An Automatic Monotonicity Annotation Tool Based on CCG Trees

Hai Hu and Larry Moss {huhai,lmoss}@indiana.edu

Introduction Despite the importance of monotonicity in logic and linguistic semantics, there have been few attempts to automatically annotate monotonicity information in natural language. In this paper, we present a tool that automatically annotates monotonicity information (i.e. polarity) at the word and constituent level at the same time, based on parse trees in Combinatory Categorial Grammar (CCG) [4]. Then we compare our tool's performance with another tool in the literature, NatLog [2], on a small hand-crafted dataset involving various monotonicity phenomena. We highlight the issues and difficulties in automatic monotonicity tagging.

Our tool In essence, our tool uses an extension of the algorithm described in [5] which only deals with the Ajdukiewicz/Bar-Hillel (AB) variant of Categorial Grammar (i.e. application rules (>) and (<)). That is, our tool can handle not only application rules, but type-raising and composition in CCG [4] (the T and B rules in Figure 1).

$$\frac{(x \xrightarrow{m} y)^{d} \quad x^{md}}{y^{d}}\,{>} \qquad \frac{(x \xrightarrow{m} y)^{d} \quad (y \xrightarrow{n} z)^{md}}{(x \xrightarrow{mn} z)^{d}}\,\textsc{b} \qquad \frac{x^{md}}{((x \xrightarrow{m} y) \xrightarrow{+} y)^{d}}\,\textsc{t}$$

Figure 1: This figure contains core rules of the markings and polarities. The letters m and n stand for one of the markings $+$, $-$, or $\cdot$; d stands for $\uparrow$ or $\downarrow$ (but not $=$). Some rules are omitted to save space. See the following charts for the operations $m, d \mapsto md$ and $m, n \mapsto mn$.

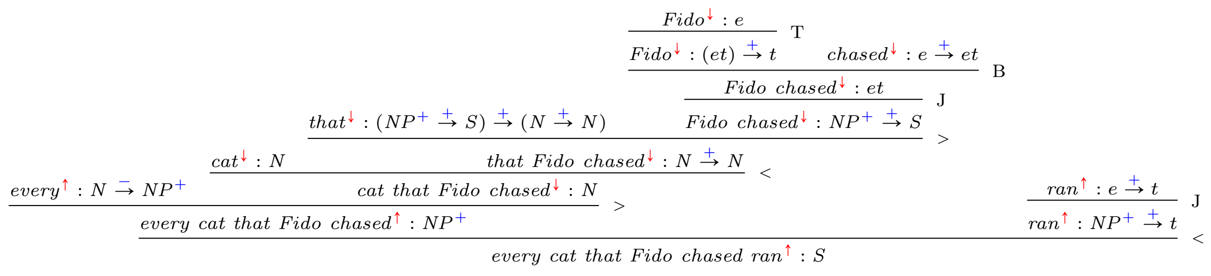
In simple terms, we first obtain a CCG parse tree of the input sentence by calling existing CCG parsers. Then our tool works in two steps: 1) `mark`, which adds *markings* ($\xrightarrow{+}$, $\xrightarrow{-}$ and $\xrightarrow{\cdot}$) for each node in the tree from leaf to root, and 2) `polarize`, which populates the *polarities* ($\uparrow$, $\downarrow$ and $=$) from root to leaf. The rules for the two steps are summarized in the following charts. The chart on the left is for combining two markings m and n, and the one on the right is for combining a marking m and a polarity d, obtaining a new polarity.

m \ n	$+$	$-$	$\cdot$
$+$	$+$	$-$	$\cdot$
$-$	$-$	$+$	$\cdot$
$\cdot$	$\cdot$	$\cdot$	$\cdot$

d \ m	$+$	$-$	$\cdot$
$\uparrow$	$\uparrow$	$\downarrow$	$=$
$\downarrow$	$\downarrow$	$\uparrow$	$=$

$flip\uparrow\, = \downarrow \quad flip\downarrow\, = \uparrow$

It is worth noting that our lexicon is manually coded with monotonicity information for quantifiers and other downward-entailing operators, e.g. *every*: $N \xrightarrow{-} NP^{+}$; *some*: $N \xrightarrow{+} NP^{+}$; *no*: $N \xrightarrow{-} NP^{-}$; *most*: $N \xrightarrow{\cdot} NP^{+}$; *without*: $NP \xrightarrow{-} ((S \xrightarrow{+} NP) \xrightarrow{+} (S \xrightarrow{+} NP))$; *refuse*: $(S \xrightarrow{+} NP) \xrightarrow{-} (S \xrightarrow{+} NP)$. Because of space limitation, we will not give formal definition of the *markings*, but informally, a $-$ sign means the monotonicity/polarity of its argument will be flipped. An example of the two steps in our algorithm is illustrated in the following tree *every cat that Fido chased ran.*


$Fido^{\downarrow} : e$
T
$Fido^{\downarrow} : (et) \xrightarrow{+} t$
$chased^{\downarrow} : e \xrightarrow{+} et$
B
$Fido\ chased^{\downarrow} : et$
J
$that^{\downarrow} : (NP^{+} \xrightarrow{+} S) \xrightarrow{+} (N \xrightarrow{+} N)$
$Fido\ chased^{\downarrow} : NP^{+} \xrightarrow{+} S$
>
$cat^{\downarrow} : N$
$that\ Fido\ chased^{\downarrow} : N \xrightarrow{+} N$
<
$every^{\uparrow} : N \xrightarrow{-} NP^{+}$
$cat\ that\ Fido\ chased^{\downarrow} : N$
>
$ran^{\uparrow} : e \xrightarrow{+} t$
J
$every\ cat\ that\ Fido\ chased^{\uparrow} : NP^{+}$
$ran^{\uparrow} : NP^{+} \xrightarrow{+} t$
<
$every\ cat\ that\ Fido\ chased\ ran^{\uparrow} : S$

One advantage of our system is that we obtain monotonicity at the constituent level as well, e.g., *every cat that Fido chased* receives $\uparrow$ while *cat that Fido chased* has $\downarrow$. This is not possible in NatLog [2].

Evaluation We hand-crafted 55 sentences containing a wide range of quantifiers, mixed with conditionals and conjunctions (see below) to test the system's polarization ability. We manually annotate the monotonicity labels in each sentence and then test our system, as well as the NatLog system in [2][1].

Sentence	Linguistic phenomenon
Some$^{\uparrow}$ rat$^{\uparrow}$ sees$^{\uparrow}$ every$^{\uparrow}$ squirrel$^{\downarrow}$	some/every
Most$^{\uparrow}$ dogs$^{=}$ chase$^{\uparrow}$ some$^{\uparrow}$ cat$^{\uparrow}$	most/some
Many$^{\uparrow}$ people$^{=}$ like$^{\uparrow}$ dogs$^{\uparrow}$ as$^{\uparrow}$ pets$^{\uparrow}$	many
At$^{\uparrow}$ least$^{\uparrow}$ seven$^{\downarrow}$ fish$^{\uparrow}$ died$^{\uparrow}$ yesterday$^{\uparrow}$ in$^{\uparrow}$ Morocco$^{\uparrow}$	at least n
My$^{\uparrow}$ parents$^{\uparrow}$ said$^{\uparrow}$ I$^{\uparrow}$ could$^{\uparrow}$ have$^{\uparrow}$ three$^{\downarrow}$ candies$^{\downarrow}$	numbers
Three$^{=}$ out$^{\downarrow}$ of$^{\downarrow}$ five$^{=}$ dentists$^{=}$ recommend$^{\uparrow}$ that$^{\uparrow}$ their$^{\downarrow}$ patients$^{\downarrow}$ brush$^{\uparrow}$ their$^{\downarrow}$ teeth$^{\downarrow}$ at$^{\uparrow}$ least$^{\uparrow}$ four$^{\downarrow}$ times$^{\uparrow}$ a$^{\uparrow}$ day$^{\uparrow}$	numbers
If$^{\uparrow}$ every$^{\downarrow}$ cat$^{\uparrow}$ runs$^{\downarrow}$,$^{\uparrow}$ then$^{\uparrow}$ some$^{\uparrow}$ dog$^{\uparrow}$ runs$^{\uparrow}$ also$^{\uparrow}$	conditional
A$^{\uparrow}$ dog$^{\uparrow}$ who$^{\uparrow}$ ate$^{\uparrow}$ two$^{\downarrow}$ rotten$^{\downarrow}$ biscuits$^{\downarrow}$ was$^{\uparrow}$ sick$^{\uparrow}$ for$^{\uparrow}$ three$^{\downarrow}$ days$^{\downarrow}$	relative clause/numbers
Ursula$^{\uparrow}$ refused$^{\uparrow}$ to$^{\uparrow}$ sing$^{\downarrow}$ or$^{\downarrow}$ dance$^{\downarrow}$	disjunction

Table 1: Example sentences in our evaluation dataset, with hand-annotated monotonicity information.

First, out of the 55 sentences, the CCG parser [1] gave wrong parse trees for 11 of them. The NatLog system depends on dependency parses given by the Stanford dependency parser, which produces 9 wrong parses. Since both systems make use of parse trees, both are in general unlikely to get a correct polarity annotation if the parse is problematic. Next, we evaluate the accuracy of polarity annotation on both the token level and the sentence level, same as how part-of-speech tagging evaluation is done [3]. We performed one evaluation on all tokens, another only on content words (plus determiners such as *some* and *every* and numbers, which are important for making inferences); the rationale is that it is hard to say what correct polarity we should give to the function word *as* in *Many*$^{\uparrow}$ *people*$^{=}$ *like*$^{\uparrow}$ *dogs*$^{\uparrow}$ *as*$^{\uparrow}$ *pets*$^{\uparrow}$. Most of the useful polarity information for inference is on content words.

	token-level			sentence-level		
system	majority	NatLog	ours	majority	NatLog	ours
accuracy (all tokens)	51.0	70.8	76.0	5.4	28.0	44.6
accuracy (content words + determiners + numbers)	49.1	69.4	78.2	5.4	28.6	50.0

Clearly, both systems are doing much better than the majority baseline which assigns $\uparrow$ to every token. Now the mistakes of the two systems. NatLog seems to consider "many" and "most" to be non-monotonic on both the first and second argument; it also treats "the" as "some" in that both of its arguments receive $\uparrow$ polarity, which is problematic. Negation is sometimes handled wrongly in NatLog where the NP in "NP doesn't VP" is tagged $\downarrow$. Both systems are unable to disambiguate the universal and existential "any", with NatLog tagging all "any" as a universal "any" and our system tagging it as the existential "any". Our system also has a hard time recognizing multi-word expressions such as "except for". While we can correctly tag conditionals and complements of verbal downward entailing operators (e.g., "refuse") as $\downarrow$, NatLog seems incapable in such cases. Overall, our system outperforms NatLog on this small evaluation dataset.

We intentionally put some hard examples involving numbers into the evaluation set, e.g., the second to last example in Table 1. The difficulty lies in determining whether the numbers are to be interpreted as *at least n*, *at most n* or *exactly n*. The correct polarities often cannot be determined without an understanding of the context and some world knowledge. This is what future work needs to address.

References

[1] M. Lewis and M. Steedman. A* CCG parsing with a supertag-factored model. In *Proceedings of EMNLP*, pages 990–1000, 2014.

[2] B. MacCartney. *Natural Language Inference.* PhD thesis, Stanford University, 2009.

[3] C. D. Manning. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *International conference on intelligent text processing and computational linguistics*, pages 171–189. Springer, 2011.

[4] M. Steedman. *The Syntactic Process.* The MIT Press, Cambridge, MA, 2000. ISBN 0-262-19420-1.

[5] J. van Benthem. *Essays in Logical Semantics.* Reidel, Dordrecht, 1986.

[1]We use the `natlog` annotator in Stanford CoreNLP 3.9.2: https://stanfordnlp.github.io/CoreNLP/natlog.html