priori* that TEXTRUNNER will be faster, but cannot quantify the difference, as no previous work has studied this. This paper reports that, contrary to our beliefs, SRL is robust to noisy Web text, and achieves a much larger recall; whereas TEXTRUNNER obtains a much higher precision than SRL extractors, at lower recalls. Finally, TEXTRUNNER is 20-700 times faster than SRL systems we have tested (dependent on use of dependency parser versus constituency).

While previous studies assume a small data set and ample time available for processing, to our knowledge, we are the first to study the alternative experimental conditions in which the data set is large and the processing

time is limited. This is especially important for most universities and companies, which do not have access to Microsoft or Google sized cluster of machines. Even cloud computing resources are expensive and hence limited. In this paper, we examine both of these conditions, SmallCorpus and LargeCorpus respectively. For LargeCorpus, we devise a scheme to make use of the available time carefully. We first run TextRunner, which is enormously faster than the SRL-extractors we tested. For the remaining available time, we run other extractors over an intelligently chosen subset of the corpus. This hybrid scheme obtains the best value for time compared to the individual extractors.

We first describe the Open IE paradigm in more detail and then explain how SRL systems can be used to produce Open IE output. Next, we discuss qualitative differences between the output produced by traditional Open IE systems and the output produced by SRL-based systems. Finally, we describe two experiments under different conditions and end with related work and conclusions.

2. BACKGROUND

Open Information Extraction [5] is a paradigm where the system makes a single (or constant number of) pass(es) over its corpus and extracts a large set of relational tuples without requiring any relation-specific training data. These tuples attempt to capture the salient relationships expressed in each sentence. For instance, given the sentence, *"McCain fought hard against Obama, but finally lost the election,"* an Open IE system would extract two tuples <McCain, fought against, Obama>, and <McCain, lost, the election>. These tuples can be binary or n-ary, where the relationship is expressed between more than two entities, or nested (*e.g.*, <Microsoft, announced,<they, acquired, Komoku>>).

TextRunner is a state-of-the-art Open IE system that performs extraction in two key steps. (1) A self-supervised learner outputs a CRF-based classifier that uses unlexicalized features (it models closed class words but not function words) for extracting relationships. The self-supervised nature alleviates the need for hand-labeled training data and unlexicalized features help scale to the multitudes of relations found on the Web. (2) A single pass extractor, which uses shallow syntactic techniques like POS tagging and NP chunking, applies the CRF classifier to extract an unbounded number of relationships expressed in text. The use of shallow features makes TextRunner highly efficient.

The extractions from TextRunner are ranked using redundancy, an assessor that assigns higher confidence to tuples occurring multiple times based on a probabilistic model [9].

Semantic Role Labeling consists of detecting semantic arguments associated with a verb in a sentence and their roles (such as Agent, Patient, Instrument, *etc.*). Given the sentence *"The pearls I left to my son are fake"* an SRL system would conclude that for the verb 'leave', 'I' is the agent, 'pearls' is the patient and 'son' is the benefactor. Because not all roles feature in each verb, the roles are commonly divided into meta-roles (A0-A7) and additional common classes such as location, time, *etc.* Each Ai can represent a different role based on the verb, though A0 and A1 most often refer to agents and patients respectively.

Tuples	Binary, N-ary, Nested
Metrics	Precision, Recall, F-measure
Settings	SmallCorpus, LargeCorpus
Systems	TextRunner, SRL-IE-UIUC, SRL-IE-Lund

Table 1: A table outlining the various experimental conditions in this paper.

Availability of lexical resources such as PropBank [13] and FrameNet [3], both of which annotate text with roles for each argument, has enabled significant progress in SRL systems over the last few years [18, 12, 8, 14]. We use UIUC-SRL [15] and Lund-SRL [12] as our base SRL systems. We choose these systems, as they represent the state-of-the-art for systems based on constituency and dependency parsing – they are winners of the CoNLL shared tasks 2005 and 2008 respectively.

Both these SRL systems apply a pipeline of parsing, argument identification, and classification trained over PropBank. UIUC-SRL operates in four key steps: pruning, argument identification, argument classification and inference. Pruning involves using a full parse tree and heuristic rules to eliminate constituents that are unlikely to be arguments. Argument identification uses a classifier to identify constituents that are potential arguments. In argument classification, a classifier assigns role labels to the candidates identified in the previous stage. Argument information is incorporated across arguments in the inference stage, which uses an integer linear program to make global role predictions.

Lund-SRL has a similar process. It first applies a dependency parser and then uses a pipeline of classifiers to identify predicates and identify and classify arguments. Next it applies a set of linguistic constraints and uses a predicate-argument reranker to rank the candidates. Lastly, it uses a syntactic-semantic reranker to score the joint syntactic-semantic models.

3. SRL-BASED OPEN IE

Our first observation is that verbs and their semantically labeled arguments almost always correspond to Open IE relations and arguments respectively. SRL computes more information than Open IE requires. Therefore, for the purpose of a comparison with Open IE systems, we convert SRL output into extractions. We

illustrate this conversion process via an example.

For example, given the sentence, *"Eli Whitney created the cotton gin in 1793,"* TEXTRUNNER extracts two tuples, one binary and one n-ary:

arg0	Eli Whitney
rel	created
arg1	the cotton gin

binary tuple

arg0	Eli Whitney
rel	created (arg1) in
arg1	the cotton gin
arg2	1793

n-ary tuple

The SRL systems label constituents of a sentence with the role they play in regards to the verb in the sentence. An SRL system will identify the following semantic roles for the verb 'create':

A0	Eli Whitney
verb	created
A1	the cotton gin
temporal	in 1793

It is easy to see that the two formats are very related. For fair comparisons we convert SRL output to *equivalent* number of Open IE tuples. Our method first assigns the verb along with its modifiers, following preposition, and negation, if present, to be the relation. It then assigns all constituents labeled Ai for that verb, as well as any that are marked *Direction*, *Location*, or *Temporal* to be the arguments of the relation. We order the arguments in the same order as they are in the sentence and with regard to the relation (except for direction, location and temporal, which cannot be arg0 of an Open IE extraction and are placed at the end of argument list). As we are interested in relationships between entities, we consider only the verbs that have at least two arguments.

The generation of nested relations happens similarly. The key difference is in identifying whether a semantic tuple is a nested extraction. SRL-IE identifies such cases by noticing that an argument to one verb is long and contains a full semantic tuple with a different verb. This is easy to operationalize since an SRL system always reports all the semantic tuples found in the sentence.

In our experiments, we ignore part of the semantic information (such as distinctions between various Ai's) that UIUC-SRL and LUND-SRL provide. An IE system built using SRL may retain this information, if the downstream process (such as question answering engine) can make use of this information. Notice that, in our conversion, an SRL extraction that was correct in the original format is never changed to an incorrect Open IE extraction. However, an incorrectly labeled SRL extraction could convert to a correct Open IE extraction, if the arguments were correctly identified but assigned incorrect semantic roles.

4. QUALITATIVE COMPARISON OF EXTRACTORS

Because SRL and Open IE are developed from different perspectives, we first study their differences qualitatively.

Argument boundaries: The SRL systems are lenient in deciding what constitutes an argument and tend to err on the side of including too much rather than too little; TEXTRUNNER is more conservative, sometimes to the extent of omitting crucial information, particularly post-modifying clauses and PPs. For example, TEXTRUNNER extracts <Bunsen, invented, a device> from the sentence *"Bunsen invented a device called the Spectroscope"*. SRL extractors include the entire phrase "a device called the Spectroscope" as the second argument. Generally, the longer arguments in SRL-IE-UIUC and SRL-IE-LUND are more informative, but TEXTRUNNER's succinct arguments normalize better leading to an effective use of redundancy in ranking.

Out-of-vocabulary verbs: While we expected TEXTRUNNER to handle unknown verbs with little difficulty due to its unlexicalized nature, the SRL-based systems could have had severe trouble leading to a limited applicability in the context of Web text. However, contrary to our expectations, both SRL systems gracefully handle new verbs (*i.e.*, verbs not in their PropBank training) by only attempting to identify A0 (the agent) and A1 (the patient). In practice, this is very effective – both SRL extractors recognize the verb and its two arguments correctly in *"Larry Page googled his name and launched a revolution."* In practice, out-of-vocabulary verbs are rare and appeared in only 5% of our data.

Part-of-speech ambiguity: All systems have difficulty in cases where the part of speech of a word is ambiguous or difficult to tag automatically. For example, the word 'write' when used as a noun causes trouble for both systems. In the sentence, *"Be sure the file has write permission."*, all three extractors extract <the file, write, permission>. Part-of-speech ambiguity affected about 20% of sentences.

Complex sentences: Because TEXTRUNNER relies on shallow syntactic features, it performs more poorly on complex sentences. SRL-based systems, due to their deeper processing, can better handle complex syntax and long-range dependencies.

N-ary and nested relations All extractors suffer significant quality loss in complex extractions compared to binary. For example, given the sentence *"Google announced it will acquire YouTube,"* TEXTRUNNER mistakenly extracts <Google, announced, it>. N-ary and nested relations were present in 40% and 35% of our data respectively.

5. EXPERIMENTAL RESULTS

In our quantitative evaluation we examine the strengths and weaknesses of these extractors under two experimental conditions: (1) SmallCorpus (Section 5.1), in which we have a small corpus and ample computation time available, and (2) LargeCorpus (Section 5.2), in which the corpus is large and all systems cannot complete the processing. We evaluate the quality of binary, n-ary, and nested extractions in all these settings (see Table 1). In all experiments, we threshold TextRunner's CRF-confidence at 0.4, which maximizes F-measure on a development set.

Data sets: Because of the SRL-based systems' relatively slow processing time, we required our test sets be of manageable size. Moreover, Open IE on the Web has benefited from redundancy, and so the data sets needed to mimic the redundancy found on the Web. We created two test sets, one for binary and n-ary, and the other for nested extractions. The first set focused on five target relations – *invent, graduate, study, write,* and *develop*, and the second used two relations with common nested extractions – *say* and *announce*. The first set of relations is similar to that used in [5]. To our knowledge, we are the first to investigate nested extractions, hence our dataset to study that is unique.

A key challenge was dataset construction that was of manageable size but still mimicked the redundancy found on the Web. We could not use prior datasets, since either they were not Web text, or they were too large. To obtain redundant data commonly found on the Web, we first queried a corpus of 500M Web documents for a sample of sentences with these verbs (or their inflected forms, *e.g.*, invents). We chose 200 typical agents per verb (*e.g.*, Edison (for invent), Microsoft (for announce)) and searched for sentences with both the verb as well as these agents. This resulted in a test set of 29,842 sentences for binary and n-ary relations, and a second set of 16,777 sentences for nested relations. These test sets have the desired properties and enable us to study the performance of different extractors on Web text.

To compute precision on these test sets, we tagged a random sample of over 4,800 extractions. A tuple is correct if the arguments have correct boundaries and the relation accurately expresses the relationship between all of the arguments, even if the relation is uninformative. For example, for the sentence *"Bunsen invented a device called the Spectroscope"*, both second arguments, 'a device' and 'a device called the Spectroscope' would be marked as correct.

Determining the absolute recall is precluded by the amount of hand labeling necessary. Instead, we compute pseudo-recall by taking the union of correct tuples from all methods as denominator.[1]

5.1 SmallCorpus Setting

Table 2 reports the performance of the three extractors on our data sets for this traditional NLP setting. Overall, SRL-IE-Lund achieves the highest precision, and SRL-IE-UIUC achieves the highest recall and the highest F1 score. TextRunner's performance on nested extractions is especially poor, but that is expected, since it only extracts relations between noun phrases and nested extractions, by definition, have a full extraction as an argument. On the other hand, both SRL-based systems run *far slower* than TextRunner. TextRunner on average processes a sentence in under 0.02 secs whereas SRL-IE-Lund and SRL-IE-UIUC are over 20x and 500x slower respectively. SRL-IE-UIUC ran much slower because of its use of constituency parsing (which is slower compared to dependency parsing) and its use of an integer linear program for global inference.

By taking a union of the SRL-based systems' output and the *highest precision subset* of TextRunner's extractions, we achieve the highest recall and F-measure (Table 2). We identify the highest precision subset of TextRunner's extractions by our novel *locality* ranking (see Figure 2).[2] This shows the benefit of using multiple systems for extraction – they extract different tuples. We call this the *smart union* of all systems, and this is the method of choice for the SmallCorpus setting.

Although SRL-IE-Lund has higher overall precision, there are some conditions under which the shallow processing of TextRunner can obtain superior precision! We analyze the performance of these systems under two different rankings – redundancy, which has been examined before for TextRunner[4], and locality, a novel measure.

Redundancy: Redundancy is the number of times a relation has been extracted from unique sentences. Intuitively, we have more confidence in a relation that is extracted many times than a relation that is extracted only a small number of times. We compute redundancy over normalized extractions, ignoring noun modifiers, adverbs, and verb inflection. Figure 1(a) displays the results for binary extractions, ranked by redundancy. We use a log scale on the x-axis, since high redundancy extractions account for less than 1% of the recall. For binary extractions, redundancy improved TextRunner's precision significantly, but at a *dramatic* loss in recall – it achieved 0.82 precision at 0.009 recall. For a highly redundant corpus, TextRunner would be the

[1] Tuples from two systems are considered equivalent if for the relation and each argument, the extracted phrases are equal or if one phrase is contained within the phrase of the other.

[2] discussed in more detail next.

	TextRunner			SRL-IE-Lund			SRL-IE-UIUC			Smart Union		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Binary	.55	.26	.35	**.70**	.51	.59	.63	.75	.68	.67	**.95**	**.77**
N-ary	.42	27	.32	**.61**	.27	.37	.53	.57	.55	.56	**.78**	**.66**
Cpu Time: Binary, N-ary	6 minutes			155 minutes			3126 minutes			3287 minutes		
Nested	.09	.02	.03	**.63**	.44	.59	.52	.84	.64	.57	**1.0**	**.72**
Cpu Time: Nested	3 minutes			60 minutes			2016 minutes			2078 minutes		

Table 2: **In SmallCorpus, SRL-IE-Lund has the highest precision. Taking the union of the SRL systems and the higher precision results from TextRunner achieves the highest recall and F-measure. Both SRL-based systems require over an order of magnitude more processing time. The bold values indicate the highest values for the metric and relation-type.**

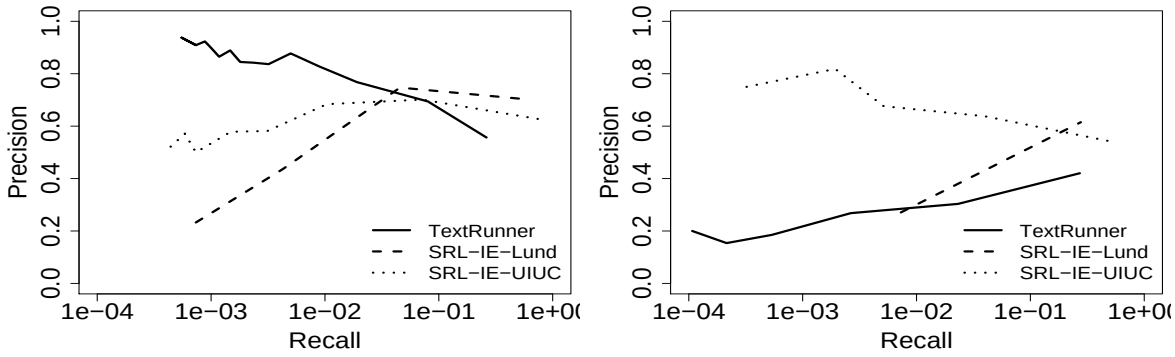

Figure 1: **SmallCorpus: redundancy ranking for binary and n-ary relations. (Note the log scale) (a) TextRunner has highest precision at highest redundancy, but at a very low recall (0.01). The arguments in SRL-based extractors do not normalize well; the high redundancy region has a large fraction of systematic extraction errors. (b) For n-ary extractions, redundancy is not very effective.**

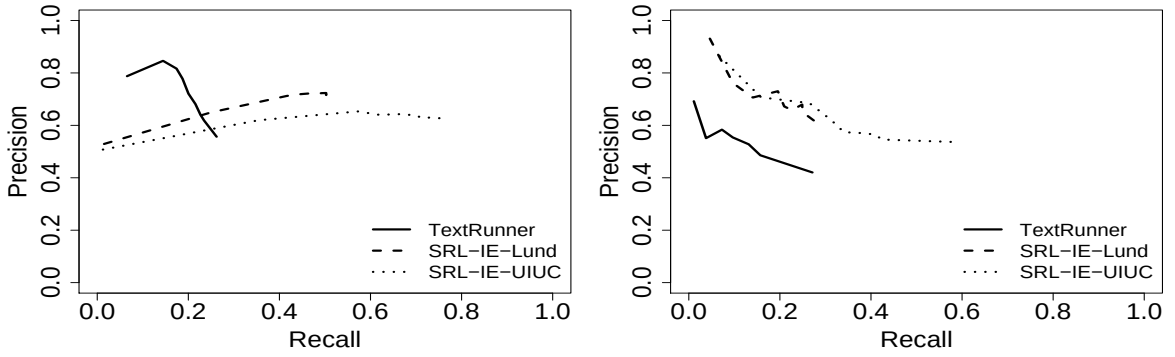

Figure 2: **SmallCorpus: locality ranking for binary, and n-ary relations. (a) Locality ranking gives a large boost to TextRunner's precision for binary relations, and at a recall of 0.2, much higher than that achieved by redundancy ranking. (b) For n-ary extractions, locality helps all systems.**

algorithm of choice, however, this experiment clearly shows that highly redundant extractions are usually very limited, even in Web-style text.

For n-ary and nested relations, and binary relations for SRL-based extractors (Figure 1(a,b)), redundancy actually hurts precision (nested relation graphs omitted). These extractions tend to be so specific that genuine redundancy is rare, and the highest frequency extractions are often systematic errors. *E.g.*, the most frequent SRL-IE-UIUC extraction was <nothing, write, home>, from sentences with the phrase *"nothing to write home about"*.

Locality: Our experiments with TEXTRUNNER led us to discover a new validation scheme for the extractions – *locality*. We define locality as the number of tokens in between the first and the last arguments in the sentence. We observed that TEXTRUNNER's shallow features can identify relations more reliably when the arguments are closer to each other in the sentence. Figure 2 reports the results from ranking extractions by locality.

We find a clear correlation between locality and precision of TEXTRUNNER, with precision 0.81 at recall 0.17, where the locality is 3 tokens or less for binary extractions. This result is very surprising because SRL systems perform deep syntactic and additional semantic analysis, still TEXTRUNNER's shallow syntactic processing with simple locality assessment is able to obtain significantly higher precision (though at reasonable, but lower recall of 0.17). Comparing this result with precision 0.82 at recall 0.01 for redundancy-based assessing, we find that locality-based ranking is dramatically more useful. SRL extractors do not benefit from locality for binary extractions. For n-ary relations, all systems can improve precision by varying locality. Locality has little effect on nested relations.

This new assessor allows us to construct a high precision subset of TEXTRUNNER, which we use in the smart union (see the experiment above). For binary extractions we filter all extractions where locality > 5, *i.e.*, there are more than five tokens between the arguments.

An Ablation Study: SRL systems perform two key types of processing in addition to TEXTRUNNER's shallow syntactic techniques: (1) they use a full syntactic parser, and (2) they perform argument identification and classification. To tease apart the benefits, we perform an additional experiment in which we create extractions directly from the output of LUND-SRL's parser. These extractions achieve a precision of 0.62 at recall of 0.41 for binary extractions. This is much lower than SRL-IE-LUND's results (precision 0.7 and recall 0.51) illustrating the gain due to a complete SRL system.

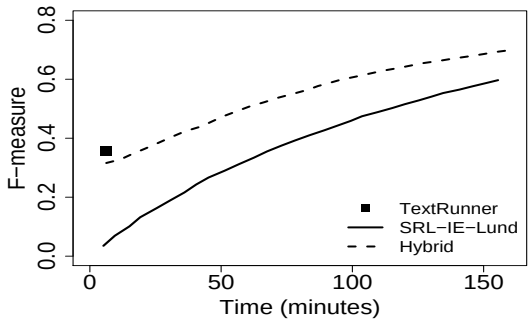

Figure 3: LargeCorpus: F-measure achieved in a given amount of computation time. The hybrid extractor obtains the best F-measure for binary extractions.

In summary, we find that SRL extractors perform better overall, however, TEXTRUNNER, under locality ranking, achieves superior precision at lower recalls. Neither redundancy nor locality benefits SRL extractors much (except for n-ary). SRL is orders of magnitude slower, which becomes a bottleneck in the next experiment, when the available time is limited.

5.2 LARGECORPUS Setting

To determine scaling to Web-scale knowledge extraction, we study an experimental condition, often not considered in NLP and IE communities – the setting in which the data is large enough that all processing cannot be achieved. Because experimenting with massive corpora makes it very hard to estimate (pseudo)recall and thus the F-measure, we simulate this setting by using our current data set, and varying the amount of available time.

In the extreme setup when the time is so limited that no extractor can complete processing, TEXTRUNNER is the extractor of choice, because the recalls of the slower extractors will be so low that they will far outweigh the benefits of higher precision. TEXTRUNNER, on the other hand, will be able to generate a large number of extractions at reasonable precision.

In the more interesting and likely more realistic case, where additional time is available after TEXTRUNNER completes its processing, we have the opportunity to combine the different extractors. We present a hybrid system that combines the strengths of TEXTRUNNER (fast processing time and high precision on a subset of sentences) with the strengths of SRL-IE-LUND (higher recall and better handling of long-range dependencies). The focus is on using the remaining time efficiently. We illustrate the binary setting, though results on n-ary are similar, and don't consider SRL-IE-UIUC owing to its very slow speed, though it is straightforward to include.

We first run TEXTRUNNER over all the sentences and then use the remaining time to run SRL-IE-LUND and take the union of all extractions. We add to this idea by

using a *filter policy* and an intelligent *order of sentences for extraction* to improve the precision.

TextRunner's precision is low when the redundancy of the extraction is low, and when the arguments are far apart. Thus, redundancy, and locality form the key factors for our filter policy: if both of these factors are below a given threshold, discard the tuple. The thresholds were determined by a parameter search over a small development set.

A good ordering policy would apply SRL-IE-Lund first to the sentences in which TextRunner extractions have been filtered by the filter policy. We could rank a sentence S according to the average distance between pairs of arguments from all tuples extracted by TextRunner from S. While this ranking system would order sentences according to their likelihood of yielding maximum new information, it would miss the cost of computation. To account for computation time, we additionally estimate the amount of time SRL-IE-Lund will take to process each sentence using a linear model trained on the sentence length. We then choose the sentence that maximizes information gain divided by its estimated computation time.

If the available time is minimal then our hybrid extractor reduces to TextRunner. If it is very large, then the hybrid is similar to the smart union of two systems (see SmallCorpus). In intermediate cases, hybrids make effective use of available time. Overall, the hybrid extractor run the best algorithm given the available computation time.

In summary, we find that all extractors are useful and, for best results, they should all be employed to varying degrees, which are based on the available time and their efficiency.

Evaluation: Figure 3 reports F-measure for binary extractions measured against available computation time. Hybrid has substantially better F1 scores than both TextRunner's and SRL-IE-Lund's demonstrating the power of combining extractors.

6. RELATED WORK

Open information extraction is a relatively recent paradigm and hence, has been studied by only a small number of researchers. The most salient is TextRunner, which also introduced the model [5, 6].

A recent Open IE system, WOE [21], uses dependency features (WOE^{parse}) and training data generated using Wikipedia infoboxes to learn a series of open extractors (WOE^{pos}). Our ablation study in Section 5.1 suggests the quality of the parser-based extractor to be between TextRunner and complete SRL systems; we expect WOE^{parse}, their better performing system, to be similar. Moreover, WOE does not output n-ary or nested extractions.

A paradigm related to Open IE is Preemptive IE [16]. While one goal of Preemptive IE is to avoid relation-specificity, Preemptive IE does not emphasize Web scalability, which is essential to Open IE.

A version of Knext uses heuristic rules and syntactic parses to convert a sentence into an unscoped logical form [19]. This work is more suitable for extracting common sense knowledge as opposed to factual information.

Another related system is Wanderlust [2]. After annotating 10,000 sentences parsed with LinkGrammar, it learns 46 general linkpaths as patterns for relation extraction. In contrast to our approaches, this requires a large set of hand-labeled examples.

We are the first to use SRL for Open IE, but its use for traditional IE is investigated by Harabagiu *et al.* [11]. They used a lexico-semantic feedback loop in a question-answering system for a set of pre-defined relations.

7. CONCLUSIONS

This paper investigates the use of Semantic Role Labeling for the task of Open Information Extraction. Although the two tasks were developed in isolation, they are quite related. We describe SRL-IE-UIUC and SRL-IE-Lund, the first SRL-based Open IE systems. We empirically study the trade-offs between these systems and TextRunner, a state-of-the-art Open IE system under several settings: SmallCorpus with unbounded computation time, and LargeCorpus with limited amount of time; using different metrics: precision, recall, F1 score and running time; and for different kinds of extractions: binary, n-ary and nested.

We find that in the traditional NLP setting (SmallCorpus), the deeper analysis of SRL-based systems overall outperforms TextRunner. However, TextRunner output can be ranked using our novel measure, *locality*, leading to superior precision at non-trivial recalls. A smart union of the three approaches performs best.

TextRunner is over an order of magnitude faster, making it the algorithm of choice when time is extremely limited. These complimentary strengths lead us to design a hybrid extractor that intelligently chooses sentences to extract from, and thus, efficiently uses the remaining computation time. Our hybrid extractor achieves better performance than either system if an intermediate amount of time is available for processing. Overall, we provide evidence that, contrary to belief in the Open IE literature [6], deep syntactic approaches have a lot to offer for the task of Open IE.

8. REFERENCES

[1] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM International Conference on Digital Libraries*, 2000.

[2] Alan Akbik and Jügen Broß. Wanderlust: Extracting semantic relations from natural language text using dependency grammar patterns. In *Proceedings of the WWW 2009 Workshop on Semantic Search*, 2009.

[3] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. In *COLING '98: Proceedings of the 17th international conference on Computational linguistics*, pages 86–90, 1998.

[4] Michele Banko. *Open Information Extraction for the Web*. PhD thesis, University of Washington, 2009.

[5] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2670–2676, 2007.

[6] Michele Banko and Oren Etzioni. The tradeoffs between open and traditional relation extraction. In *ACL '08: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 28–36, 2008.

[7] Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. Coupled semi-supervised learning for information extraction. In *WSDM '10: Proceedings of the Third ACM International Conference on Web Search and Data Mining*, 2010.

[8] Bonaventura Coppola, Alessandro Moschitti, and Giuseppe Riccardi. Shallow semantic parsing for spoken language understanding. In *NAACL '09: Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 85–88, 2009.

[9] Doug Downey, Oren Etzioni, and Stephen Soderland. A probabilistic model of redundancy in information extraction. In *IJCAI '05: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1034–1041, 2005.

[10] Oren Etzioni, Michele Banko, and Michael J. Cafarella. Machine reading. In *AAAI'06: Proceedings of the 21st national conference on Artificial intelligence*, pages 1517–1519, 2006.

[11] Sanda Harabagiu, Dan Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Răzvan Bunescu, Roxana Gîrju, Vasile Rus, and Paul Morărescu. The role of lexico-semantic feedback in open-domain textual question-answering. In *ACL '01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 282–289, 2001.

[12] Richard Johansson and Pierre Nugues. The effect of syntactic representation on semantic role labeling. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 393–400, 2008.

[13] Paul Kingsbury Martha and Martha Palmer. From treebank to propbank. In *LREC '02: Proceedings of the Third International Conference on Language Resources and Evaluation*, 2002.

[14] Alessandro Moschitti, Daniele Pighin, and Roberto Basili. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224, 2008.

[15] V. Punyakanok, D. Roth, and W. Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2), 2008.

[16] Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *NAACL '06: Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 304–311, 2006.

[17] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999.

[18] Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191, 2008.

[19] Benjamin Van Durme and Lenhart Schubert. Open knowledge extraction through compositional language processing. In *STEP '08: Proceedings of the 2008 Conference on Semantics in Text Processing*, pages 239–254, 2008.

[20] Daniel S. Weld, Raphael Hoffmann, and Fei Wu. Using wikipedia to bootstrap open information extraction. *SIGMOD Rec.*, 37(4):62–68, 2008.

[21] Fei Wu and Daniel S. Weld. Open information extraction using wikipedia. In *ACL '10: Proceedings of the 48th Annual Meeting on Association for Computational Linguistics*, 2010.