

# Machine Reading Between the Lines: A Simple Evaluation Framework for Extracted Knowledge Bases

Avirup Sil and Alexander Yates

Center for Data Analytics and Biomedical Informatics

Temple University

Broad St. and Montgomery Ave.

Philadelphia, PA 19122

{avirup.sil, yates}@temple.edu

## Abstract

The traditional method to evaluate a knowledge extraction system is to measure precision and recall. But this method only partially measures the quality of a knowledge base (KB) as it cannot predict whether a KB is *useful* or not. One of the ways in which a KB can be useful is if it is able to deduce implicit information from text which standard information extraction techniques cannot extract. We propose a novel, simple evaluation framework called “Machine Reading between the Lines” (MRbtL) which measures the usefulness of extracted KBs by determining how much they can help improve a relation extraction system. In our experiments, we compare two KBs which both have high precision and recall according to annotators who evaluate the knowledge in the KBs independently from any application or context. But, we show that one outperforms the other in terms of MRbtL experiments, as it can accurately deduce more new facts from the output of a relation extractor more accurately. In short, one extracted KB can read between the lines to identify extra information, whereas the other one cannot.

## 1 Introduction

Evaluating knowledge bases (KBs), and especially extracted KBs can be difficult. Researchers typically measure the accuracy of extracted KBs by measuring precision and recall. But this only partially measures the value of a KB. Size and correctness are important intrinsic measures, but a KB that states “1 is an integer, 2 is an integer, ...” contains an infinite number of correct facts, but is not very useful for most tasks. Researchers

have proposed a variety of applications as testbeds for evaluating the *usefulness* of knowledge bases, and the Recognizing Textual Entailment Challenge (Dagan et al., 2006) has received increasing attention as an interesting testbed (Clark et al., 2007). However, evaluating a knowledge base on RTE requires implementing a functioning RTE system, which is in itself a nontrivial task. Furthermore, even if a particular kind of knowledge could be useful for RTE, it may not help improve an RTE system’s score unless all of the other knowledge required for the complex inferences in this task are already present. In short, an effective KB evaluation framework is one that:

- is easy to implement
- is able to measure a KB’s utility on a valued application such as relation extraction

In response, we propose a task called “Machine Reading between the Lines” (MRbtL). In this task, a relation extraction system first extracts a base set of facts from a corpus. An extracted KB is then used to deduce new facts from the output of the relation extractor. The KB is evaluated on the precision and amount of “new” facts that can be inferred.

We also argue that MRbtL evaluation is more rigorous than asking an annotator to evaluate the usefulness of a stand-alone piece of knowledge, because it forces the annotator to consider the application of the knowledge in a specific context. In addition, success on MRbtL provides an immediate benefit to relation extraction, an area which many NLP practitioners care about.

## 2 Previous Work

The MRbtL task is similar in spirit to the vision of Peñas and Hovy (2010), but they focus on a background knowledge about the names for relations between two nouns. MRbtL also provides a

quantitative evaluation framework for the implicit extractions which is missing from the Peñas and Hovy work. Goyal *et al.* (2010) present a system for annotating stories with information about whether specific events are positive or negative for characters in the story. Viewed as an MRbtL task, they extract knowledge about whether actions and events cause people to be happy or unhappy, a very specific kind of knowledge. They then implement an inference technique, much more sophisticated than our variable substitution method, for applying this specific extracted knowledge to stories.

### 3 Machine Reading between the Lines

Let us assume that we have a knowledge base  $KB$  and a corpus  $C$  from which a Machine Reading or information extraction system has extracted a database of relational extractions ( $RE$ ). In the MRbtL framework, a KB is evaluated on the set of additional ground extractions, called *implicit extractions* ( $IE$ ) which are entailed by the KB:  $KB \wedge RE \models IE$ . MRbtL systems are then judged on the precision, amount, and *redundancy* of  $IE$ . By redundancy, we mean the fraction of extracted knowledge in  $IE$  that overlaps with  $RE$ , or is obvious *a priori*. If  $IE$  is *accurate* and contains many non-redundant extractions, then we judge  $KB$  to be a useful knowledge base. The advantage of this setup is that a system’s score on the task depends only on the  $KB$ , a relation extractor, and a simple logical inference engine for performing variable substitutions and *modus ponens*. These last two are often freely available or quite cheap to build. Formally, our three evaluation metrics are defined as follows:

$$Accuracy : \frac{|\text{Correct Extractions in } IE|}{|IE|}$$

$$Amount : |IE|$$

$$Redundancy : \frac{|IE \cap RE|}{|IE|}$$

Consider a very simple knowledge base which extracts knowledge about the President of a country. It has an axiom:  $\forall_{p,c} \text{president\_of}(p,c) \Rightarrow \text{person}(p) \wedge \text{country}(c)$ . Using this axiom, a relation extraction system can extract  $\text{president\_of}(\text{Obama}, \text{USA})$  which would then belong to  $RE$ . If  $RE$  also contains  $\text{person}(\text{Obama})$  extracted separately from the same sentence or document, then this extraction would be correct but redundant.

### 4 Evaluating Two STRIPS KBs with MRbtL

Common-sense knowledge about the changes in the state of the world over time is one of the most crucial forms of knowledge for an intelligent agent, since it informs an agent of the ways in which it can act upon the world. A recent survey of the common-sense knowledge involved in the recognizing textual entailment task demonstrates that knowledge about action and event semantics, in particular, constitutes a major component of the knowledge involved in understanding natural language (LoBue and Yates, 2011). In this section, we describe two example KBs of action and event semantics extracted by our previous work and also discuss an evaluation of these KBs using MRbtL.

We define *actions* as observable phenomena, or *events*, that are brought about by rational agents. One of the best-known, and still widely used, representations for action semantics is the STRIPS representation (Fikes and Nilsson, 1971). Formally, a STRIPS representation is a 5-tuple  $(a, \text{args}, \text{pre}, \text{add}, \text{del})$  consisting of the action name  $a$ , a list  $\text{args}$  of argument variables that range over the set of objects in the world, and three sets of predicates that reference the argument variables. The first, the *precondition* list  $\text{pre}$ , is a set of conditions that must be met in order for the action to be allowed to take place. For instance, in order for someone to *awaken*, she or he must first be *asleep*. The other two sets of conditions specify how the world changes when the action takes place: the *add* list describes the set of new conditions that must be true afterwards (e.g., after the event  $\text{insert}(\text{pencil24}, \text{sharpener3})$ ,  $\text{in}(\text{pencil24}, \text{sharpener3})$  holds true), and the *del* list specifies the conditions that were true before the action happened but are no longer true. These *add* and *del* conditions are sometimes collectively referred to as *effects* or *postconditions*.

Formally, the precondition, add and delete lists correspond to a set of rules describing the logical consequence of observing an event. Let  $t_1$  be the time point immediately preceding an event  $e$  with arguments  $\mathbf{args}$ ,  $t_2$  the time of event  $e$ , and  $t_3$  the time immediately following  $e$ . For each precondition  $p$ , each add effect  $a$ , and each delete effect  $d$ , the following rules hold:

$$\begin{aligned}
\forall \mathbf{args} e(\mathbf{args}, t_2) &\Rightarrow p(\mathbf{args}_p, t_1) \\
\forall \mathbf{args} e(\mathbf{args}, t_2) &\Rightarrow a(\mathbf{args}_a, t_3) \\
\forall \mathbf{args} e(\mathbf{args}, t_2) &\Rightarrow \neg d(\mathbf{args}_d, t_3)
\end{aligned}
\tag{1}$$

where  $\mathbf{args}_x$  represents the subset of the arguments to which the predicate  $x$  applies.

#### 4.1 Two extracted STRIPS KBs

We earlier introduced two different KBs that extract preconditions and postconditions (add and delete effects) of actions. One of the KBs (Sil et al., 2010) (henceforth, S10) uses candidate pre and postconditions which have high pointwise mutual information (PMI) with the action words. Given a corpus where each document contains an event  $e$ , S10 begins by identifying relations and arguments in a large text corpus using an open-domain semantic role labeler and OpenNLP’s noun-phrase coreference resolution system<sup>1</sup>. Taking a set of candidate predicate words, we then define different features of the labeled corpus that measure the proximity in the annotated corpus between a candidate word and the action word. Using a small sample of labeled action words with their correct preconditions and effects, we then train an RBF-kernel Support Vector Machine (SVM) to rank the candidate predicate words by their proximity to the action word.

But, S10 does not generalize adequately *e.g.* it extracts *hammer* as a precondition for the action *crush*. While it is true that if one has a hammer, then one can crush things, this is too strict of a precondition. Hence, we introduce another KB, HYPER (Sil and Yates, 2011), which adds generality to the extractions. HYPER uses Wordnet superclasses as additional candidates (potential pre and postconditions) of actions. Figure 1 shows sample STRIPS extractions from S10 and HYPER.

#### 4.2 MRbtL for S10 and HYPER

We now describe how we can build a MRbtL system for the extracted STRIPS representations. We use the set of predicates and their arguments discovered by the semantic role labeler used by S10 as explicit relational extractions *RE*; a number of off-the-shelf extractors are available for this purpose. Next, for each occurrence of one of the action words as a predicate in the corpus, we apply the axioms (1) and the knowledge in the S10 and

	amputate	crush
<b>args:</b>	$x, y$	$o, p$
<b>pre:</b>	organism#1( $x$ ), body_part#1( $y$ ), has( $x, y$ )	object#1( $o$ ), object#1( $p$ ), whole( $p$ )
<b>add:</b>	$\neg$ has( $x, y$ )	$\neg$ whole( $p$ )
<b>del:</b>	has( $x, y$ )	whole( $p$ )
<b>args:</b>	$a, b$	$m, n$
<b>pre:</b>	person( $a$ ), legs( $b$ ), has( $a, b$ )	hammer( $m$ ), ice( $n$ ), whole( $n$ )
<b>add:</b>	$\neg$ has( $a, b$ )	$\neg$ whole( $n$ )

Figure 1: Two example STRIPS representations extracted by the HYPER system (above), and representations for the same actions extracted by our prior work, S10 (below). In contrast with S10, the HYPER representations require extracting delete effects. Also, HYPER disambiguates and generalizes predicates by identifying WordNet synsets for predicate names. Here, *organism* and *object* (in HYPER) are hypernyms of *person* and *hammer* (in S10) respectively.

HYPER KBs to deduce predicates that must be true immediately before or after the occurrence of the action. For example, the semantic role labeler discovers the formula  $draining(a_0, a_1) \wedge the\ acid\ solution(a_1)$  from the sentence, “This is done by inverting the battery and draining the acid solution out the vent holes in the battery cover”. By applying the extracted precondition that the second argument of a *draining* event must be a liquid, we can infer that  $liquid(a_1)$  is true immediately before the event. Since our MRbtL setup extracts tens of thousands of implicit facts, we evaluate precision and redundancy on samples.

## 5 Experiments

We perform MRbtL experiments on extractions from S10 and HYPER. S10 uses a dataset of 40 actions from the lexical units in the frames that inherit from the *Transitive\_action* frame in FrameNet. The document collection has 15,088 documents downloaded from the Web for the 40 action words. We use the annotated Web corpus for HYPER with semantic role information. We measure the quality of our implicit extractions by taking a random sample of 100 and having two judges classify each extraction for accuracy and redundancy (as per the definitions in Sec 3) in the context of the sentence and document from which it was extracted. As per our earlier work, the pre-

<sup>1</sup><http://opennlp.sourceforge.net>

	S10	HYPER	$\kappa$	signif.
accur.	45%	<b>73%</b>	0.65	$p < 0.01$
redun.	21%	<b>12%</b>	0.91	$p = .13$
num.	54,300	<b>67,192</b>	N/A	N/A

Table 1: **The knowledge base extracted by HYPER can identify more, and more accurate, implicit extractions than S10’s knowledge base, and fewer implicit extractions overlap with explicit extractions.** The first two columns record the accuracy and redundancy (averaged over two annotators on sample of 100), and total number of implicit extractions.  $\kappa$  indicates Cohen’s  $\kappa$  inter-annotator agreement score, and  $p$ -values for the significance tests are calculated using a two-sided Fisher’s exact test.

cision of S10 is 93% at 85% recall, whereas for HYPER the precision is 89% at 90% recall. At a first glance, both of the systems look impressive to someone by just looking at the precision/recall numbers.

Table 1 shows the results of our Machine Reading between the Lines experiment. These extractions are based on 15,000 occurrences of the 40 action words, but as we scale the extractors to new action words, we should increasingly be able to read between the lines of texts. Hence, we observe that even when both S10 and HYPER report similar (and high) precision and recall, they report significantly different scores on MRbtL experiments. From Table 1, we clearly see that HYPER outperforms S10. HYPER’s implicit extractions are nearly 30% more accurate than S10’s, and roughly half as redundant. Extrapolating from the accuracy and redundancy scores in the evaluated sample, HYPER extracts 41,659 correct, non-redundant relationships compared with 7602 extractions for S10 from the Web corpus that does not appear explicitly in the documents.

Example extractions indicate that HYPER’s stronger performance on MRbtL is because its extracted pre and postconditions generalize to hypernyms. From the sentence “Doctors need to heal patients.”, HYPER extracts *medical\_practitioner(doctors)* indicating that *doctors* are of type *medical\_practitioner* which is an accurate and non-redundant extraction. Here, *medical\_practitioner* is a precondition for action *heal*. But S10 concludes that *doctors* are of type *doctor* (a Wordnet subclass of *medical\_practitioner*) which is a redundant extraction. Another example: from the sentence “When a sharp object, like

a fingernail or thorn, scrapes along your skin . . .”, the MRbtL system extracted that the fingernail is an object, since the instrument of a scraping action needs to be an object. Both annotators considered this extraction correct, but redundant, since the sentence explicitly mentions that a fingernail is a kind of object.

## 6 Conclusion and Future Work

We show that the extracted knowledge base can be used to accurately identify information in a document that is never stated explicitly. We call this evaluation scenario “Machine Reading between the Lines”. We demonstrate that HYPER’s extracted knowledge base outperforms the closest comparable one though both perform extremely well when measured under only precision and recall. A future direction includes comparing very different KBs with MRbtL.

## References

- Peter Clark, William R. Murray, John Thompson, Phil Harrison, Jerry Hobbs, and Christiane Fellbaum. 2007. On the role of lexical and world knowledge in rte3. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE ’07, pages 54–59, Morristown, NJ, USA. Association for Computational Linguistics.
- I. Dagan, O. Glickman, and B. Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. *Lecture Notes in Computer Science*, 3944:177–190.
- R. Fikes and N. Nilsson. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3/4):189–208.
- Amit Goyal, Ellen Riloff, and Hal Daumé III. 2010. Automatically producing plot unit representations for narrative text. In *EMNLP*.
- Peter LoBue and Alexander Yates. 2011. Types of common-sense knowledge needed for recognizing textual entailment. In *ACL*.
- Anselmo Pe nas and Eduard Hovy. 2010. Semantic enrichment of text with background knowledge. In *Proceedings of the NAACL Workshop(FAM-LBR)*.
- Avirup Sil and Alexander Yates. 2011. Extracting strips representations of actions and events. In *RANLP*.
- Avirup Sil, Fei Huang, and Alexander Yates. 2010. Extracting action and event semantics from web text. In *AAAI Fall Symposium on Common-Sense Knowledge (CSK)*.