Robust textual inference via learning and abductive reasoning

Rajat Raina, Andrew Y. Ng and **Christopher D. Manning**
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

We present a system for textual inference (the task of inferring whether a sentence follows from another text) that uses learning and a logical-formula semantic representation of the text. More precisely, our system begins by parsing and then transforming sentences into a logical formula-like representation similar to the one used by (Harabagiu et al., 2000). An abductive theorem prover then tries to find the minimum "cost" set of assumptions necessary to show that one statement follows from the other. These costs reflect how likely different assumptions are, and are learned automatically using information from syntactic/semantic features and from linguistic resources such as WordNet. If one sentence follows from the other given only highly plausible, low cost assumptions, then we conclude that it can be inferred. Our approach can be viewed as combining statistical machine learning and classical logical reasoning, in the hope of marrying the robustness and scalability of learning with the preciseness and elegance of logical theorem proving. We give experimental results from the recent PASCAL RTE 2005 challenge competition on recognizing textual inferences, where a system using this inference algorithm achieved the highest confidence weighted score.

Introduction

The performance of many text processing applications would improve substantially if they were able to reason with natural language representations. These applications typically use information specified in natural language (e.g., broad-coverage question answering), and process input/output in natural language (e.g., document summarization).

Several semantic reasoning tasks crop up within these applications. For example, in an automatic question answering system, a question is given and we might require the system to find a piece of text from which the answer can be inferred. In a document summarization system, we might stipulate that the summary should not contain any sentences that can be inferred from the rest of the summary. These semantic reasoning tasks are largely addressed one application at a time, despite the many intersections.

There have been recent suggestions to isolate the core reasoning aspects and formulate a generic task that could potentially be useful for many of the above text processing applications (PASCAL RTE Challenge, PASCAL Recognizing Textual Entailment Challenge 2005). Along these lines, the basic task we consider is to decide whether the meaning of one natural language fragment can be approximately inferred from another. The most general case of natural language reasoning might require rich commonsense reasoning, which is known to be hard. Our focus is on inferences that can be made using language features and semantics, possibly with some world knowledge.

In this paper, we present an abductive inference algorithm to perform semantic reasoning. More precisely, it learns how to combine information from several knowledge sources to decide what are plausible "assumptions" during its reasoning process. Thus, our system is capable of performing flexible semantic reasoning, given just a training set of inferable sentences (see Table 1 for examples).

Even beyond natural language inference, our methods can be used for robust inference with other logical representations. Logical inference is well known to frequently be brittle and have poor coverage, especially when it uses axioms that must be manually tweaked; this limits its applicability to real-world tasks. However, by using abduction and learning, we combine the elegance and preciseness of the logical formalism with the robustness and scalability of a statistically trained system.

Problem definition

We describe our algorithms on the task of "recognizing textual entailment" (PASCAL RTE Challenge, PASCAL Recognizing Textual Entailment Challenge 2005), but the ideas extend easily to the general case of textual inference.

Each example in this domain consists of two parts: one or more *text* sentences and a *hypothesis* sentence. The task is to identify whether the hypothesis sentence can be plausibly inferred ("entailed") from the text sentences. Throughout this section, we use the following as our running example:

```
TEXT: Bob purchased an old convertible.
HYPOTHESIS: Bob bought an old car.
```

Actual examples in the task are significantly more complex; see the examples in Table 1.

Many successful text applications rely only on "keyword" (or phrasal) counting. But information retrieval techniques that use only word counts (such as bag-of-words representa-

Class	Text	Hypothesis	Entailed
Comparable Documents	A Filipino hostage in Iraq was released.	A Filipino hostage was freed in Iraq.	Yes
Information Extraction	Global oil prices clung near their highest levels in at least 21 years yesterday after Russian oil giant Yukos was ordered to stop sales, ...	Oil prices rise.	Yes
Information Retrieval	Spain pulled its 1,300 troops from Iraq last month.	Spain sends troops to Iraq.	No
Machine Translation	The economy created 228,000 new jobs after a disappointing 112,000 in June.	The economy created 228,000 jobs after disappointing the 112,000 in June.	No
Paraphrase Acquisition	Clinton is a very charismatic person.	Clinton is articulate.	Yes
Question Answering	VCU School of the Arts In Qatar is located in Doha, the capital city of Qatar.	Qatar is located in Doha.	No
Reading Comprehension	Eating lots of foods that are a good source of fiber may keep your blood glucose from rising fast after you eat.	Fiber improves blood sugar control.	Yes

Table 1: Some illustrative examples from different classes of the PASCAL RTE dataset (PASCAL RTE Challenge, PASCAL Recognizing Textual Entailment Challenge 2005).

tions) are not well suited to such semantic reasoning tasks. For example, a successful system must differentiate between the hypothesis above and a similar-looking one which cannot be inferred:

`HYPOTHESIS: Old Bob bought a car.`

Thus, rather than relying on keyword counting, we use a significantly richer representation for the syntax of the sentence, and augment it with semantic annotations. We choose a logical formula-like representation similar to (Harabagiu, Pasca, & Maiorano, 2000; Moldovan *et al.*, 2003), which allows us to pose the textual inference problem as one of "approximate" (or, more formally, abductive) logical inference. For example, the sentences above become:

```
(∃ A,B,C) Bob(A) ∧ convertible(B) ∧ old(B)
∧ purchased(C,A,B)
(∃ X,Y,Z) Bob(X) ∧ car(Y) ∧ old(Y) ∧
bought(Z,X,Y)
```

In our notation, `C` and `Z` are event variables that denote the events of purchasing and buying, respectively.

With this representation, the hypothesis is entailed by the text if and only if it can be logically proved from the latter.

However, the main obstacle to using such logical inferences is that most non-trivial entailments require making certain "leap-of-faith" assumptions. Thus, in order to correctly infer the entailment above, one must either know or assume that "*a convertible is a car*". Building on (Hobbs *et al.*, 1993), we propose an algorithm in which each "assumption" is associated with a cost, and a hypothesis is plausible if it has a simple—meaning low-cost—proof.

Instead of representing all possible leaps-of-faith as explicit logical axioms ("rules"), we take the view that these leaps-of-faith correspond to assumptions about the real world and have some degree of plausibility; we assign a cost to each assumption to quantify its plausibility. The cost of an assumption is computed using a cost model that integrates features of the assumption based on many knowledge sources. We believe this approach is more robust and

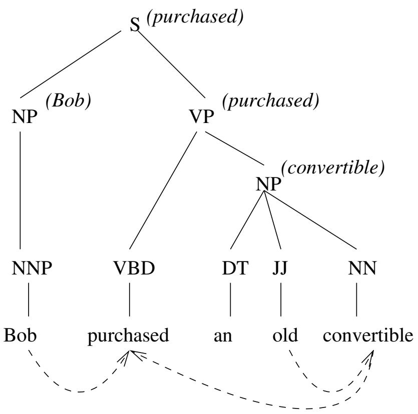

Figure 1: Procedure to discover syntactic dependencies, illustrated on the parse tree for an example sentence. Heads are marked on nonterminal nodes in parentheses. The dotted lines show the dependencies discovered.

scalable than one which requires axioms that are always true (Moldovan *et al.*, 2003). Given this cost model, we can then use a search procedure to find the minimum-cost proof, and judge entailment based on this cost.

One of the challenges is to obtain a cost model for a large set of assumptions. Such costs have typically been manually tuned in previous work, or have been left out completely. We describe a learning algorithm that automatically discovers "good" costs given a training set of labeled examples (as in Table 1).

Representation

In this section, we sketch how our logical representation is derived from raw English text.

The first step in processing a sentence is to construct a syntactic dependency graph. The sentence is parsed using the parser in (Klein & Manning, 2003). Hand-written rules are used to find the heads of all nodes in this parse tree (these reles are an adaptation of those in Collins, 1999).

This procedure implicitly discovers syntactic relationships between the words in the sentence, because when a word is chosen as the head, its siblings must be syntactic modifiers to that word. These relationships can be laid out as a *dependency graph* where each node is a word/phrase and the links represent particular dependencies. Figure 1 illustrates this procedure on our example text sentence.

We now translate the relations represented in the dependency graph into a logical formula-like representation, as in (Harabagiu, Pasca, & Maiorano, 2000). Each node in the graph is converted into a logical term and is assigned a unique constant. Edges in the graph are represented by sharing arguments across nodes; verbs and prepositions can have multiple arguments, and receive arguments from all the linked nodes.[1]

To help the inference process, we can also augment the logical formula with several kinds of semantic annotations. These include annotations added on the predicate (e.g., if the corresponding word is part of a Named Entity), and also annotations added on the arguments of certain terms (e.g., if that argument has a subject/object relation to the predicate of the term). These annotations can be used as features in determining assumption costs in the inference process. For our running example, we discover, for example, that `Bob` is a Person (using a Named Entity classifier), and that `convertible` is the object to the verb `purchased` (using the syntactic dependency "types").

Using this restricted representation, we can accurately capture only a subset of all English language semantics. For example, hypotheses such as *"Bob did not buy an old car"* are also converted to a similar representation (without any negated terms) – the negation is spotted from the parse tree and annotated on the verb *buy*; this annotation is used later to achieve the effect of negation. These restrictions imposed on the representation will allow us to use more tractable inference algorithms later.

Inference

Strict theorem proving

To motivate the form for our abductive assumptions, assume first that we are not allowed to make assumptions in the proof.

Then, for the remaining inference problem, we must find a proof for the hypothesis given the text. The method of resolution refutation performs theorem proving by adding the negation of the goal logical formula to a knowledge base consisting of the given axioms, and then deriving a null clause through successive resolution steps. This corresponds to justifying (i.e., "proving") the goal by deriving a contradiction for its negation. Thus, to use the method of resolution refutation, we construct a knowledge base with the text logical formula and the negation of the hypothesis logical formula; finding a proof then corresponds to producing the null clause through successive resolution steps. For our example, we get the following clauses:

(∃ A,B,C) Bob(A) ∧ convertible(B) ∧ old(B) ∧ purchased(C,A,B)

(∀ X, Y, Z) ¬Bob(X) ∨ ¬car(Y) ∨ ¬old(Y) ∨ ¬bought(Z,X,Y)

By construction, our representation converts each sentence into a conjunction of logical terms; thus, each clause in the above knowledge base has at most one non-negated term. In other words, the knowledge base contains only Horn clauses. Unit resolution is a proof strategy that considers only resolution steps in which at least one of the participating clauses is a unit clause (i.e., has one term). It is known that unit resolution is a complete proof strategy for this restricted class of logical clauses. (see, e.g., Genesereth and Nilsson, 1987).

For all the theorem proving problems produced by our logical formulae, it is thus sufficient for completeness to consider only proof steps that unify a single-term clause produced from the text with the first term of the clause produced from the hypothesis.[2] Further, this is also a tractable (i.e., polynomial-time) proof strategy for this (non-abductive) setting.

Abductive theorem proving

The previous proof strategy considers unifications between a single-term clause and the first term of another clause only. With the standard logical definition of unification, this strategy is too strict for our language representation – we would like to "unify" terms such as `purchased(C,A,B)` and ¬`bought(Z,X,Y)` in our proofs. We thus consider the setting of weighted abduction (Hobbs *et al.*, 1993), where we allow such pairs of terms to "unify" at some computed cost.

In particular, consider two logical terms $S(s_1, s_2, \ldots, s_m)$ and $\neg T(t_1, t_2, \ldots, t_n)$, where S, T are the predicates and $s_{1\ldots m}$, $t_{1\ldots n}$ are the arguments (variables or constants). The standard definition of unification requires that $S = T$, $m = n$ and each s_i be consistently unified with t_i. It seems useful to relax the standard definition of unification in the following ways:

1. The predicates S and T are allowed to be different. e.g.: S and T might be synonyms of each other.

[1] In our implementation, the conversion considers further linguistic phenomena in constructing the logical formula. For example, it propagates dependencies by preposition folding. These modifications were driven by data analysis, and details are omitted due to space constraints. To give an example of a complex sentence, the last text sentence in Table 1 is transformed to:
eating(A) lots(A) of(B,A,C) foods(C) that(D) are(E,C,D,F) good(F) source(F) of(G,F,H) fiber(H) may(I) keep(J,A,I,K,L) your(M) blood_glucose(K) from(N,A,L) rising(L,O,P,Q) too(O) fast(P) after(R) you(S) eat(Q,R,S)

[2] The following facts allow us to conclude this:

1. Unit resolution is complete for our problems.
2. Each proof step must resolve a negated and a non-negated term. Non-negated terms arise only from the text as unit clauses. Negated terms arise only from the negated hypothesis.

2. The numbers of arguments m and n need not be the same, and each s_i may be unified with some argument other than t_i. The same pair of terms may unify in several ways corresponding to the different ways of matching their arguments.
   e.g.: Two verbs that convey the same meaning might take different numbers of modifiers, the order of the modifiers might be changed and the correspondence of the modifiers across the terms might be ambiguous.
3. Two constant arguments could unify with each other.
   e.g.: The logical representation might have two constants that actually represent the same physical entity through coreference.

Each of the above relaxations is interpreted as an abductive assumption about the world, and its degree of plausibility is quantified as a nonnegative cost by the *assumption cost model*. The cost model is responsible for tackling language semantics and is described in the next section.

Given a cost model, abductive theorem proving can be framed as a search problem. For every pair of terms, there might be many resulting resolvents (each with one way of matching the term arguments) and each resolvent is assigned a nonnegative cost by the cost model. A proof is complete when the null clause is reached through successive steps; the total cost of the proof is given by the sum of individual step costs. Since we are interested in finding a minimum cost proof, the theorem prover can use uniform cost search to discover it efficiently. In general, A^* search can also be used for large-scale problems if a good admissible heuristic is available.

Assumption cost model

The assumption cost model quantifies the plausibility of any given assumption. As described in the previous section, an assumption $\mathcal{A}$ is identified by the two logical terms being unified (say $S(s_1, s_2, \ldots, s_m)$ and $\neg T(t_1, t_2, \ldots, t_n)$) and the argument matching under consideration. We assign a cost $\mathcal{C}_w(\mathcal{A})$ to assumption $\mathcal{A}$ as a linear function of *features* of $\mathcal{A}$:

$$\mathcal{C}_w(\mathcal{A}) = \sum_{d=1}^{D} w_d f_d(\mathcal{A}) \tag{1}$$

where $f_1, \ldots, f_D$ are *arbitrary* nonnegative feature functions and $w_1, \ldots, w_D$ are the relative weights assigned to these feature functions. The linear combination of features $f_d(\mathcal{A})$ allows each w_d to be interpreted as the cost for a particular kind of basic assumption. Further, we show in the sequel that "good" weights w_d can be learnt automatically for this cost function.

The features can be derived from a number of knowledge sources, such as WordNet (Miller, 1995), syntactic features, etc. that measure the degree of similarity between two terms. Table 2 lists the features used in our experiments. These features can be broadly divided into the following five classes:

1. Predicate "similarity": We use (Resnik, 1995; Pedersen, Patwardhan, & Michelizzi, 2004) to compute a nonnegative measure of similarity between the two predicates based on their proximity in the WordNet hierarchy. This feature indicates when two words are synonyms or otherwise have similar meanings. An additional feature indicates whether the predicates are listed as antonyms in WordNet (properly accounting for any negation annotations on either predicate).
2. Predicate compatibility: These features measure if the two predicates are the same "type" of word. More specifically, three features indicate if the two words (i) have the same part-of-speech tag, (ii) the same word stem, and (iii) the same named entity tag (if any).
3. Argument compatibility: Several features consider each pair of arguments matched with each other, and penalize mismatches between annotations attached to them. For example, the annotation might be the type of verb dependency, and then we would prefer a subject argument to be matched with another subject argument; similarly, the annotation might be the semantic role of that argument, and we would prefer a location modifier argument to be matched with another location modifier.
4. Constant unification: Different constants in our representation might refer to the same physical entity, say because of anaphoric coreference. We thus precompute a matrix of "distances" between constants, using cues from coreference or appositive reference, for example. A feature function of the assumption computes the sum of distances for all constant unifications among the matched arguments.
5. Word frequency: Some terms in the hypothesis may not unify well with any term in the text, but can be "ignored" at some cost because they are used very commonly in language (e.g.: *"The watch is good."* might be considered to entail *"The watch is rather good."* as *rather* is a common word.) To capture this behavior, we compute a feature that is inversely proportional to the relative frequency of the hypothesis predicate in a large corpus of English text.

For illustration, we show how our representation and inference steps can be extended to tackle somewhat more complicated language phenomena. Consider the case of event nouns. These are cases where a verb in the hypothesis is represented as a noun in the text – for example, *"murder of police officer"* entails *"Police officer killed."*. The representation for text sentences is augmented by looking up the noun→verb derivational forms in WordNet for all nouns in the text, and finding the possible dependencies for that verb. The usual predicate similarity routines then work with this augmented representation. Such techniques can be devised to tackle several other linguistic constructions, but these extensions are not important for the current discussion.

Learning good assumption costs

Our discussion thus far has assumed that the weight vector $w = (w_1, \ldots, w_D)^T$ is given. We now describe a learning algorithm for automatically choosing these weights.

Discriminative learning of weights

We first introduce some notation. Consider an entailment example; any abductive proof $\mathcal{P}$ of the hypothesis consists of a sequence of assumptions, say $\mathcal{A}_1, \ldots, \mathcal{A}_N$. Us-

1. Similarity score for S and T.
   Are S and T antonyms?
   If S and T are numeric, are they "compatible"?
2. Mismatch type for part-of-speech tags of S and T.
   Do S and T have same word stem?
   Do S and T have same named entity tag?
3. Difference in number of arguments: $|m - n|$.
   Number of matched arguments with:
   – different dependency types.
   – different semantic roles.
   – each type of part-of-speech mismatch.
   Number of unmatched arguments of T.
4. Total coreference "distance" between matched constants.
5. Inverse word frequency of predicate S.
   Is S a noun, pronoun, verb or adjective, and is it being "ignored" by this unification?

Table 2: List of features extracted in our experiments for unifying terms $S(s_1, s_2, \ldots, s_m)$ and $\neg T(t_1, t_2, \ldots, t_n)$ according to some specified argument matching. The features specified as questions are binary features that are equal to 1 if the condition is true, and 0 otherwise. Part-of-speech mismatch features are computed by binning the part-of-speech pair into 10 "types" (e.g., "same part-of-speech", "noun/pronoun with verb", etc.).

ing the previously defined per-assumption feature functions $f_d(\mathcal{A}_1) \ldots f_d(\mathcal{A}_N)$ for each $d \in \{1, 2, \ldots, D\}$, we can compute the aggregated feature functions for the proof $\mathcal{P}$:

$$f_d(\mathcal{P}) = \sum_{s=1}^{N} f_d(\mathcal{A}_s) \tag{2}$$

Let $f(\mathcal{P}) = (f_1(\mathcal{P}) \ldots f_D(\mathcal{P}))^T$ denote the aggregated feature vector for the proof $\mathcal{P}$. Then, the total cost of the proof is a linear function of weights w:

$$\mathcal{C}_w(\mathcal{P}) = \sum_{d=1}^{D} w_d f_d(\mathcal{P}) = w^T f(\mathcal{P}) \tag{3}$$

Assume for simplicity that we augment the feature vector with a constant feature, so that we classify a hypothesis as ENTAILED if and only if its minimum proof cost is less than zero.

Suppose we are given a labeled dataset of entailment examples $(\tau^{(i)}, y^{(i)})$, where each $\tau^{(i)}$ represents a text-hypothesis pair and $y^{(i)} \in \{0, 1\}$ is the corresponding label (say $y^{(i)} = 1$ implies $\tau^{(i)}$ is ENTAILED, while $y^{(i)} = 0$ implies $\tau^{(i)}$ is not ENTAILED). For text-hypothesis pair $\tau^{(i)}$ and assumption weights w, our label prediction is based on the cost of the minimum cost proof:

$$\mathcal{P}_w^{(i)} = \arg\min\, w^T f(\mathcal{P}^{(i)}) \qquad \text{s.t. } \mathcal{P}^{(i)} \text{ is a proof for } \tau^{(i)}.$$

Using the sigmoid function $\sigma(z) = 1/(1 + \exp(-z))$, we can assume a logistic model for the prediction:

$$P(y^{(i)} = 0 | \tau^{(i)}, w) \triangleq \sigma(w^T f(\mathcal{P}_w^{(i)})) \tag{4}$$

and use this definition to optimize the discriminative log-likelihood of the training set (possibly with regularization):

$$\ell(w) = \sum_i \log P(y^{(i)} | \tau^{(i)}, w) \tag{5}$$

Unfortunately, the feature vector $f(\mathcal{P}_w^{(i)})$ for each example depends on the current weight vector w, and it can be shown that the log-likelihood function $\ell(w)$ is not convex in w. There are local maxima, and exact optimization is intractable (Rockafellar, 1972). Intuitively, as the weights w change, the feature vector $f(\mathcal{P}_{min}^{(i)})$ itself might change if a new proof becomes the minimal cost one from among the large number of possible proofs.

An iterative approximation

We approximate the above optimization problem to get a tractable solution. To deal with the large number of proofs, we optimize iteratively over proofs and weights. First we fix the current minimal proof $\mathcal{P}_w$, and use this fixed proof to analytically compute the local gradient of the (regularized) log-likelihood function $\ell(w)$. This is possible since, in general, the minimum cost proof stays the same in the immediate vicinity of the current weight vector w.[3] We update w by taking a short step along the direction of the gradient, producing weights with higher likelihood.

Since the minimum cost proofs might have changed at the new weight setting, we recompute these proofs using the abductive theorem prover. We now iterate, since the gradient of the log-likelihood function for the new weights can be computed using the new proofs.

Intuitively, the algorithm finds the assumption types that seem to contribute to proofs of entailed examples, and lowers their cost; similarly, it raises the cost for assumption types that seem to mislead the theorem prover into discovering low-cost proofs of non-entailed examples.

The proposed algorithm is shown below with a Gaussian prior for regularization. The algorithm uses the abductive theorem prover iteratively. The likelihood function $\ell(w)$ can have local maxima, so we start at a "reasonable guess" $w^{(0)}$.

1. Initialize $w^{(0)}$, set $k = 0$, choose step size α and a regularization parameter λ.
2. Using weights $w^{(k)}$ in the abductive theorem prover, find the minimum cost proofs $\mathcal{P}_{min}^{(i)}$ for each text-hypothesis pair $\tau^{(i)}$ in the training set.
3. Using the features computed on the minimum cost proofs above, move the weights in the direction of increasing approximate log-likelihood $\tilde{\ell}(w)$ of training data.

$$\tilde{\ell}(w) \quad \triangleq \quad \sum_{i:y^{(i)}=0} \log(\sigma(w^T f(\mathcal{P}_{min}^{(i)}))) +$$

[3] More formally, if $y = 1$ for a given training example, then letting $P_w = \arg\min_{\mathcal{P}} \sigma(w^T f(\mathcal{P}))$ be the current minimal cost proof (the $\arg\min$ should be replaced by an $\arg\max$ if instead $y = 0$), we have that on all but a measure zero set of points, $\nabla_w \sigma(w^T f(P_w)) = \nabla_w \arg\min_{\mathcal{P}} \sigma(w^T f(\mathcal{P}))$. In fact, our procedure is, almost everywhere, computing the true gradient of $\ell(w)$, and thus can be viewed as an instance of (sub)gradient ascent.

$$\sum_{i:y^{(i)}=1} \log(1-\sigma(w^T f(\mathcal{P}^{(i)}_{min}))) + \lambda||w||^2$$

$$w^{(k+1)} \quad \leftarrow \quad w^{(k)} + \alpha \left. \frac{\partial \tilde{\ell}(w)}{\partial w} \right|_{w=w^{(k)}}$$

4. $k \leftarrow k+1$

5. Loop to step 2 until convergence.

One final detail is that the weights (except the constant offset) must be nonnegative (for uniform cost search to work in theorem proving). So, we just set any negative components to 0 at the end of each iteration; the overall update direction is still a subgradient of the log-likelihood function.

Since the algorithm essentially performs gradient ascent on the (regularized) log-likelihood function for w, it is guaranteed to reach a local optimum of the original log-likelihood $\ell(w)$ if the step size is sufficiently small.

Results

We report results on the PASCAL Recognizing Textual Entailment (RTE) dataset, which was used in a recent challenge on recognizing textual entailment (PASCAL RTE Challenge, PASCAL Recognizing Textual Entailment Challenge 2005)[4].

The development set contains 567 examples of labeled text-hypothesis pairs, while the test set contains 800 examples. These examples are roughly equally divided into 7 classes, with each class derived from different sources. For example, the Information Extraction class is constructed by taking a news story sentence and then framing a simple relation entailed by the sentence. Table 1 lists some examples from this dataset.

The RTE dataset is known to be hard – all entrants to the challenge achieved only 50-60% accuracy on the test set. Many of the standard information retrieval algorithms are reduced to random guessing, pointing to the need for deeper semantic reasoning. Further, some of the classes appear to be significantly tougher than the others.

Experiments

We test our algorithm and some baseline algorithms:

- Random guessing: All decisions are made randomly.
- Term Frequency (TF): A standard information retrieval style method that represents each sentence as a vector of word counts, and finds a similarity between sentences (text and hypothesis) using the angle between these vectors. The hypothesis is assigned a "cost" according to its similarity with the most similar text sentence. For the RTE dataset, the method was applied once separately to each class and again to all classes together; only the better accuracy is reported.
- Term Frequency + Inverse Document Frequency (TFIDF): This is the same as TF, except that each word count is scaled so that rare words get more weight in the sentence vectors. This could potentially help match the "important" parts of the sentences better.
- Abductive theorem prover (ThmProver): This is the algorithm described in this paper. We picked the cost threshold (the cost above which an example is judged not-entailed) in two ways: in ThmProver1, we found a single threshold for all classes; in ThmProver2, a separate threshold was trained per class.
- Partially abductive theorem provers: To gauge the importance of our abductive assumptions, we disallow some of the assumptions possible. Partial1 allows the logical arguments to unify only according to strict (i.e., standard) logical rules. Partial2 allows unification only when the predicates match exactly.

Algorithm	RTE Dev Set		RTE Test Set	
	Acc	CWS	Acc	CWS
Random	50.0%	0.500	50.0%	0.500
TF	52.1%	0.537	49.5%	0.548
TFIDF	53.1%	0.548	51.8%	0.560
ThmProver1	**57.8%**	0.661	55.5%	0.638
ThmProver2	56.1%	**0.672**	**57.0%**	**0.651**
Partial1	53.9%	0.535	52.9%	0.559
Partial2	52.6%	0.614	53.7%	0.606

Table 3: Performance on the RTE datasets; CWS stands for confidence weighted score (see footnote 5).

We report the raw accuracy and the confidence weighted score (CWS) in Table 3.[5] Table 4 shows the performance of the theorem prover split by RTE example class (as illustrated in Table 1).

Class	RTE Dev Set		RTE Test Set	
	Acc	CWS	Acc	CWS
CD	71.4%	0.872	79.3%	0.906
IE	50.0%	0.613	49.2%	0.577
IR	50.0%	0.523	50.0%	0.559
MT	53.7%	0.708	58.3%	0.608
PP	62.2%	0.685	46.0%	0.453
QA	54.4%	0.617	50.0%	0.485
RC	47.6%	0.510	53.6%	0.567

Table 4: Performance of the overall best theorem prover on individual classes of the RTE dataset.

On the PASCAL RTE dataset, our system attains a CWS of 0.651, which is significantly higher than that attained by all other research groups in the competition, the next best score being 0.617. (Our research group's submission to the competition, using the algorithm described in this paper and also an additional inference algorithm, actually attained a CWS of 0.686 in the competition, thus coming in first place

[4]http://www.pascal-network.org/Challenges/RTE/

[5]CWS is a recommended measure for the RTE dataset. It is obtained by sorting all the confidence values, say $\{c_1, \ldots, c_n\}$, and then computing the "average precision": $1/n \sum_i$ (accuracy within i most confident predictions). This lies in $[0, 1]$ and is higher for better calibrated predictions.

on this evaluation metric.) We attribute this to the learning procedure optimizing a likelihood function and not a raw accuracy value. Our accuracy of 57% is also competitive with the best reported results. (PASCAL RTE Challenge, PASCAL Recognizing Textual Entailment Challenge 2005) (The best reported accuracy on this test set is 58.6%, and the official Stanford submission to the competition had an accuracy of 56.3%.) Interestingly, the performance varies heavily by class, (see Table 4), possibly indicating that some classes are inherently more difficult.

The baseline accuracy is close to random guessing, and the difference between our system performance and the baseline performance on the test set is statistically significant ($p < 0.02$). Further, the partially abductive theorem prover versions show the utility of using abductive assumptions.

For practical applications such as question answering, the end user might require that inferences be accompanied with human-readable justifications. In such cases, the theorem prover is especially useful, as its minimum cost proof generally provides a good justification for its inferences. For our simple running example, the minimum cost proof of the hypothesis can be translated into a justification such as the following:

"Bob bought an old car" can be inferred using the following assumptions:

- A convertible is a car.
- "A purchased B" implies "A bought B".

Discussion and Related Work

We have already compared our work with previous work on logical representations of natural language and on weighted abduction. We believe that the current work provides a much-needed missing link by using a learning algorithm to aid abductive inference, all over a rich feature space that uses diverse linguistic resources.

Several dependency graph-based representations have been used for question answering (Punyakanok, Roth, & Yih, 2004) and for recognizing textual entailment (Raina *et al.*, 2005). They utilize particular graph-matching procedures to perform inferences. Since our logical formulae essentially restate the information in the dependency graph, our abductive inference and learning algorithms are not tied to the logical representation; in particular, the inference algorithm can be modified to work with these graph-based representations, where it can be interpreted as a "graph-matching" procedure that prefers globally "consistent" matchings.

Table 4 shows that certain classes require more effort in linguistic modeling, and improvements in those classes can lead to great overall gains in performance. The current representation fails to capture some important interactions in its dependencies (e.g., the implication in *"If it rains, then there will be a rainbow."*). Several language resources have sparse knowledge (e.g., antonyms in WordNet); for effective semantic reasoning, it is desirable to have broader coverage resources that can represent fine distinctions and similarities in word meanings.

Acknowledgments

We give warm thanks to Aria Haghighi, Christopher Cox, Jenny Finkel, Jeff Michels, Rion Snow, Kristina Toutanova, Bill MacCartney and Marie-Catherine de Marneffe for helpful discussions on the PASCAL dataset, and for providing many of the linguistic modules used in the experiments. This work was supported by the ARDA AQUAINT program.

References

Collins, M. J. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. Dissertation, University of Pennsylvania.

Genesereth, M., and Nilsson, N. J. 1987. *Logical Foundations of Artificial Intelligence*. San Mateo, California: Morgan Kaufmann.

Harabagiu, S. M.; Pasca, M. A.; and Maiorano, S. J. 2000. Experiments with open-domain textual question answering. In *COLING*, 292–298.

Hobbs, J. R.; Stickel, M. E.; Appelt, D. E.; and Martin, P. 1993. Interpretation as abduction. *Artif. Intell.* 63(1-2):69–142.

Klein, D., and Manning, C. D. 2003. Accurate unlexicalized parsing. In *ACL*, 423–430.

Miller, G. A. 1995. Wordnet: A lexical database for English. *Commun. ACM* 38(11):39–41.

Moldovan, D. I.; Clark, C.; Harabagiu, S. M.; and Maiorano, S. J. 2003. Cogex: A logic prover for question answering. In *HLT-NAACL*.

PASCAL Recognizing Textual Entailment Challenge, 2005. http://www.pascal-network.org/challenges/rte/.

Pedersen, T.; Patwardhan, S.; and Michelizzi, J. 2004. Wordnet: : Similarity - measuring the relatedness of concepts. In *AAAI*, 1024–1025.

Punyakanok, V.; Roth, D.; and Yih, W. 2004. Natural language inference via dependency tree mapping: An application to question answering. *Computational Linguistics*. In submission.

Raina, R.; Haghighi, A.; Cox, C.; Finkel, J.; Michels, J.; Toutanova, K.; MacCartney, B.; de Marneffe, M.-C.; Manning, C. D.; and Ng, A. Y. 2005. Robust textual inference using diverse knowledge sources. In *PASCAL Challenges Workshop*.

Resnik, P. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, 448–453.

Rockafellar, R. T. 1972. *Convex Analysis*. Princeton University press.