Learning Alignments and Leveraging Natural Logic

Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon
Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage
Eric Yeh, Christopher D. Manning
Computer Science Department
Stanford University
Stanford, CA 94305
{natec,dcer,grenager,dlwh,loeki,wcmac,mcdm,dramage,yeh1,manning}@stanford.edu

Abstract

We describe an approach to textual inference that improves alignments at both the typed dependency level and at a deeper semantic level. We present a machine learning approach to alignment scoring, a stochastic search procedure, and a new tool that finds deeper semantic alignments, allowing rapid development of semantic features over the aligned graphs. Further, we describe a complementary semantic component based on natural logic, which shows an added gain of 3.13% accuracy on the RTE3 test set.

1 Introduction

Among the many approaches to textual inference, alignment of dependency graphs has shown utility in determining entailment without the use of deep understanding. However, discovering alignments requires a scoring function that accurately scores alignment and a search procedure capable of approximating the optimal mapping within a large search space. We address the former requirement through a machine learning approach for acquiring lexical feature weights, and we address the latter with an approximate stochastic approach to search.

Unfortunately, the most accurate aligner cannot capture deeper semantic relations between two pieces of text. For this, we have developed a tool, Semgrex, that allows the rapid development of dependency rules to find specific entailments, such as familial or locative relations, a common occurence in textual entailment data. Instead of writing code by hand to capture patterns in the dependency graphs, we develop a separate rule-base that operates over aligned dependency graphs. Further, we describe a separate natural logic component that complements our textual inference system, making local entailment decisions based on monotonic assumptions.

The next section gives a brief overview of the system architecture, followed by our proposal for improving alignment scoring and search. New coreference features and the Semgrex tool are then described, followed by a description of natural logic.

2 System Overview

Our system is a three stage architecture that conducts linguistic analysis, builds an alignment between dependency graphs of the text and hypothesis, and performs inference to determine entailment.

Linguistic analysis identifies semantic entities, relationships, and structure within the given text and hypothesis. Typed dependency graphs are passed to the aligner, as well as lexical features such as named entities, synonymity, part of speech, etc. The alignment stage then performs dependency graph alignment between the hypothesis and text graphs, searching the space of possible alignments for the highest scoring alignment. Improvements to the scorer, search algorithm, and automatically learned weights are described in the next section.

The final inference stage determines if the hypothesis is entailed by the text. We construct a set of features from the previous stages ranging from antonyms and polarity to graph structure and semantic relations. Each feature is weighted according to a set of hand-crafted or machine-learned weights over

Proceedings of the Workshop on Textual Entailment and Paraphrasing, pages 165–170,
Prague, June 2007.

the development dataset. We do not describe the features here; the reader is referred to de Marneffe et al. (2006a) for more details. A novel component that leverages natural logic is also used to make the final entailment decisions, described in section 6.

3 Alignment Model

We examine three tasks undertaken to improve the alignment phase: (1) the construction of manually aligned data which enables automatic learning of alignment models, and effectively decouples the alignment and inference development efforts, (2) the development of new search procedures for finding high-quality alignments, and (3) the use of machine learning techniques to automatically learn the parameters of alignment scoring models.

3.1 Manual Alignment Annotation

While work such as Raina et al. (2005) has tried to learn feature alignment weights by credit assignment backward from whether an item is answered correctly, this can be very difficult, and here we follow Hickl et al. (2006) in using supervised gold-standard alignments, which help us to evaluate and improve alignment and inference independently.

We built a web-based tool that allows annotators to mark semantic relationships between text and hypothesis words. A table with the hypothesis words on one axis and the text on the other allows relationships to be marked in the corresponding table cell with one of four options. These relationships include text to hypothesis entailment, hypothesis to text entailment, synonymy, and antonymy. Examples of entailment (from the RTE 2005 dataset) include pairs such as *drinking/consumption*, *coronavirus/virus*, and *Royal Navy/British*. By distinguishing between these different types of alignments, we can capture some limited semantics in the alignment process, but full exploitation of this information is left to future work.

We annotated the complete RTE2_dev and RTE3_dev datasets, for a total of 1600 aligned text/hypothesis pairs (the data is available at http://nlp.stanford.edu/projects/rte/).

3.2 Improving Alignment Search

In order to find "good" alignments, we define both a formal model for scoring the quality of a proposed alignment and a search procedure over the alignment space. Our goal is to build a model that maximizes the total alignment score of the full dataset $\mathcal{D}$, which we take to be the sum of the alignment scores for all individual text/hypothesis pairs (t, h).

Each of the text and hypothesis is a semantic dependency graph; $n(h)$ is the set of nodes (words) and $e(h)$ is the set of edges (grammatical relations) in a hypothesis h. An alignment $a\colon n(h) \mapsto n(t) \cup \{null\}$ maps each hypothesis word to a text word or to a *null* symbol, much like an IBM-style machine translation model. We assume that the alignment score $s(t, h, a)$ is the sum of two terms, the first scoring aligned word pairs and the second the match between an edge between two words in the hypothesis graph and the corresponding path between the words in the text graph. Each of these is a sum, over the scoring function for individual word pairs s_w and the scoring function for edge path pairs s_e:

$$\begin{aligned} s(t,h,a) = & \sum_{h_i \in n(h)} s_w(h_i, a(h_i)) \\ & + \sum_{(h_i,h_j) \in e(h)} s_e((h_i,h_j),(a(h_i),a(h_j))) \end{aligned}$$

The space of alignments for a hypothesis with m words and a text with n words contains $(n+1)^m$ possible alignments, making exhaustive search intractable. However, since the bulk of the alignment score depends on local factors, we have explored several search strategies and found that *stochastic local search* produces better quality solutions.

Stochastic search is inspired by Gibbs sampling and operates on a complete state formulation of the search problem. We initialize the algorithm with the complete alignment that maximizes the greedy word pair scores. Then, in each step of the search, we seek to randomly replace an alignment for a single hypothesis word h_i. For each possible text word t_j (including *null*), we compute the alignment score if we were to align h_i with t_j. Treating these scores as log probabilities, we create a normalized distribution from which we sample one alignment. This Gibbs sampler is guaranteed to give us samples from the posterior distribution over alignments defined implicitly by the scoring function. As we wish to find a maximum of the function, we use simulated annealing by including a temperature parameter to smooth the sampling distribution as a function of time. This allows us to initially explore the space widely, but later to converge to a local maximum which is hopefully the global maximum.

3.3 Learning Alignment Models

Last year, we manually defined the alignment scoring function (de Marneffe et al., 2006a). However, the existence of the gold standard alignments described in section 3.1 enables the automatic learning of a scoring function. For both the word and edge scorers, we choose a linear model where the score is the dot product of a feature and a weight vector:

$$s_w(h_i, t_j) = \theta_w \cdot \mathbf{f}(h_i, t_j), \text{ and}$$
$$s_e((h_i, h_j), (t_k, t_\ell)) = \theta_e \cdot \mathbf{f}((h_i, h_j), (t_k, t_\ell)).$$

Recent results in machine learning show the effectiveness of online learning algorithms for structure prediction tasks. Online algorithms update their model at each iteration step over the training set. For each datum, they use the current weight vector to make a prediction which is compared to the correct label. The weight vector is updated as a function of the difference. We compared two different update rules: the *perceptron update* and the *MIRA update*. In the perceptron update, for an incorrect prediction, the weight vector is modified by adding a multiple of the difference between the feature vector of the correct label and the feature vector of the predicted label. We use the adaptation of this algorithm to structure prediction, first proposed by (Collins, 2002). The *MIRA update* is a proposed improvement that attempts to make the *minimal* modification to the weight vector such that the score of the incorrect prediction for the example is lower than the score of the correct label (Crammer and Singer, 2001).

We compare the performance of the perceptron and MIRA algorithms on 10-fold cross-validation on the RTE2_dev dataset. Both algorithms improve with each pass over the dataset. Most improvement is within the first five passes. Table 1 shows runs for both algorithms over 10 passes through the dataset. MIRA consistently outperforms perceptron learning. Moreover, scoring alignments based on the learned weights marginally outperforms our hand-constructed scoring function by 1.7% absolute.

A puzzling problem is that our overall performance decreased 0.87% with the addition of RTE3_dev alignment data. We believe this is due to a larger proportion of "irrelevant" and "relation" pairs. Irrelevant pairs are those where the text and hypothesis are completely unrelated. Relation pairs are those where the correct entailment judgment relies on the extraction of relations such as *X* works for *Y*, *X* is located in *Y*, or *X* is the wife of *Y*. Both of these categories do not rely on alignments for entailment decisions, and hence introduce noise.

	Perfectly aligned	
	Individual words	Text/hypothesis pairs
Perceptron	4675	271
MIRA	4775	283

Table 1: Perceptron and MIRA results on 10-fold cross-validation on RTE2_dev for 10 passes.

4 Coreference

In RTE3, 135 pairs in RTE3_dev and 117 in RTE3_test have lengths classified as "long," with 642 personal pronouns identified in RTE3_dev and 504 in RTE3_test. These numbers suggest that resolving pronomial anaphora plays an important role in making good entailment decisions. For example, identifying the first "he" as referring to "Yunus" in this pair from RTE3_dev can help alignment and other system features.

P: Yunus, who shared the 1.4 million prize Friday with the Grameen Bank that he founded 30 years ago, pioneered the concept of "microcredit."
H: Yunus founded the Grameen Bank 30 years ago.

Indeed, 52 of the first 200 pairs from RTE3_dev were deemed by a human evaluator to rely on reference information. We used the OpenNLP[1] package's maximum-entropy coreference utility to perform resolution on parse trees and named-entity data from our system. Found relations are stored and used by the alignment stage for word similarity.

We evaluated our system with and without coreference over RTE3_dev and RTE3_test. Results are shown in Table 3. The presence of reference information helped, approaching significance on the development set ($p < 0.1$, McNemar's test, 2-tailed), but not on the test set. Examination of alignments and features between the two runs shows that the alignments do not differ significantly, but associated

[1] http://opennlp.sourceforge.net/

weights do, thus affecting entailment threshold tuning. We believe coreference needs to be integrated into all the featurizers and lexical resources, rather than only with word matching, in order to make further gains.

5 Semgrex Language

A core part of an entailment system is the ability to find semantically equivalent patterns in text. Previously, we wrote tedious graph traversal code by hand for each desired pattern. As a remedy, we wrote Semgrex, a pattern language for dependency graphs. We use Semgrex atop the typed dependencies from the Stanford Parser (de Marneffe et al., 2006b), as aligned in the alignment phase, to identify both semantic patterns in a single text and over two aligned pieces of text. The syntax of the language was modeled after tgrep/Tregex, query languages used to find syntactic patterns in trees (Levy and Andrew, 2006). This speeds up the process of graph search and reduces errors that occur in complicated traversal code.

5.1 Semgrex Features

Rather than providing regular expression matching of atomic tree labels, as in most tree pattern languages, Semgrex represents nodes as a (non-recursive) attribute-value matrix. It then uses regular expressions for subsets of attribute values. For example, `{word:run;tag:/^NN/}` refers to any node that has a value *run* for the attribute *word* and a *tag* that starts with *NN*, while {} refers to any node in the graph.

However, the most important part of Semgrex is that it allows you to specify relations between nodes. For example, `{} <nsubj {}` finds all the dependents of `nsubj` relations. Logical connectives can be used to form more complex patterns and node naming can help retrieve matched nodes from the patterns. Four base relations, shown in figure 1, allow you to specify the type of relation between two nodes, in addition to an alignment relation (@) between two graphs.

5.2 Entailment Patterns

A particularly useful application of Semgrex is to create relation entailment patterns. In particular, the IE subtask of RTE has many characteristics that are not well suited to the core alignment features of our system. We began integrating Semgrex into our system by creating semantic alignment rules for these IE tasks.

Semgrex Relations

Symbol	#Description
{A} >reln {B}	A is the governor of a reln relation with B
{A} <reln {B}	A is the dependent of a reln relation with B
{A} >>reln {B}	A dominates a node that is the governor of a reln relation with B
{A} <<reln {B}	A is the dependent of a node that is dominated by B
{A} @ {B}	A aligns to B

Figure 1: Semgrex relations between nodes.

T: Bill Clinton's wife Hillary was in Wichita today, continuing her campaign.
H: Bill Clinton is married to Hillary. (*TRUE*)
Pattern:

```
({}=1
  <nsubjpass ({word:married} >pp_to {}=2))
@ ({} >poss ({lemma:/wife/} >appos {}=3))
```

This is a simplified version of a pattern that looks for marriage relations. If it matches, additional programmatic checks ensure that the nodes labeled 2 and 3 are either aligned or coreferent. If they are, then we add a MATCH feature, otherwise we add a MISMATCH. Patterns included other familial relations and employer-employee relations. These patterns serve both as a necessary component of an IE entailment system and as a test drive of Semgrex.

5.3 Range of Application

Our rules for marriage relations correctly matched six examples in the RTE3 development set and one in the test set. Due to our system's weaker performance on the IE subtask of the data, we analyzed 200 examples in the development set for Semgrex applicability. We identified several relational classes, including the following:

- **Work**: works for, holds the position of
- **Location**: lives in, is located in
- **Relative**: wife/husband of, are relatives
- **Membership**: is an employee of, is part of
- **Business**: is a partner of, owns
- **Base**: is based in, headquarters in

These relations make up at least 7% of the data, suggesting utility from capturing other relations.

6 Natural Logic

We developed a computational model of *natural logic*, the NatLog system, as another inference engine for our RTE system. NatLog complements our core broad-coverage system by trading lower recall for higher precision, similar to (Bos and Markert, 2006). Natural logic avoids difficulties with translating natural language into first-order logic (FOL) by forgoing logical notation and model theory in favor of natural language. Proofs are expressed as incremental edits to natural language expressions. Edits represent conceptual *contractions* and *expansions*, with truth preservation specified natural logic. For further details, we refer the reader to (Sánchez Valencia, 1995).

We define an entailment relation $\sqsubseteq$ between nouns (*hammer* $\sqsubseteq$ *tool*), adjectives (*deafening* $\sqsubseteq$ *loud*), verbs (*sprint* $\sqsubseteq$ *run*), modifiers, connectives and quantifiers. In ordinary (*upward-monotone*) contexts, the entailment relation between compound expressions mirrors the entailment relations between their parts. Thus *tango in Paris* $\sqsubseteq$ *dance in France*, since *tango* $\sqsubseteq$ *dance* and *in Paris* $\sqsubseteq$ *in France*. However, many linguistic constructions create *downward-monotone* contexts, including negation (*didn't sing* $\sqsubseteq$ *didn't yodel*), restrictive quantifiers (*few beetles* $\sqsubseteq$ *few insects*) and many others.

NatLog uses a three-stage architecture, comprising linguistic pre-processing, alignment, and entailment classification. In pre-processing, we define a list of expressions that affect monotonicity, and define Tregex patterns that recognize each occurrence and its scope. This *monotonicity marking* can correctly account for multiple monotonicity inversions, as in *no soldier without a uniform*, and marks each token span with its final effective monotonicity.

In the second stage, word alignments from our RTE system are represented as a sequence of *atomic edits* over token spans, as entailment relations are described across incremental edits in NatLog. Aligned pairs generate *substitution* edits, unaligned premise words yield *deletion* edits, and unaligned hypothesis words yield *insertion* edits. Where possible, contiguous sequences of word-level edits are collected into span edits.

In the final stage, we use a decision-tree classifier to predict the *elementary entailment relation* (table 2) for each atomic edit. Edit features include the type, effective monotonicity at affected tokens, and their lexical features, including syntactic category, lemma similarity, and WordNet-derived measures of synonymy, hyponymy, and antonymy. The classifier was trained on a set of 69 problems designed to exercise the feature space, learning heuristics such as *deletion in an upward-monotone context yields* $\sqsubset$, *substitution of a hypernym in a downward-monotone context yields* $\sqsupset$, and *substitution of an antonym yields* $|$.

relation	symbol	in terms of $\sqsubseteq$	RTE
equivalent	$p = h$	$p \sqsubseteq h, h \sqsubseteq p$	yes
forward	$p \sqsubset h$	$p \sqsubseteq h, h \not\sqsubseteq p$	yes
reverse	$p \sqsupset h$	$h \sqsubseteq p, p \not\sqsubseteq h$	no
independent	$p \# h$	$p \not\sqsubseteq h, h \not\sqsubseteq p$	no
exclusive	$p \mid h$	$p \sqsubseteq \neg h, h \sqsubseteq \neg p$	no

Table 2: NatLog's five elementary entailment relations. The last column indicates correspondences to RTE answers.

To produce a top-level entailment judgment, the atomic entailment predictions associated with each edit are composed in a fairly obvious way. If r is any entailment relation, then $(= \circ r) \equiv r$, but $(\# \circ r) \equiv \#$. $\sqsubset$ and $\sqsupset$ are transitive, but $(\sqsubset \circ \sqsupset) \equiv \#$, and so on.

We do not expect NatLog to be a general-purpose solution for RTE problems. Many problems depend on types of inference that it does not address, such as paraphrase or relation extraction. Most pairs have large edit distances, and more atomic edits means a greater chance of errors propagating to the final output: given the entailment composition rules, the system can answer *yes* only if all atomic-level predictions are either $\sqsubset$ or $=$. Instead, we hope to make reliable predictions on a subset of the RTE problems.

Table 3 shows NatLog performance on RTE3. It makes positive predictions on few problems (18% on development set, 24% on test), but achieves good precision relative to our RTE system (76% and 68%, respectively). For comparison, the FOL-based system reported in (Bos and Markert, 2006) attained a precision of 76% on RTE2, but made a positive prediction in only 4% of cases. This high precision suggests that superior performance can be achieved by hybridizing NatLog with our core RTE system.

The reader is referred to (MacCartney and Man-

ID	Premise(s)	Hypothesis	Answer
518	The French railway company SNCF is cooperating in the project.	The French railway company is called SNCF.	*yes*
601	NUCOR has pioneered a giant mini-mill in which steel is poured into continuous casting machines.	Nucor has pioneered the first mini-mill.	*no*

Table 4: Illustrative examples from the RTE3 test suite

RTE3 Development Set (800 problems)				
System	% yes	precision	recall	accuracy
Core +coref	50.25	68.66	66.99	67.25
Core -coref	49.88	66.42	64.32	64.88
NatLog	18.00	**76.39**	26.70	58.00
Hybrid, bal.	50.00	69.75	67.72	68.25
Hybrid, opt.	55.13	69.16	**74.03**	**69.63**

RTE3 Test Set (800 problems)				
System	% yes	precision	recall	accuracy
Core +coref	50.00	61.75	60.24	60.50
Core -coref	50.00	60.25	58.78	59.00
NatLog	23.88	**68.06**	31.71	57.38
Hybrid, bal.	50.00	64.50	62.93	63.25
Hybrid, opt.	54.13	63.74	**67.32**	**63.62**

Table 3: Performance on the RTE3 development and test sets. *% yes* indicates the proportion of *yes* predictions made by the system. Precision and recall are shown for the *yes* label.

ning, 2007) for more details on NatLog.

7 System Results

Our core system makes *yes*/*no* predictions by thresholding a real-valued *inference score*. To construct a hybrid system, we adjust the inference score by $+x$ if NatLog predicts *yes*, $-x$ otherwise. x is chosen by optimizing development set accuracy when adjusting the threshold to generate balanced predictions (equal numbers of *yes* and *no*). As another experiment, we fix x at this value and adjust the threshold to optimize development set accuracy, resulting in an excess of *yes* predictions. Results for these two cases are shown in Table 3. Parameter values tuned on development data yielded the best performance. The optimized hybrid system attained an absolute accuracy gain of 3.12% over our RTE system, corresponding to an extra 25 problems answered correctly. This result is statistically significant ($p < 0.01$, McNemar's test, 2-tailed).

The gain cannot be fully attributed to NatLog's success in handling the kind of inferences about monotonicity which are the staple of natural logic. Indeed, such inferences are quite rare in the RTE data. Rather, NatLog seems to have gained primarily by being more precise. In some cases, this precision works against it: NatLog answers *no* to problem 518 (table 4) because it cannot account for the insertion of *called*. On the other hand, it correctly rejects the hypothesis in problem 601 because it cannot account for the insertion of *first*, whereas the less-precise core system was happy to allow it.

Acknowledgements

This material is based upon work supported in part by the Disruptive Technology Office (DTO)'s AQUAINT Phase III Program.

References

Johan Bos and Katja Markert. 2006. When logical inference helps determining textual entailment (and when it doesn't). In *Proceedings of the Second PASCAL RTE Challenge*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP-2002*.

Koby Crammer and Yoram Singer. 2001. Ultraconservative online algorithms for multiclass problems. In *Proceedings of COLT-2001*.

Marie-Catherine de Marneffe, Bill MacCartney, Trond Grenager, Daniel Cer, Anna Rafferty, and Christopher D. Manning. 2006a. Learning to distinguish valid textual entailments. In *Second Pascal RTE Challenge Workshop*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006b. Generating typed dependency parses from phrase structure parses. In *5th Int. Conference on Language Resources and Evaluation (LREC 2006)*.

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing textual entailment with LCC's GROUNDHOG system. In *Proceedings of the Second PASCAL RTE Challenge*.

Roger Levy and Galen Andrew. 2006. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *ACL Workshop on Textual Entailment and Paraphrasing*.

Rajat Raina, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *AAAI 2005*, pages 1099–1105.

Victor Sánchez Valencia. 1995. Parsing-driven inference: Natural logic. *Linguistic Analysis*, 25:258–285.