Natural Logic for Textual Inference

Bill MacCartney
Stanford University
wcmac@cs.stanford.edu

Christopher D. Manning
Stanford University
manning@cs.stanford.edu

Abstract

This paper presents the first use of a computational model of *natural logic*—a system of logical inference which operates over natural language—for textual inference. Most current approaches to the PASCAL RTE textual inference task achieve robustness by sacrificing semantic precision; while broadly effective, they are easily confounded by ubiquitous inferences involving monotonicity. At the other extreme, systems which rely on first-order logic and theorem proving are precise, but excessively brittle. This work aims at a middle way. Our system finds a low-cost edit sequence which transforms the premise into the hypothesis; learns to classify entailment relations across atomic edits; and composes atomic entailments into a top-level entailment judgment. We provide the first reported results for any system on the FraCaS test suite. We also evaluate on RTE3 data, and show that hybridizing an existing RTE system with our natural logic system yields significant performance gains.

1 Introduction

The last five years have seen a surge of interest in the problem of *textual inference*, that is, automatically determining whether a natural-language *hypothesis* can be inferred from a given *premise*. A broad spectrum of approaches have been explored, ranging from shallow-but-robust to deep-but-brittle. Up to now, the most successful approaches have used fairly impoverished semantic representations, relying on measures of lexical or semantic overlap (Jijkoun and de Rijke, 2005), pattern-based relation extraction (Romano et al., 2006), or approximate matching of predicate-argument structure (Hickl et al., 2006). Such methods, while robust and broadly effective, are imprecise, and are easily confounded by ubiquitous inferences involving monotonicity, particularly in negative polarity contexts, as in:

P: *No case of indigenously acquired rabies infection has been confirmed in the past 2 years.*
H: *No rabies cases have been confirmed.*

Because it drops important qualifiers in a negative context, the hypothesis does not follow; yet both the lexical content and the predicate-argument structure of the hypothesis closely match the premise.

At the other extreme, textual inference can be approached as deduction, building on work in formal computational semantics to translate sentences into first-order logic (FOL), and then applying a theorem prover or a model builder (Akhmatova, 2005; Fowler et al., 2005). However, such approaches tend to founder on the difficulty of accurately translating natural language in FOL—tricky issues include idioms, intensionality and propositional attitudes, modalities, temporal and causal relations, certain quantifiers, and so on. FOL-based systems that have attained high precision (Bos and Markert, 2006) have done so at the cost of very poor recall.

In this work, we explore a different point on the spectrum, by developing a computational model of *natural logic*, that is, a logic whose vehicle of inference is natural language.[1] Natural logic eschews logical notation and model theory. Its proofs proceed by incremental edits to expressions of natural language, and its inference rules specify conditions under which semantic expansions or contractions preserve truth. It thus permits us to do precise reasoning about monotonicity, while sidestepping the difficulties of translating sentences into FOL.

It should be emphasized that there are many

[1]Natural logic should not be confused with *natural deduction*, a proof system for first-order logic.

Proceedings of the Workshop on Textual Entailment and Paraphrasing, pages 193–200,
Prague, June 2007.

important kinds of inference which are not addressed by a natural logic system, including temporal reasoning, causal reasoning (*Khan sold nuclear plans* ⇒ *Khan possessed nuclear plans*), paraphrase (*McEwan flew to Rome* ⇒ *McEwan took a flight to Rome*), relation extraction (*Bill Gates and his wife, Melinda...* ⇒ *Melinda Gates is married to Bill Gates*), etc. Moreover, a natural logic system will struggle with inferences requiring model-building or deep proof search, which are more suitable for formal deduction systems. However, the applicability of natural logic is broader than it might at first appear, and a natural logic system can be designed to integrate with other kinds of reasoners.

2 Foundations of natural logic

Natural logic aims to explain inferences involving monotonicity, in which the concepts or constraints expressed are expanded or contracted. Consider, for example, the sentence *Every meal without wine is a terrible crime*. Some semantic elements can be expanded (but not contracted) *salva veritate*, and are therefore said to have positive polarity: *wine* may be broadened to *drink*, *terrible crime* may be relaxed to *crime*, or *every* may be weakened to *some*. Other elements can only be contracted (not expanded) *salva veritate*, and thus have negative polarity: *meal* can be narrowed to *dinner*. The monotonicity calculus developed in (Sánchez Valencia, 1991) explains these polarity effects by (1) defining an entailment relation over multifarious expressions of natural language, (2) defining monotonicity properties of semantic functions, and finally (3) specifying how monotonicities combine during Fregean composition of semantic functions.

The entailment relation. Most work in textual inference reflects a simple concept of entailment: one sentence entails another, or does not. In natural logic, however, entailment is a semantic containment relation (analogous to the set containment relation $\subseteq$) over expressions of all types, including words and phrases as well as sentences. We define the entailment relation $\sqsubseteq$ recursively over the semantic types familiar from Montague semantics. If c and d are of type t (truth values), then $c \sqsubseteq d$ iff $c \rightarrow d$. If c and d are of type e (entities), then $c \sqsubseteq d$ iff $c = d$. Finally, if c and d are of functional type $\langle \alpha, \beta \rangle$, then $c \sqsubseteq d$ iff for all $a \in \alpha$, $c(a) \sqsubseteq d(a)$. Otherwise, if $c \not\sqsubseteq d$ and $d \not\sqsubseteq c$, we write $c \# d$.

Using these formal definitions, we can establish entailment relations between common nouns (*penguin* $\sqsubseteq$ *bird*), common and proper adjectives (*tiny* $\sqsubseteq$ *small*, *French* $\sqsubseteq$ *European*), transitive and intransitive verbs (*kick* $\sqsubseteq$ *strike*, *hover* $\sqsubseteq$ *fly*), temporal and locative modifiers (*this morning* $\sqsubseteq$ *today*, *in Beijing* $\sqsubseteq$ *in China*), connectives (*and* $\sqsubseteq$ *or*), and quantifiers (*everyone* $\sqsubseteq$ *someone*, *all* $\sqsubseteq$ *most* $\sqsubseteq$ *some*).[2] Among noun phrases, we have *everyone* $\sqsubseteq$ *Einstein* $\sqsubseteq$ *some physicist*. Finally, observe that dropping a modifier generally yields entailment (*eat quickly* $\sqsubseteq$ *eat*) though this heuristic can be violated, e.g., by operator adjectives (*fake vaccine* $\not\sqsubseteq$ *vaccine*).

Monotonicity. Under the Fregean hypothesis, the meaning of a compound expression is the result of function application. In semantics as in mathematics, we can describe a function as *upward monotone* if "larger" inputs yield larger outputs. Formally, given a function f of functional type $\langle \alpha, \beta \rangle$:

- f is upward-monotone ($\uparrow$) iff for all $x, y \in \alpha$, $x \sqsubseteq y$ entails $f(x) \sqsubseteq f(y)$.
- f is downward-monotone ($\downarrow$) iff for all $x, y \in \alpha$, $x \sqsubseteq y$ entails $f(y) \sqsubseteq f(x)$.
- f is non-monotone ($\not\updownarrow$) iff it is neither upward- nor downward-monotone.

Most linguistic expressions may be regarded as upward-monotone semantic functions. Thus *tango in Paris* $\sqsubseteq$ *dance in France*, since *tango* $\sqsubseteq$ *dance* and *in Paris* $\sqsubseteq$ *in France*. However, a number of important linguistic constructions are downward-monotone, including negation (*not*), restrictive quantifiers (*no*, *few*, *at most* n), restrictive verbs (*lack*, *fail*, *prohibit*), certain adverbs (*without*, *except*), the antecedent of a conditional, and so on. We thus have *didn't dance* $\sqsubseteq$ *didn't tango*, *few athletes* $\sqsubseteq$ *few sprinters*, *lack weapons* $\sqsubseteq$ *lack guns*,

[2]The entailment relations among quantifiers may be counterintuitive to those prone to what Peter Geach called "quantificatious thinking", who might consider *someone* "smaller" than *everyone*. But in the theory of generalized quantifiers, the denotation of a quantified noun phrase is the set of predicates which it satisfies, and the predicates satisfied by *everyone* are a subset of those satisfied by *someone*. Note also that logicians will deny that the universal entails the existential: $\forall x\, P(x) \not\rightarrow \exists x\, P(x)$. However, most people are happy to infer *someone is hungry* from *everyone is hungry*.

without clothes $\sqsubseteq$ *without pants*, and *If stocks rise, we win* $\sqsubseteq$ *If stocks soar, we win*. Finally, a few expressions must be considered non-monotone, including superlative adjectives and quantifiers such as *most*. Thus *prettiest butterfly* # *prettiest insect* and *most boats* # *most vehicles*. Note that certain generalized quantifiers must be treated as binary functions having different monotonicities in different arguments. Thus *every* is downward-monotone in its first argument (*every fish swims* $\sqsubseteq$ *every shark swims*) but upward-monotone in its second argument (*every shark swims* $\sqsubseteq$ *every shark moves*).

Composition of monotonicity. Finally, we must specify how monotonicities combine during Fregean composition of semantic functions. In Sánchez Valencia's *marking algorithm*, we represent each input expression as a parse in the Lambek categorial grammar. We then (1) mark leaf nodes with appropriate lexical monotonicity values, (2) project monotonicities to internal nodes representing function applications, and finally (3) compose monotonicities along the path from the root to each leaf in order to determine effective polarities. The composition of monotonicities is straightforward. Suppose $h = f \circ g$. If either f or g is non-monotone, then so is h. Otherwise, if the monotonicities of f and g are the same, then h is upward-monotone; if they are different, then h is downward-monotone. (Thus, *wine* has positive polarity in *no meal without wine* because it falls under two downward-monotone operators.)

3 The NatLog System

Our natural logic system, dubbed the *NatLog* system, has a three-stage architecture similar to those in (Marsi and Krahmer, 2005; MacCartney et al., 2006), comprising (1) linguistic pre-preprocessing, (2) alignment, and (3) entailment classification.

3.1 Linguistic pre-processing

Relative to other textual inference systems, the NatLog system does comparatively little linguistic preprocessing. We rely on the Stanford parser (Klein and Manning, 2003), a Treebank-trained statistical parser, for tokenization, part-of-speech tagging, and phrase-structure parsing. By far the most important analysis performed at this stage is *monotonicity marking*, in which we compute the effective monotonicity for each token span in each input sentence. For this, we use an adaptation of the marking algorithm of Sánchez Valencia (section 2); however, our choice of a Treebank-trained parser (driven by the goal of broad coverage) requires us to modify the algorithm substantially. Unlike the categorial grammar parses assumed by Sánchez Valencia, the nesting of constituents in phrase-structure parses does not always correspond to the composition of semantic functions, which introduces a number of complications. We define a list of downward-monotone and non-monotone expressions, and for each item we specify its arity and a Tregex pattern (Levy and Andrew, 2006) which permits us to identify its occurrences. We also specify, for each argument, both the monotonicity and another Tregex pattern which helps us to determine the sentence span over which the monotonicity is projected. (Figure 1 shows some example definitions.) The marking process computes these projections, performs monotonicity composition where needed, and marks each token span with its final effective monotonicity.

unary operator: *without*
  pattern: `IN < /^[Ww]ithout\$/`
  argument 1: monotonicity $\downarrow$ on dominating PP
    pattern: `__ > PP=proj`

binary operator: *most*
  pattern: `JJS < /^[Mm]ost\$/ !> QP`
  argument 1: monotonicity $\not\updownarrow$ on dominating NP
    pattern: `__ >+(NP) (NP=proj !> NP)`
  argument 2: monotonicity $\uparrow$ on dominating S
    pattern: `__ >+(/.*/) (S=proj !> S)`

Figure 1: Two examples of monotonicity operator definitions. The patterns employ Tregex syntax.

3.2 Alignment

The second stage of processing establishes an *alignment* between the premise and the hypothesis. While there are many notions of alignment, in this work we have chosen to represent alignments as sequences of *atomic edits* over spans of word tokens. We define four types of atomic edits: *deletion* of a span from the premise, *insertion* of a span into the hypothesis, *substitution* of a hypothesis span for a premise span, and *advance* over a span without modification. Each atomic edit is parameterized by the token indices at which it operates. As an example, the first problem in table 3 may be aligned using following edits:

An Irishman	$\Longrightarrow$	An Irishman	`ADV`
won	$\Longrightarrow$	won	`ADV`
a	$\Longrightarrow$	the	`SUB`
Nobel prize	$\Longrightarrow$	Nobel prize	`ADV`
	$\Longrightarrow$	for literature	`INS`
.	$\Longrightarrow$	.	`ADV`

Clearly, this representation imposes certain limitations: there is no atomic edit type representing the movement of a token span from one sentence location to another (instead a combination of deletion and insertion must be used), and there can be no alignments to non-contiguous sets of tokens. However, the span edit representation also offers important benefits. First, there is always a well-defined sequence of intermediate forms through which the sentence progresses during editing, which is important for the computation of monotonicity features. Second, given a cost function over edits, it is possible to construct an efficient dynamic program to find the lowest-cost edit sequence between any pair of sentences, using a straightforward extension of the Levenshtein string-edit algorithm.

For this purpose, we have designed a cost function which prefers edits which operate on longer spans; penalizes edits operating on spans which are not parse-tree constituents; imposes nomimal cost on substitutions of words having the same lemma; and imposes little cost on certain "light" edits, involving prepositions, articles, auxiliaries, etc. When applied to problems like those in the FraCaS test suite (section 4), this cost model gives intuitively pleasing results. However, because our focus in this work is on entailment, we have not devoted much energy to optimizing our alignment model, and will not discuss it further. (For the RTE experiments described in section 5, we use alignments derived from an independent RTE system. Translating those alignments into the span edit representation requires relaxing some of its constraints, as we'll explain.)

3.3 Entailment classification

The edit sequence obtained during the alignment stage effectively decomposes the global entailment problem into a sequence of atomic entailment problems, one for each atomic edit. In the final stage, we train a model for atomic entailment classification, and predict an entailment relation for each atomic edit. We then compose our atomic entailment predictions to produce a global entailment prediction.

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

Table 1: The five elementary entailment relations. The last two columns indicate correspondences to FraCaS and RTE answers; see sections 4 and 5.

The atomic entailment model uses a classifier to predict one of five *elementary entailment relations* (table 1) for each atomic edit. This model uses a feature representation designed to capture characteristics of the edit pertinent to a natural logic analysis: the type of the edit (DEL, INS, or SUB), the effective monotonicity at the affected token span ($\uparrow$, $\downarrow$, or $\not\updownarrow$), and various lexical features of the affected tokens. In the case of a SUB edit, the lexical features help to indicate whether the substitution constitutes a semantic expansion, contraction, equivalence, or exclusion, using WordNet-derived measures of synonymy, hyponymy, and antonymy, and a measure of lemma similarity based on Levenshtein string-edit distance. In addition, for edits of all types, we have found it useful to generate a "light edit" feature indicating whether the affected tokens belong to categories which are usually negligible for inferential purposes, including prepositions, articles, auxiliaries, and punctuation.

The entailment model uses a decision tree classifier, trained on a small data set of 69 problems custom-designed to exercise diverse regions of the feature space.[3] From these examples, the decision tree effectively learns such heuristics 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* $\mid$.

To produce a top-level entailment judgment, the atomic entailment predictions associated with each

[3]Thus, in using learning, we are not trying to estimate statistical properties of some natural distribution of data. Rather, the learning framework provides (1) a modular way to add features which may impact the entailment decision, (2) a principled way to combine evidence from diverse features, such as real-valued lexical features, and (3) a convenient way to verify the proper functioning of the system.

atomic edit: SUB(*a*, *the*)
features:
type: SUB, *monotonicity*: $\uparrow$, *isLightEdit*: true,
wnSyno: 0.0, *wnHypo*: 0.0, *wnAnto*: 0.0, *lemmaSim*: 0.0
predicted entailment relation: $=$

atomic edit: INS(*for literature*)
features:
type: INS, *monotonicity*: $\uparrow$, *isLightEdit*: false
predicted entailment relation: $\sqsupset$

top-level inference:
composition of entailment relations: $= \circ \sqsupset \Rightarrow \sqsupset$
mapping to FraCaS answer: $\sqsupset \Rightarrow$ *unk*

Figure 2: The operation of the entailment model on FraCaS problem 33 (see table 3).

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. Compositions are commutative and associative.

Figure 2 shows an example of the operation of the entailment model.

4 Experiments with the FraCaS test suite

The FraCaS test suite (Cooper et al., 1996) was developed as part of a collaborative research effort in computational semantics. It contains 346 inference problems reminiscent of a textbook on formal semantics. In the authors' view, "inferencing tasks [are] the best way of testing an NLP system's semantic capacity." Yet, to our knowledge, this work is the first to present a quantitative system evaluation using FraCaS.[4]

The problems are divided into nine sections, each focused on a category of semantic phenomena, such as quantifiers or anaphora (see table 2). Each problem consists of one or more premise sentences, followed by a one-sentence question. For this project, the questions were converted into declarative hypotheses. Each problem also has an answer, which (usually) takes one of three values: *yes* (the hypothesis can be inferred from the premise(s)), *no* (the negation of the hypothesis can be inferred), or *unk* (neither the hypothesis nor its negation can be inferred). Some examples are shown in table 3.

[4]Indeed, our first step was to put the FraCaS data into machine-readable form, which we make publicly available at *http://nlp.stanford.edu/~wcmac/downloads/fracas.xml*.

§	Category	Count	% Acc.
1	Quantifiers	44	84.09
2	Plurals	24	41.67
3	Anaphora	6	50.00
4	Ellipsis	25	28.00
5	Adjectives	15	60.00
6	Comparatives	16	68.75
7	Temporal	36	61.11
8	Verbs	8	62.50
9	Attitudes	9	55.56
Applicable sections: 1, 5, 6		75	76.00
All sections		183	59.56

Table 2: NatLog's accuracy on the FraCaS test suite, by section. We exclude degenerate problems and multiple-premise problems; see text.

Not all of the 346 problems were used in this work. First, 12 of the problems were excluded because they are degenerate, lacking either a hypothesis or a well-defined answer. Second, an additional 151 problems (about 45% of the total) were excluded because they involve multiple premises. While many of the multiple-premise problems should be feasible for NatLog in the future, such inferences require search, and for now we have chosen to sidestep this complexity.

Finally, it should be noted that several sections of the test suite involve semantic phenomena, such as ellipsis, which the NatLog system makes no attempt to model. While we report results for these sections, we do not expect performance to be good, and in development we have concentrated on the sections where we expect NatLog to have relevant expertise. In table 2, results for these sections are aggregated under the label "applicable sections".

Results are shown in table 2. On the "applicable" sections, performance is good. (Not suprisingly, we make little headway with, e.g., ellipsis.) Of course, this does not constitute a proper evaluation on unseen test data—but on the other hand, the system was never trained on the FraCaS problems, and has had no opportunity to learn biases implicit in the data.[5] Our main goal in testing on FraCaS is to evaluate the representational and inferential adequacy of our model of natural logic, and from that perspective, the strong performance in quantifiers,

[5]This also explains why NatLog's performance on some FraCaS sections falls below that of a baseline most-common-label classifier.

§	ID	Premise(s)	Hypothesis	Ans
1	33	An Irishman won a Nobel prize.	An Irishman won the Nobel prize for literature.	*unk*
1	38	No delegate finished the report.	Some delegate finished the report on time.	*no*
2	99	Clients at the demonstration were all impressed by the system's performance. Smith was a client at the demonstration.	Smith was impressed by the system's performance.	*yes*
9	335	Smith believed that ITEL had won the contract in 1992.	ITEL won the contract in 1992.	*unk*

Table 3: Illustrative examples from the FraCaS test suite

	guess			
answer	*yes*	*unk*	*no*	total
yes	62	40	–	102
unk	15	45	–	60
no	6	13	2	21
total	90	91	2	183

Table 4: Confusions on FraCaS data (all sections)

adjectives, and comparatives is satisfying.

The confusion matrix shown in table 4 is instructive. By far the largest category of confusions comprise problems where we guess *unk* when the correct answer is *yes*. This reflects both the bias toward *yes* in the FraCaS data, and the system's tendency to predict *unk* (entailment relation $\#$) when confused: given the composition rules for entailment relations, the system can predict *yes* only if all atomic-level predictions are either $\sqsubset$ or $=$. On the other hand, there are a number of problems where we predict *yes* mistakenly. Several of these errors arise in a series of problems in §5 which concern operator adjectives such as *former*. The entailment model wrongly assumes that such modifiers, like any others, can safely be deleted in upward-monotone contexts, but in fact *former student* $\not\sqsubseteq$ *student*. If the feature set used by the entailment model were extended to represent occurrences of operator adjectives, and if appropriate examples were included in the training data, our accuracy in §5—and the average accuracy for the "applicable" sections—could easily be boosted over 80%.

5 Experiments with RTE data

Textual inference problems from the PASCAL RTE Challenge (Dagan et al., 2005) differ from FraCaS problems in several important ways. (See table 5 for examples.) Instead of textbook examples of semantic phenomena, RTE problems are more natural-seeming, with premises collected "in the wild" from newswire text. The premises are much longer, averaging 35 words (vs. 11 words for FraCaS). Also, the RTE task aims at a binary classification: the RTE *no* answer combines the *no* and *unk* answers in FraCaS.

Due to the character of RTE problems, we do not expect NatLog to be a good general-purpose solution to solving RTE problems. First, most RTE problems depend on forms of inference, such as paraphrase, temporal reasoning, or relation extraction, which NatLog is not designed to address. Second, in most RTE problems, the edit distance between premise and hypothesis is relatively large. More atomic edits means a greater chance that prediction errors made by the atomic entailment model will propagate, via entailment composition, to the system's final output. Rather, in applying NatLog to RTE, we hope to make reliable predictions on a subset of RTE problems, trading recall for precision. If we succeed, then we may be able to hybridize with a broad-coverage RTE system to obtain better results than either system individually—the same strategy that was adopted by (Bos and Markert, 2006) for their FOL-based system.

For this purpose, we have chosen to use the Stanford RTE system described in (de Marneffe et al., 2006). In applying NatLog to RTE problems, we use alignments from the Stanford system as input to our entailment model. A Stanford alignment is a map from hypothesis words to premise words. When we translate such alignments into the NatLog representation described in section 3, each pair of aligned words generates a *substitution* edit (or, if the words are identical, an *advance* edit). Unaligned premise words yield *deletion* edits, while unaligned hypothesis words yield *insertion* edits. Where possible, contiguous sequences of word-level edits are then collected into equivalent span edits. While the result of this translation method cannot be interpreted as a conventional edit script (there is no well-defined or-

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 5: Illustrative examples from the RTE3 test suite

RTE3 Development Set (800 problems)				
System	% yes	precision	recall	accuracy
Stanford	50.25	68.66	66.99	67.25
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
Stanford	50.00	61.75	60.24	60.50
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 6: 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.

dering of edits, and multiple edits can operate on the same input spans), we find that this poses no great impediment to subsequent processing by the entailment model.

Table 6 shows the performance of the NatLog system on RTE3 data. Relative to the Stanford RTE system, NatLog achieves high precision on its *yes* predictions—about 76% on the development set, and 68% on the test set—suggesting that hybridizing may be effective. For comparison, the FOL-based system reported in (Bos and Markert, 2006) attained a similarly high precision of 76% on RTE2 problems, but was able to make a positive prediction in only about 4% of cases. NatLog makes positive predictions far more often—at a rate of 18% on the development set, and 24% on the test set.

The Stanford RTE system makes *yes*/*no* predictions by thresholding a real-valued *inference score*. To construct a hybrid system, we adjust the Stanford inference scores by $+x$ or $-x$, depending on whether NatLog predicts *yes* or *no*/*unk*. We choose the value of x by optimizing development set accuracy, while adjusting the threshold to generate balanced predictions (that is, equal numbers of *yes* and *no* predictions). As an additional experiment, we fix x at this value and then adjust the threshold to optimize development set accuracy, resulting in an excess of *yes* predictions. (Since this optimization is based solely on development data, its use on test data is fully legitimate.) Results for these two cases are shown in table 6. The parameters tuned on development data were found to yield good performance on test data. The optimized hybrid system attained an absolute accuracy gain of 3.12% over the Stanford system, corresponding to an extra 25 problems answered correctly. This result is statistically significant ($p < 0.01$, McNemar's test, 2-tailed).

However, the gain cannot be 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 5) because it cannot account for the insertion of *called* in the hypothesis. 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 Stanford system was happy to allow it.

6 Related work

While the roots of natural logic can be traced back to Aristotle's syllogisms, the modern conception of natural logic began with George Lakoff, who proposed "a logic for natural language" which could "characterize all the valid inferences that can be made in natural language" (Lakoff, 1970). The study of natural logic was formalized by Johan van Benthem, who crucially connected it with categorial grammar (van Benthem, 1986), and later was brought to fruition by Victor Sánchez Valencia, who first gave a precise definition of a calculus of monotonicity (Sánchez Valencia, 1991). A small current of theoretical work has continued up to the present, for example (Zamansky et al., 2006).

There has been surprisingly little work on building computational models of natural logic. (Fyodorov et al., 2003) describes a Prolog implementation for a small fragment of English, based on a categorial grammar parser.[6] In an unpublished draft, (van Eijck, 2005) describes a preliminary implementation in Haskell.

Doing inference with representations close to natural language has also been advocated by Jerry Hobbs, as in (Hobbs, 1985).

To our knowledge, the FraCaS results reported here represent the first such evaluation. (Sukkarieh, 2003) describes applying a deductive system to some FraCaS inferences, but does not perform a complete evaluation or report quantitative results.

7 Conclusion

Our NatLog implementation of natural logic successfully handles a broad range of inferences involving monotonicity, as demonstrated on the FraCaS test suite. While a post-hoc analysis of performance on the RTE3 Challenge suggests that monotonicity-related inferences have limited applicability in RTE data, the greater precision of the NatLog system nevertheless significantly improved the performance of a hybrid RTE system. An area for future work is further consideration of what kinds of inference are prevalent and important in prominent computational linguistic applications.

Acknowledgements The authors wish to thank Marie-Catherine de Marneffe and the anonymous reviewers for their helpful comments on an earlier draft of this paper. This work was supported in part by ARDA's Advanced Question Answering for Intelligence (AQUAINT) Program.

References

Elena Akhmatova. 2005. Textual entailment resolution via atomic propositions. In *Proc. of the PASCAL RTE Challenge Workshop*.

Johan Bos and Katja Markert. 2006. When logical inference helps determining textual entailment (and when it doesn't). In *Proc. of the 2nd PASCAL RTE Challenge Workshop*.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, and Steve Pulman. 1996. Using the framework. Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. In *Proc. of the PASCAL RTE Challenge Workshop*.

Marie-Catherine de Marneffe, Bill MacCartney, Trond Grenager, Daniel Cer, Anna Rafferty, and Christopher D. Manning. 2006. Learning to distinguish valid textual entailments. In *Proc. of the 2nd PASCAL RTE Challenge Workshop*.

Abraham Fowler, Bob Hauser, Daniel Hodges, Ian Niles, Adrian Novischi, and Jens Stephan. 2005. Applying COGEX to recognize textual entailment. In *Proc. of the PASCAL RTE Challenge Workshop*.

Yaroslav Fyodorov, Yoad Winter, and Nissim Francez. 2003. Order-based inference in natural logic. *Logic Journal of the IGPL*, 11(4):385–416.

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing textual entailment with LCC's GROUNDHOG system. In *Proc. of the 2nd PASCAL RTE Challenge Workshop*.

Jerry R. Hobbs. 1985. Ontological promiscuity. In *Proc. of ACL-85*, pages 61–69.

Valentin Jijkoun and Maarten de Rijke. 2005. Recognizing textual entailment using lexical similarity. In *Proc. of the PASCAL RTE Challenge Workshop*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL-03*.

George Lakoff. 1970. Linguistics and natural logic. *Synthese*, 22:151–271.

Roger Levy and Galen Andrew. 2006. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *Proc. of LREC-06*.

Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proc. of HLT-NAACL-06*.

Erwin Marsi and Emiel Krahmer. 2005. Classification of semantic relations by humans and machines. In *Proc. of the ACL 2005 Workshop on Empirical Modeling of Semantic Equivalence and Entailment*.

Lorenza Romano, Milen Kouylekov, Idan Szpektor, Ido Dagan, and Alberto Lavelli. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *Proc. of EACL-06*.

Victor Sánchez Valencia. 1991. *Studies on Natural Logic and Categorial Grammar*. Ph.D. thesis, Univ. of Amsterdam.

Jana Z. Sukkarieh. 2003. An expressive efficient representation: Bridging a gap between NLP and KR. In *Proc. of the 7th Int'l Conf. on Knowledge-Based Intelligent Information and Engineering Systems*.

Johan van Benthem. 1986. *Essays in logical semantics*. Reidel, Dordrecht.

Jan van Eijck. 2005. Natural logic for natural language. *http://homepages.cwi.nl/~jve/papers/05/nlnl/NLNL.pdf*.

Anna Zamansky, Nissim Francez, and Yoad Winter. 2006. A 'natural logic' inference system using the Lambek calculus. *Journal of Logic, Language and Information*, 15:273–295.

[6] Available at *http://yeda.cs.technion.ac.il/~yaroslav/oc/*