Learning Transformation Rules for Semantic Role Labeling

Ken Williams **Christopher Dozier** **Andrew McCulloh**
Research & Development
Thomson Legal and Regulatory
Eagan, MN 55123 USA
{ken.williams|chris.dozier|andrew.mcculloh}@thomson.com

Abstract

This paper presents our work on Semantic Role Labeling using a Transformation-Based Error-Driven approach in the style of Eric Brill (Brill, 1995). Our approach achieved an overall F_1 score of 43.48 on non-verb annotations. We believe our approach is noteworthy because of its novelty in this area and because it produces short lists of human-understandable transformation rules as its output.

1 Introduction to Transformation-Based Error-Driven Learning

For the 2004 Conference on Computational Natural Language Learning (CoNLL), our team has applied the methodology popularized by Eric Brill for part-of-speech tagging and linguistic parsing (Brill, 1995; Brill, 1993). In this methodology, illustrated in Figure 1, a system learns a sequence of rules that best labels training data. These rules are then used to annotate previously unseen data.

According to (Brill, 1995), a Transformation-Based Error-Driven learning application is defined by:

1. The initial annotation scheme
2. The space of allowable transformations
3. The iterative algorithm for choosing a transformation sequence

The initial annotation may be extremely simple. For example, in a part-of-speech tagging task, the initial annotation may assign each token its most likely tag without any regard to context (Brill, 1995).

The iterative learning algorithm typically consists of simply searching for a rule that maximizes the increase in some objective function using a greedy hill-climbing strategy. For the CoNLL shared task, since participants are evaluated by their F_1 scores, it is reasonable to use the F_1 score as an objective function. We also implemented some extensions to the hill-climbing strategy that we describe in Section 2.3.

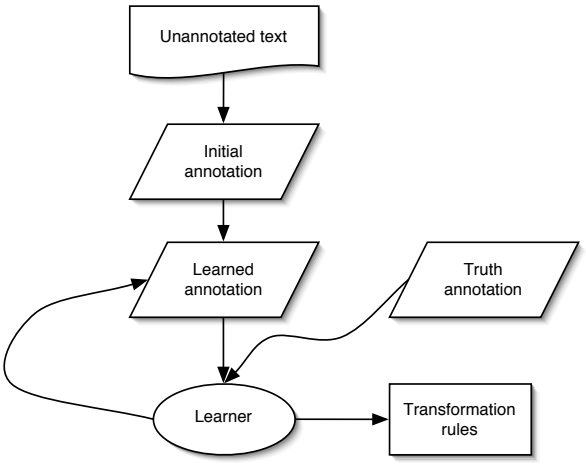

Figure 1: Overview of general Transformation-Based Error-Driven learning

2 Experimental Setting

In our approach, we used three successive learning stages–the first stage tags the verb region **V**, the second tags the **A0** and **A1** arguments, and the third tags all remaining arguments. The output of each stage becomes the initial annotation for the following stage. Therefore, our system only defines an explicit initial annotation for the verb-tagging phase: for each proposition, we initially tag only the single token containing the verb as **V**.

The search for new transformation templates is terminated when no new transformation can be found that would improve the objective function by at least 0.03%.

2.1 Transformation templates

For the first stage, transformations are generated from the following eight transformation templates:

Lengthen [shorten] the end of region[1] **V** by one token if:

a,b) followed by chunk with tag=X

c,d) followed by token with POS[2]=X

e,f) followed by chunk with tag=X and token with POS=Y

g,h) the verb token's lemma is X

In this formulation, "chunk" refers to the IOB2 chunks, and "clause" refers to the nested clause structure (**S** regions) given as task input. "Lemma" refers to the infinitive form of the verb, identified in the task input and coreferenced in the PropBank data. X and Y are variables that range over all types of chunks, POS tags, or lemmas. The rule-learning system must determine which values for these variables will produce the most effective transformations. For example, a rule that the system might produce from template 'e' is:

> *Lengthen the end of region **V** by one token if the region is followed by chunk with tag=**PRT** and token with POS=**RP**.*

Based on the observation that all **V** regions in the training data were either one or two tokens in length, an additional constraint was added to the first stage, requiring that lengthening-rules only apply to regions of length one, and shortening-rules only apply to regions of length two.

The second and third stages use a common set of eleven transformation templates, but in the second stage the learner is restricted to adding or altering only **A0** and **A1** regions. The transformation templates are as follows:

A,B) If chunk with tag=X is followed [preceded] directly by region with tag=Y, mark chunk as Z.

C,D) If token with POS=X is followed [preceded] directly by region with tag=Y, mark token as Z.

E,F) If chunk with tag=X is followed [preceded] (perhaps indirectly) by region with tag=Y, mark chunk as Z.

G,H) If region with tag=X is followed [preceded] by chunk with tag=**PP**, which is in turn followed [preceded] by chunk with tag=Y, extend X forward [backward] through Y.

I,J) If verb's first token has POS=X [and is preceded by POS=Y], switch **A0** and **A1**.

K) If region with tag=X is contained in a clause-starting verb phrase, and this is preceded by a clause-starting token with POS=Y, mark token Y as Z.

Templates "A-H" are meant to capture structural relationships among arguments, such as the fact that **A1** regions usually follow **V** regions, or that arguments may consist of several **NP** chunks joined by **PP** chunks. Templates "I" and "J" were written to discover passive verb relationships. Template "K" was an explicit (admittedly ad hoc) attempt to recognize **R-A0** and **R-A1** arguments.

To avoid creating tagged regions that overlap, we use a first-tag-wins strategy: if a transformation would tag a new region that overlaps an existing tagged region, the new region is trimmed until any overlaps vanish.

Notice that unlike the templates in the first stage, these templates make no reference to lexical information. In particular, no rule takes advantage of PropBank data in its tagging process[3]. We anticipate that using PropBank data would potentially improve performance, but we have not yet experimented with it. Also, without any lexical information in these templates, we are capturing only general patterns of argument structure within the training corpus, not the statistical patterns of particular verb frames. In future experiments we expect to incorporate lexical data into transformation rules.

2.2 Arguments and clause structure

In order to gain some traction on the problem, we analyzed the relationship between semantic arguments and clause boundaries. To investigate this, we labeled each argument with the smallest clause containing it as a proper subset. We then tallied the number of each type of argument labeled with the same clause as its verb, and the number labeled with a different clause. The results are shown in Table 1.

Note that for almost all argument types, the overwhelming majority of arguments are found in the same clause as the verb. This motivated us to add an additional constraint to the transformation templates A-J: only create arguments in the same clause as the verb. This simplification necessarily will miss any legitimate arguments outside the clause (most notably 20% of **A0** arguments).

2.3 Reordering of learned rules

In observing the sequence of transformations learned by the system, it became apparent that the system's strict

[1] In this paper, we use the term "region" to refer to a section of corpus text that has been labeled in the output as a verb or as a verb argument. We also use the term in rule definitions to refer to the type of label assigned to that section of text.

[2] part-of-speech

[3] We actually do use PropBank in a limited way: no transformation will assign an argument **A0**-**A5** to a verb unless that argument is listed for one of the verb's senses in PropBank.

Tag	Same clause	Different clause	Percent same
A1	16896	1150	93.6%
A0	10134	2575	79.7%
A2	4022	201	95.2%
AM-TMP	3329	238	93.3%
AM-MOD	1752	1	99.9%
AM-ADV	1675	52	97.0%
AM-MNR	1277	60	95.5%
AM-LOC	1184	95	92.6%
AM-DIS	1042	35	96.8%
A3	758	26	96.7%
AM-NEG	687	0	100.0%
A4	625	1	99.8%
AM-PNC	432	14	96.9%
C-A1	313	129	70.8%
AM-CAU	261	22	92.2%
AM-DIR	228	3	98.7%
AM-EXT	150	2	98.7%
R-A0	10	728	1.4%
R-A1	7	353	1.9%

Table 1: Verb-argument clause agreement on training data (arguments with fewer than 50 examples omitted)

greedy-hill-climbing strategy often learned a non-optimal ordering of rules. This is because the system has no look-ahead capability to check whether a sequence of multiple rules applied in succession might produce a good final result despite providing little or no initial improvement.

The addition of a look-ahead searcher has been suggested (Brill, 1995), but we have not seen it implemented in a research context, likely due to the fact that a straightforward implementation of the concept would at minimum square the amount of time required for training.

Instead, we implemented a *look-behind* search strategy, which allows rules to be reordered after discovery. It is meant to address the case in which the system learns a set of rules that each produce improvements in the target function, but interact with each other in a non-optimal way. Whenever our system discovers a new rule, rather than simply applying it and searching for the next rule, it is allowed to try all permutations of the last n discovered rules to see whether performance would improve by using a different ordering. If so, the rules are re-ordered.

To our knowledge, this strategy has not been employed in Transformation-Based Error-Driven learning settings. In our experiments, the strategy discovered transformation sequences that better annotated the input data without using more rules, and therefore seems to produce a labeler less likely to overfit the training data. In our testing, the technique seems to have increased the overall F_1 score by between 0.5% and 1.0%—we caution, however, that we have not undertaken a rigorous comparative study of the technique.

	Precision	Recall	$F_{\beta=1}$
Overall	57.73%	34.35%	43.07
A0	60.92%	52.98%	56.67
A1	53.90%	45.18%	49.16
AM-MOD	99.81%	58.59%	73.83
AM-NEG	44.89%	77.29%	56.79
AM-TMP	38.34%	6.36%	10.92
R-A0	64.61%	76.69%	70.14
V	99.19%	99.19%	99.19

Table 2: Annotation agreement on training data (rows with all-zero entries omitted)

3 Results

The quality of our transformation rules on the training set is shown in Table 2, and the results on the test set are shown in Table 3. The rules that generated these results are shown in Table 4, along with the iterative F_1 scores on the training set as the rules are learned.

4 Discussion

First, note that only one rule was learned in the verb-tagging phase: *Lengthen region **V** if followed by chunk with tag=**PRT***. With earlier releases of the data the system did learn multiple rules, including lexically-based rules, but in later releases only this one rule was learned.

Second, observe that the system actually did reorder rules after discovering them, as evidenced by the non-monotonic "discovery order" column. To attain this result, we used a look-behind of 2, i.e. the last 3 rules learned were candidates for reordering.

Third, several of the rules in the sequence are identical. In some cases, this seems to be because multiple applications of a rule were necessary to achieve full results (e.g. rule "H", which extended an **A0** or **A1** region through joined **NP** chunks several times). In other cases, this seems to be one rule re-applying itself after another rule modified the results of its earlier application (e.g. rule "E", which was affected by applications of rule "H").

Finally, note that only 23 transformations were found. The last few rules begin dealing with lesser-represented argument types like **R-A0** and **AM-NEG**, but many types remain completely unaddressed by the system. We may be able to increase performance on those types by adding additional rule templates, or by decreasing the learning termination threshold for the system. Rule "K" was created as an explicit attempt to recognize **R-A0** and similar argument types, and seems to have been reasonably successful. There may be other relatively simple templates we can create to recognize other arguments.

	Precision	Recall	$F_{\beta=1}$
Overall	58.08%	34.75%	43.48
A0	60.26%	52.73%	56.24
A1	54.53%	44.56%	49.05
A2	0.00%	0.00%	0.00
A3	0.00%	0.00%	0.00
A4	0.00%	0.00%	0.00
A5	0.00%	0.00%	0.00
AM-ADV	0.00%	0.00%	0.00
AM-CAU	0.00%	0.00%	0.00
AM-DIR	0.00%	0.00%	0.00
AM-DIS	0.00%	0.00%	0.00
AM-EXT	0.00%	0.00%	0.00
AM-LOC	0.00%	0.00%	0.00
AM-MNR	0.00%	0.00%	0.00
AM-MOD	100.00%	56.68%	72.35
AM-NEG	48.34%	80.31%	60.36
AM-PNC	0.00%	0.00%	0.00
AM-PRD	0.00%	0.00%	0.00
AM-TMP	40.48%	6.83%	11.68
R-A0	66.45%	64.78%	65.61
R-A1	0.00%	0.00%	0.00
R-A2	0.00%	0.00%	0.00
R-A3	0.00%	0.00%	0.00
R-AM-LOC	0.00%	0.00%	0.00
R-AM-MNR	0.00%	0.00%	0.00
R-AM-PNC	0.00%	0.00%	0.00
R-AM-TMP	0.00%	0.00%	0.00
V	98.21%	98.21%	98.21

Table 3: Results on test data

Rule	Parameters	Discovery order	F_1
	initial annotation		0.00
a	PRT	1	0.00
E	NP, V, A0	2	20.20
I	VBN	5	24.60
B	NP, V, A1	3	31.60
B	S, V, A1	4	35.82
H	A0, NP	6	36.44
D	NN, V, A1	9	36.67
D	NNS, V, A1	10	36.86
E	NP, V, A1	8	37.36
J	VBZ, ”	11	37.46
E	S, V, A1	7	38.55
H	A1, NP	12	38.64
H	A0, NP	13	38.71
H	A1, NP	14	38.78
J	VBD, ”	15	38.83
E	S, V, A1	16	39.07
G	A0, NP	17	39.09
H	A1, NP	18	39.11
C	MD, V, AM-MOD	19	41.23
K	V, WDT, R-A0	20	41.78
K	V, WP, R-A0	21	42.21
A	ADVP, V, AM-TMP	22	42.46
C	RB, V, AM-NEG	23	43.16

Table 4: Rules learned for semantic role labeling

In future work, there are several avenues we would like to explore. Our first-tag-wins assignment strategy mentioned above is not grounded in research into alternate strategies, and in fact we have not yet tried any others.

We also experimented with isolating common verb types into their own corpus—for example, if we train separately on the verb “say,” which represents nearly 10% of the target verbs in the training set and exhibits different argument patterns from other verbs, we achieve an F_1 value of about 82% on this subset using only five learned rules. It may be possible to leverage this work by grouping other less common verbs by their VerbNet class(es).

5 Conclusion

We have described a Transformation-Based Error-Driven learning approach to the CoNLL shared task on semantic role labeling. Although we are relative newcomers to this task and this approach has not to our knowledge been applied to it before, we believe our results are of general interest for the following reasons.

First, the learned output of the system is highly scrutable, in the sense that the transformation rules can easily be reviewed and understood by a human supervisor. This may benefit real-world application of the technique as rules may be manually reordered, switched on or off, or modified. It also allows a developer to closely monitor changes in the system, creating new rules as he or she identifies areas of the data that are being underserved by the current set of transformation templates.

Second, as alluded to above, there are several appealing directions to direct future research, and we believe the results obtained here can be significantly improved.

Third, we know of no previous work using our look-behind reordering technique in conjunction with rule-based learning, and the technique may have broad applicability beyond semantic role labeling.

References

Eric Brill. 1993. *A Corpus-Based Approach to Language Learning*. Ph.D. thesis, Philadelphia, PA.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.