In: *Proceedings of CoNLL-2000 and LLL-2000*, pages 160–162, Lisbon, Portugal, 2000.

Phrase Parsing with Rule Sequence Processors: an Application to the Shared CoNLL Task

Marc Vilain and **David Day**
The MITRE Corporation
Bedford, MA 01730, USA
{mbv, day}@mitre.org

For several years, chunking has been an integral part of MITRE's approach to information extraction. Our work exploits chunking in two principal ways. First, as part of our extraction system (Alembic) (Aberdeen et al., 1995), the chunker delineates descriptor phrases for entity extraction. Second, as part of our ongoing research in parsing, chunks provide the first level of a stratified approach to syntax – the second level is defined by grammatical relations, much as in the SPARKLE effort (Carroll et al., 1997).

Because of our ongoing work with chunking, we were naturally interested in evaluating our approach on the common CoNLL task. In this note, we thus present three different evaluations of our work on phrase-level parsing. The first is a baseline of sorts, our own version of the "chunking as tagging" approach introduced by Ramshaw and Marcus (Ramshaw and Marcus, 1995). The second set of results reports the performance of a trainable rule-based system, the Alembic phrase rule parser. As a point of comparison, we also include a third set of measures produced by running the standard Alembic chunker on the common task with little or no adaptation.

1 Chunking as Tagging

For this first experiment, we coerced our part-of-speech tagger to generate chunk labels. We did so in what can only count as the most rudimentary way: by training the tagger to map part-of-speech labels to the chunk labels of the common task. The learning procedure is a re-implementation of Brill's transformation-based approach (Brill, 1993), extended to cover approximately an order of magnitude more rule schemata. As input, the training corpus was tagged with the parts-of-speech from the common data set: these provided an initial labeling of the data which was then directly converted to chunk labels through the action of transformation rules (Brill's so-called contextual rules).

N	accuracy	precision	recall	$F_{\beta=1}$
1000	86	77	77	77
2000	89	82	82	82
4000	89	81	81	81

Table 1: Performance of the brute-force re-tagging approach

Because the learning procedure is none the swiftest, we restricted ourselves to subsets of the training data, acquiring rules from the first 1000, 2000, and 4000 sentences of the training set. In each case, we acquired 500 transformation rules. We measured the following performance of these rules on the test set.

These results are hardly stellar, falling some 10 points of F below the performance of previous approaches to noun group detection. To be sure, the chunking task is more demanding than the simple identification of noun group boundaries, so one would expect lower performance on the harder problem. But the rudimentary way in which we implemented the approach is likely also to blame.

There are a number of clear-cut ways in which we could attempt to improve our performance using this approach. In particular, we would expect to obtain better results if we did not obliterate the part-of-speech of a lexeme in the process of tagging it with a chunk label. Indeed, in our experiments, the learning procedure acquired transformations that simply replaced the part-of-speech tag with a chunking tag, thereby inhibiting potentially useful downstream rules for accessing the part-of-speech information of a chunk-tagged word.

2 Chunking with the Phrase Rule Parser

Our main interest in this common evaluation, however, was not to set new high-water marks with the approach of Ramshaw and Marcus, but to exercise our phrase rule parser.

The Alembic phrase rule parser (Vilain and Day, 1996) provides the core of the system's syntactic processing. In our extraction applications, the phraser (as we call it) initially tags named entities and other fixed-class constructs (like titles). The phraser also treats as atomic units the stereotypical combinations of named entities that one finds in newswire text, e.g., the person-title-organization apposition "U.N. secretary general Kofi Anan". The three components of the apposition are initially parsed as fixed-class entities, and are then combined to form a single person-denoting phrase. These preliminary parsing steps provide part of the input to the chunker, which is itself implemented as a phrase rule parser.

The architecture of the parser is based on Brill's approach. The parser follows a sequence of rules in order to build phrases out of parse islands. These islands are initially introduced by instantiating partial phrases around individual lexemes (useful for name tagging), or around runs of certain parts of speech (useful for both name tagging and chunking). It is the job of the phrase parsing rules to grow the boundaries of these phrases to the left or right, and to assign them a type, e.g., a name tag or a chunk label. As with other rule sequence processors, the phraser proceeds in sequence through its catalogue of rules, applying each in turn wherever it matches, and then discarding it to proceed on to the next rule in the sequence.

For example, in name tagging, we seed initial phrases around runs of capitalized words. A phrase such as "meetings in Paris and Rome" would produce an initial phrase analysis of "meetings in <?>Paris</?> and <?>Rome</?>", where the "?" on the phrases are initial labels that indicate the phrase has not received a type.

The patterns that are implemented by phrase parsing rules are similar to those in Brill's transformation-based p-o-s tagger. A rule can test for the presence of a given part of speech, of a lexeme, of a list of lexemes, and so on. These tests are themselves anchored to a specific locus (a phrase or lexeme) and are performed relative to that locus. As actions, the rules can grow the boundaries of a phrase, and set or modify its label. For example, a typical name tagging rule would assign a LOCATION tag to any phrase preceded by the preposition "in". And indeed, this very rule tends to emerge as the very first rule acquired in training a phraser-based name tagger. We show it here with no further comment, trusting that its syntax is self-evident.

```
(def-phraser-rule
  :conditions (:left-1 :lex "in")
  :actions    (:set-label :LOCATION))
```

In our particular example ("meetings in <?>Paris</?> and <?>Rome</?>"), this rule would re-label the <?> phrase around Paris with the LOCATION tag. A subsequent rule might then exploit the coordination to infer that "Rome" is a location as well, implementing the transformation "LOCATION and <?>" → "LOCATION and LOCATION". This incremental patching of errors is the hallmark of Brill's approach.

An interesting property of this rule language is that the phraser can be operated either as a trainable procedure, using standard error-driven transformation learning, or as a hand-engineered system. For the purpose of the common CoNLL task, let us first present our results for the trainable case.

We again approached the task in a relatively rudimentary way, in this case by applying the phrase rule learning procedure with no particular adaptation to the task. Indeed, the procedure can be parameterized by word lists which it can then exploit to improve its performance. Since our main interest here was to see our baseline performance on the task, we did not harvest such word lists from the training data (there is an automated way to do this). We ran a number of training runs based on different partitions of the training data, with the following overall performance on test data, averaged across runs.

accuracy	precision	recall	$F_{\beta=1}$
89	89	83	86

The constituents that were most accurately recognized were noun groups (F=88), with verb groups a close second (F=87). These were followed by the ostensibly easy cases of PP's (F=86), SBAR's (F=79), and ADVP's (F=75). Our lowest performing constituent for which the learning procedure actually generated rules was ADJP's (F=37), with no rules generated to identify CONJP's, INTJ's, LST's, or PRT's (F=0 in all these cases).

In general, precision, tended to be several points of F higher than recall, and in the case of ADJP's average precision was 76 compared to average recall of 24!

3 Chunking with the Hand-Engineered System

As a point of comparison, we also applied our hand-engineered chunker to the CoNLL task. We expected that it would not perform at its best on this task, since it was designed with a significantly different model of chunking in mind, and indeed, unmodified, it produced disappointing results:

accuracy	precision	recall	$F_{\beta=1}$
84	80	75	77

The magnitude of our error term was something of a surprise. With production runs on standard newswire stories (several hundred words in lengths) the chunker typically produces fewer errors per story than one can count on one hand. The discrepancy with the results measured on the CoNLL task is of course due to the fact that our manually engineered parser was designed to produce chunks to a different standard.

The standard was carefully defined so as to be maximally informative to downstream processing. Generally speaking, this means that it tends to make distinctions that are not made in the CoNLL data, e.g., splitting verbal runs such as "failed to realize" into individual verb groups when more than one event is denoted.

Our curiosity about these discrepancies is now piqued. As a point of further investigation, we intend to apply the phraser's training procedure to adapt the manual chunker to the CoNLL task. With transformation-based rule sequences, this is easy to do: one merely trains the procedure to transform the output required for the one task into that required for the other. The rules acquired in this way are then simply tacked on to the end of the original rule sequence (a half dozen such rules written by hand bring the performance of the chunker up to F=82, for example).

test data	precision	recall	$F_{\beta=1}$
ADJP	75.89%	24.43%	36.96
ADVP	80.64%	70.21%	75.06
CONJP	0.00%	0.00%	0.00
INTJ	0.00%	0.00%	0.00
LST	0.00%	0.00%	0.00
NP	87.85%	87.77%	87.81
PP	91.77%	80.42%	85.72
PRT	0.00%	0.00%	0.00
SBAR	91.36%	69.16%	78.72
VP	90.34%	84.13%	87.13
all	88.82%	82.91%	85.76

Table 2: The results of the phrase rule parser.

A more interesting point of investigation, however, would be to analyze the discrepancies between current chunk standards from the standpoint of syntactic and semantic criteria. We look forward to reporting on this at some future point.

References

J. Aberdeen, J. Burger, D. Day, L. Hirschman, P. Robinson, and M. Vilain. 1995. Mitre: Description of the alembic system used for muc-6. In *Proc. 6th Message Understanding Conference (MUC-6)*. Defense Advanced Research Projects Agency, November.

E. Brill. 1993. *A Corpus-based Approach to Language Learning.* Ph.D. thesis, U. Pennsylvania.

J. Carroll, T. Briscoe, N. Calzolari, S. Federici, S. Montemagni, V. Pirrelli, G. Grefenstette, A. Sanfilippo, G. Carroll, and M. Rooth. 1997. SPARKLE work package 1, specification of phrasal parsing, final report. Available at `http://www.ilc.pi.cnr.it/sparkle/-sparkle.htm`, November.

L. Ramshaw and M. Marcus. 1995. Text chunking using transformation-based learning. In *Proc. of the 3rd Workshop on Very Large Corpora*, pages 82–94, Cambridge, MA, USA.

M. Vilain and D. Day. 1996. Finite-state phrase parsing by rule sequences. In *Proceedings of the 16th Intl. Conference on Computational Linguistics (COLING-96).*