An investigation into compositional features and feature merging for maximum entropy-based parse selection
Mullen, Anthony James

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2002

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.
Chapter 7

Further experiments feature merging with the Alpino Grammar

7.1 Another context for feature merging

In this chapter I discuss another environment in which a combination of feature merging and a frequency based cutoff has shown to be beneficial to the task of maximum entropy-based parse selection. The work reported here follows upon work reported in Mullen, Malouf, and van Noord (2001). The grammar looked at is the Alpino grammar of Bouna, van Noord, and Malouf (2001). The experimental conditions in the Alpino framework are quite different from those presented in chapters 5 and 6, but the task is similar, and the technique of feature merging is employed in an analogous manner with comparable success.

As will be discussed, the parsing task is approached from a different perspective in the experiments in this chapter, and the features themselves are of a different nature. Compositionality of features occurs in a different way, since the structure of the features is different, although in many cases the feature elements are similar to those discussed in chapter 6. The similarities and differences will be discussed in greater detail later in this chapter.

Another difference between the two chapters’ experiments lies in the means of evaluation. The previous task was approached strictly as a ranking exercise. The data I worked with did not facilitate deep analysis of the meaning of the results, or the significance of various parses’ similarity to each other, as discussed in section 6.5.1, and therefore the exact match of top-ranked parses was the most useful alternative for evaluating the ranking. This was primarily due to the amalgamation of various notational and grammatical assumptions which went into building the dataset used in the experiments. The Alpino framework, on the other hand, is a single, dedicated environment for grammar building and
7.2 Alpino: Wide-coverage Parsing of Dutch

Alpino is a wide-coverage computational analyzer of Dutch which aims at accurate full parsing of unrestricted text. The system is described in more detail in (Bouma, van Noord, and Malouf, 2001). The grammar produces dependency structures, thus providing a reasonably abstract and theory-neutral level of linguistic representation. The dependency relations encoded in the dependency structures are used to develop and evaluate disambiguation methods.

7.2.1 Grammar

The Alpino grammar is an extension of the successful OVIS grammar (van Noord et al., 1999; van Zanten et al., 1999), a lexicalized grammar in the tradition of Head-driven Phrase Structure Grammar (Pollard and Sag, 1994). The grammar formalism is carefully designed to allow linguistically sophisticated analyses as well as efficient and robust processing.

In contrast to earlier work on HPSG, grammar rules in Alpino are relatively detailed. However, as pointed out in (Sag, 1997), by organizing rules in an inheritance hierarchy, the relevant linguistic generalizations can still be captured. The Alpino grammar currently contains over 100 rules, defined in terms of a few general rule structures and principles. The grammar covers the basic constructions of Dutch (including main and subordinate clauses, (indirect) questions, imperatives, (free) relative clauses, a wide range of verbal and nominal complementation and modification patterns, and coordination) as well as a wide variety of more idiosyncratic constructions (appositions, verb-particle constructions, PP’s including a particle, NP’s modified by an adverb, punctuation, etc.). The lexicon contains definitions for various nominal types (nouns with various complementation patterns, proper names, pronouns, temporal nouns, deverbalized nouns), various complementizer, determiner, and adverb types, adjectives, and about 100 verbal subcategorization types.

The formalism supports the use of recursive constraints over feature-structures (using delayed evaluation, (van Noord and Bouma, 1994)). This allowed us to incorporate an analysis of cross-serial dependencies based on argument-inheritance (Bouma and van Noord, 1998) and a trace-less account of extraction along the
7.2. **ALPINO: WIDE-COVERAGE PARSING OF DUTCH**

lines of (Bouma, Malouf, and Sag, 2001).

### 7.2.2 Robust Parsing

The construction of a dependency structure (see below) proceeds in two steps. In the first step a parse forest is constructed. The second step consists of the selection of the best parse from the parse forest.

**Creating Parse Forests.** The Alpino parser takes the set of feature structures found during lexical analysis as its input, and constructs a *parse forest*: a compact representation of all parse trees. The Alpino parser is a left-corner parser with selective memoization and goal-weakening. It is a variant of the parsers described in (van Noord, 1997).

**Unpacking and Parse Selection.** The motivation to construct a parse forest is efficiency: the number of parse trees for a given sentence can be enormous. In addition to this, in most applications the objective will not be to obtain all parse trees, but rather the *best* parse tree. Thus, the final component of the parser consists of a procedure to select these best parse trees from the parse forest.

In order to select the best parse tree from a parse forest, we assume a parse evaluation function which assigns a score to each parse. In Bouma, van Noord, and Malouf (2001) some initial experiments with a variety of parse evaluation functions are described. A naive algorithm constructs all possible parse trees, assigns each one a score, and then selects the best one. Since it is too inefficient to construct all parse trees, we have implemented the algorithm which computes parse trees from the parse forest as a best-first search. This requires that the parse evaluation function is extended to partial parse trees. We implemented a variant of a best-first search algorithm in such a way that for each state in the search space, we maintain the *b* best candidates, where *b* is a small integer (the *beam*). If the beam is decreased, then we run a larger risk of missing the best parse (but the result will typically still be a relatively ‘good’ parse); if the beam is increased, then the amount of computation increases too.

### 7.2.3 Dependency Structures

The Alpino grammar produces dependency structures compatible with the CGN-guidelines. Within the CGN-project (Oostdijk, 2000), guidelines have been developed for syntactic annotation of spoken Dutch (Moortgat, Schuurman, and van der Wouden, 2000), using dependency structures similar to those used for the German Negra corpus (Skt, Krenn, and Uszkoreit, 1997). Dependency structures make explicit the dependency relations between constituents in a
sentence. Each non-terminal node in a dependency structure consists of a head-daughter and a list of non-head daughters, whose dependency relation to the head is marked. A dependency structure for the sentence

Mercedes zou haar nieuwe model gisteren hebben aangekondigd
Mercedes should her new model yesterday have announced

Mercedes should have announced its new model yesterday

is given in figure 7.1. Control relations are encoded by means of co-indexing (i.e. the subject of hebben is the dependent with index 1). Note that a dependency structure does not necessarily reflect (surface) syntactic constituency. The dependent haar nieuwe model gisteren aangekondigd, for instance, does not correspond to a (surface) syntactic constituent.

7.2.4 Treebank

We have started to annotate various smaller fragments with dependency structures. The largest fragment consists of a subset of the cdbl (newspaper) part of the Eindhoven corpus (den Boogaart, 1975). This treebank is used in the experiments described below. It contains 1,396 sentences (16,925 words): all sentences of twenty words or less from the first 2,500 sentences of Eindhoven-cdbl.
7.3 Maximum entropy modeling and Alpino

The most important aspect of the maxent modeling technique is that distributions of statistical features are modeled without requiring an assumption that the features be independent. This allows accurate modeling using feature sets in which the features’ distributions are dependent upon each other. Exploiting this fact is an important consideration in constructing feature sets for maxent modeling.

For parse selection, a context is considered to be a sentence and the events within this context are the possible parses of the sentence. Each parse is characterized by a set of feature values, and may be compared on the basis of those features with other possible parses. Parsing is performed as described in section 7.2.2. Following Johnson et al. (1999), the best-first search proceeds on the basis of the unnormalized conditional probabilities derived from equation 3.43 for each possible subtree.

7.4 The Features and Feature Merging

The first step in modeling data using log linear models is to decide on an initial set of features to be modeled. The model depends on the distribution of these features and their informativeness, thus it is important that the features used be germane to the task. In parsing, features should reflect the sort of information pertinent to making structural decisions.

In the present experiments, several types of features are employed corresponding to grammatical rules, valency frames, lexicalized dependency triples, and lexical features constituting surface forms, base forms, and lexical frames. Instances of each feature type were collected from the training data in advance to yield a feature set consisting of 82,371 distinct features. Note that these do not all correspond structurally to the feature templates described in chapter 5 and used in chapter 6. The features used here vary structurally from that template; the underlying notion of compositionality, however, is kept. Thus the merging technique employed will be a variation of that used in chapter 6.

Examples of the features used may be seen below, where example 1 is a rule for creating a VP, 2 contains a valency frame for the noun *mens*, and 3 describes a dependency triple between the noun *mens* and the adjective *modern*, and the direction of the modification. The final two feature types, 4 and 5, are added separately, for two different experiments. In the experiments described in section 7.5.1, the feature type 4, below is added, in which features contain lexical information about each word in the sentence as it occurs in context. In the subsequent experiments, reported in section 7.6.1, multiple rule features are employed, which look not only at single rules, but at depth-two expansions of rules; that is, a mother rule and the rules corresponding (expanding) each of
the daughters of that rule. These rules, as will be discussed in section 7.6.1 are designed in such a way as to maintain meaningful compositionality, that is, to make sure that all appropriate information is available to each sub-feature element.

1) rule \( vp_{arg\_v}(np) \)
2) subcategorization frame \( \text{noun (de): mens: [mod]} \)
3) dependency triple \( \text{noun: mens: mod: left: adjective: modern} \)
4) lexical feature \( \text{moderne: modern: adjective (e, adv)} \)
5) multiple rule feature \( r(vp_{mod\_v}, mod2, 1): r(vp_{mod\_v}, vp_{mod\_v}, 2) \)

### 7.4.1 Noise reduction and feature merging

The feature set used here exploits the maxent technique in that it relies on features which are overlapping and mutually dependent. The features represent varying degrees of linguistic generality and hence some occur much more frequently than others. Furthermore, the features may also represent information which is redundant in the sense that it is represented in multiple different features, in which case it is said that the features “overlap”. Features which share information in this way are necessarily dependent in their distributions.

The overlapping features allow for a variety of “backing off” in which features which share a structure but contain less specific information than others are used in the same model as features with more specific information.

It is desirable that the features be as informative as possible. The model should contain specific features to the extent that the features’ distributions are accurately represented in the training data. There is a point, however, regardless of the size of the corpus, at which the specificity of features translates to sparseness in the data, causing noise and leading to deterioration in the model’s performance. Although it is an effective way to reduce noise in a model, there is a risk with a cutoff that information encoded in the discarded features may be useful. A feature may be rare due to some rare element within it, but otherwise useful. To prevent discarding such useful features, the method of feature merging is again employed. This approach, in general terms, considers features as being composed of informative elements. Before any feature cutoff is applied, features which are identical except for particular rare elements are generalized by merging; that is, the features are unioned and considered as a single feature. The elements upon which these merges are done are determined with a pre-set threshold, and merges are done on elements which occur fewer times than this. The merging process eliminates the distinction between two features, thus eliminating the information provided by the element which distinguishes them, while the rest of the information provided by the merged features remains intact in the model.
Individual unique features may be considered as sets of instantiations in the data. A feature which is the result of merging is thus the union of the features which were merged. The count of occurrences of the new feature is the sum of the counts of the merged features. If a cutoff is incorporated subsequently, the newly merged feature is more likely to survive in the model, as its frequency is greater than the features which were merged. Thus information in features which otherwise might have been lost in a cutoff is retained in the form of a more general feature.

7.4.2 Why feature merging?

It is well accepted that many models benefit from a frequency based feature cutoff. It is hoped that by using feature merging, a more sophisticated view of the features themselves may be taken, allowing for the same degree of noise reduction as a feature cutoff, while simultaneously generalizing the features to obtain the benefits of backing off. By operating on sub-feature information sources, it may be possible to discard noisy information from the model with a greater degree of control, maintaining useful information contained by features which would otherwise be lost. The experiments described in chapter 6 suggest that in cases where a significant benefit may be gained through a cutoff, it is also likely that further benefits may be derived from prior merging. However, such improvements decrease with the degree of richness of the original model, as richer models perform better to begin with and benefit less from noise reduction.

7.4.3 Building merged models

The first step is to determine how the features are composed and what the elements are which make them up. Factors which contribute most to sparseness, such as lexical items and certain grammatical attributes, are good candidates. In the present chapter’s work, lexical items, both sentence forms and word stems, are considered as elements, as are lexical frames and relations lists. Frequency counts are taken for all elements. A threshold is determined using a held-out test set. Using this threshold, a new model is created as follows: in the representation of the original model’s features, all instances of elements which occur fewer times than the threshold are replaced by a dummy element. Features which were identical aside from these infrequent elements are thus rendered completely identical. For example, let

\[
\text{feature 1} = A : B \quad \text{with count 2} \\
\text{feature 2} = A : C \quad \text{with count 4}
\]

where \(A\), \(B\), and \(C\) are elements. We may count the occurrences of each element in the training data and find that the count of \(A\) is 20, of \(B\) is 5, and of \(C\) is 7. We determine by use of held-out test data that an optimal cutoff is, e.g., 10. Since both \(B\) and \(C\) have counts lower than this, all instances of \(B\) and \(C\)
are replaced by a dummy element $X$. Thus features 1 and 2 above are both in effect replaced by feature 3, below, whose count is now the sum of those of the features which have been merged.

$$\text{feature 3} = A : X \quad \text{with count 6}$$

Iterative scaling is performed on the new feature set to obtain the appropriate maximum entropy weights.

### 7.4.4 Composition of lexicalized features

A quality of compositionality is necessary in features in order to perform the merging operation. That is, it is necessary that features be composed of discrete elements for which frequency counts can be attained from the data. The features described in section 7.4 may be viewed as being composed of words, base forms, POS tags, grammar attributes, and other discrete elements which occur together in a particular way. Merging proceeds by first establishing a merging threshold via experiments on held-out data. Frequencies of all elements are gathered from the training data. Finally, features containing elements whose counts are fewer than the threshold are merged. This is done by replacing all instances of sub-threshold elements with a dummy element in features. For example, if it were found that the element `modern` had a count less below the threshold, all features containing that would be altered. A feature such as

```
noun:mens:mod:left:adjective:modern
```

would be changed to

```
noun:mens:mod:left:adjective:xxxxx
```

and likewise if the element `aardig` occurred with a count below the threshold, the same would be done with the feature

```
noun:mens:mod:left:adjective:aardig
```

so that both features merged as the single feature

```
noun:mens:mod:left:adjective:xxxxx
```

with a count equal to the sum of the two merged features.

### 7.4.5 Composition of multiple rule features

Multiple rule features were introduced in the hopes that they would be both infrequent enough to warrant noise reduction and yet informative enough that mitigating the loss of information brought about by a cutoff would be useful for modeling. These features incorporate the rule of each mother node as well as the rule of the daughter nodes of the mother. The example given as number 5 in the list in section 7.4 is repeated here:
7.4. THE FEATURES AND FEATURE MERGING

Figure 7.2: A visual description of the information contained in the feature \( r(vp_{mod_v}, mod2, 1) : r(vp_{mod_v}, vp_{mod_v}, 2) \). Note the redundancy in the compositional encoding.

![Diagram](image)

Figure 7.3: The two elements which compose the feature \( r(vp_{mod_v}, mod2, 1) : r(vp_{mod_v}, vp_{mod_v}, 2) \) are shown. Each element contains the rule of the mother, and the rule of a daughter, together with the numerical index of the daughter.

\[ r(vp_{mod_v}, mod2, 1) : r(vp_{mod_v}, vp_{mod_v}, 2) \]

The elements of this feature are separated by a colon, so this feature is composed of two elements. The structure of the feature is shown in figure 7.2. The information contained in each element may be seen in figure 7.3.

It is notable that the feature itself is redundant; the mother rule is repeated in both elements of the feature, and the numerical index of the daughter node is repeated explicitly within the element, rather than only encoded implicitly in the order of the feature. This is deliberate, in the interest of the notion of compositionality discussed in chapter 5. In that chapter, in section 5.4, the relation between the information contained in the feature and the information supplied by its component elements is considered to be a primary consideration for compositionality. This is primarily a question of how information is encoded; in the present case a certain degree of redundancy is necessary in order to ensure that the elements of the feature contain the information from the feature that we wish them to.

A non-compositional, non-redundant variation of the feature above might look something like this:

\[ vp_{mod_v}, mod2, vp_{mod_v} \]

where the first rule is the mother rule and the subsequent rules are the daughter rules counting from left to right. Here, all the information of the whole feature,
as shown in figure 7.2 is present, however, the elements may not be distinguished.

What kind of information elements contain is up to the person building the model. Following the same notation of colons as element separators, it is perfectly possible to represent the same information in the feature discussed above in a compositional feature such as:

\[ v_p:mod_v:mod2:v_p:mod_v \]

However, in this case the elements encode nothing more than the names of rules. This means that the element counts which are compared with the merging threshold are simply the counts of rules. This is inadequate for the present purposes in part because rules, generally, in the Alpino data, have fairly high counts. It is more useful to encode elements with more specific information about their place in the feature. This makes them more likely to occur under the merging threshold, since they will necessarily be more sparse, and also, in cases where they occur more frequently, will make them more informative. In particular, it is desirable that the elements contain pertinent to determining syntactic structure, since it is hoped that generalizations over elements will help to generalize over structures. Thus the present encoding is used, which allows elements such as those in figure 7.3 to be easily distinguished.

### 7.5 Experiments

Models built with two different feature sets were investigated; the first feature set included lexicalized features, constructed in the manner described in section 7.4.4, and the second feature set, rather than lexicalized features, incorporated features made up of multiple-rule trees, which will be described in detail in section 7.6.1. The idea which motivates these types of features is the intuition that a model would benefit most from the merging/cutoff combination if it has information sources which are potentially noisy but that also contribute (potentially) valuable information.

#### 7.5.1 Experiments with lexicalized features

Experiments were performed using the features described above with a training set of 1,220 sentences, whose parses totalled 626,699 training events, and a test set of 131 sentences. A merging threshold of 500 and a feature cutoff of 500 were determined by use of a held out test set. The number of active (non-zero weighted) features in the original model was 75,500, the number of active features in the model with cutoff alone was 11,639, and the number of active features in the model which had been merged prior to the cutoff was 11,890.
7.5.2 Evaluation

For each sentence in the test set, the dependency structure of the highest scoring parse is extracted and compared to the gold standard dependency structure in the treebank (Carroll, Briscoe, and Sanfilippo, 1998). For each sentence, the overlap is calculated: the number of relations for which the system agrees with the gold standard parse. Using this, it is possible to calculate the number of errors:

\[
\text{errors} = \max (\text{gold}, \text{system}) - \text{overlap}
\]

From the error, the concept accuracy (CA) can be computed (Boros et al., 1996; van Zanten et al., 1999) as follows:

\[
\text{CA} = 100 \times \left( 1 - \frac{\text{error}}{\text{gold}} \right)
\]

The results given in tables 7.1 and 7.2 are in terms of average per-sentence concept accuracy.

In addition to the concept accuracy, a second metric is also employed, which allows the models to be more broadly compared to others by incorporating not only the per-sentence concept accuracy of the model but also the baseline per-sentence concept accuracy (the per-section concept accuracy obtained by arbitrary selection of a parse) and the best possible per-sentence concept accuracy. This latter score reflects the accuracy of the grammar, and provides an upper bound for the task of picking the best possible of all available parses. This measure is proposed in Hawkins (1999) as the “Upper Bound Adjusted Kappa” metric and re-christened in Mullen, Malouf, and van Noord (2001) as the phi measure, shown in equation 7.1:

\[
\Phi = 100 \times \frac{\text{CA} - \text{baseline}}{\text{best} - \text{base}}
\]

where CA is the average per-sentence concept accuracy of the model, base is the average per-sentence concept accuracy achieved by arbitrarily selecting parses, and best is the average per-sentence concept accuracy that could be achieved if the parser always selected the best possible parse licensed by the grammar.

7.6 Results with lexical features

Table 7.1 shows the initial results for held-out data, where the cutoff and merge threshold levels were optimized for best performance. On the relatively small held-out set it appears that the best performance is reached by the model which employs the combination of cutoff and merge. The results for this test set are statistically significant, although just barely, due to the small size of the test set. The suggestion that the cutoff/merged model is truly superior is present,
Table 7.1: Preliminary results on held-out data with lexicalized features

<table>
<thead>
<tr>
<th>Model</th>
<th>CA</th>
<th>( \Phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>63.01</td>
<td>0.00</td>
</tr>
<tr>
<td>maxent</td>
<td>77.45</td>
<td>56.99</td>
</tr>
<tr>
<td>maxent+cutoff</td>
<td>79.36</td>
<td>64.52</td>
</tr>
<tr>
<td>maxent+cutoff+merge</td>
<td>80.01</td>
<td>67.08</td>
</tr>
<tr>
<td>best possible</td>
<td>88.35</td>
<td>100.00</td>
</tr>
</tbody>
</table>

Table 7.2: Results on unseen data, with lexicalized features

<table>
<thead>
<tr>
<th>Model</th>
<th>CA</th>
<th>( \Phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>58.92</td>
<td>0.00</td>
</tr>
<tr>
<td>maxent</td>
<td>71.53</td>
<td>45.77</td>
</tr>
<tr>
<td>maxent+cutoff</td>
<td>73.67</td>
<td>53.54</td>
</tr>
<tr>
<td>maxent+cutoff+merge</td>
<td>73.26</td>
<td>52.05</td>
</tr>
<tr>
<td>best possible</td>
<td>86.47</td>
<td>100.00</td>
</tr>
</tbody>
</table>

but tenuous. Table 7.2, however appears to refute the suggestion. This test set
is bigger and the cutoff and merge thresholds are fixed at the levels determined
with the held-out data set. It is of course possible that the combined model was
in fact the best possible model for the held-out test set. If the results of the
held-out experiments were true, then it is the case that the merge threshold level
failed to generalize well. The cutoff established by the held-out set, on the other
hand, clearly did yield an improvement on the unseen data.

### 7.6.1 Experiments with multiple-rule features

The results of the experiments using lexicalized features appear to indicate that
the lexicalized features do in fact contribute noise to the model, hence the benefit
derived from the cutoff. Unfortunately, it appears that given the quite rich fea-
ture set, which includes the features 1 through 3 listed in section 7.4, the lexical
features contribute little or nothing to the modeling themselves. For this reason,
merging does not appear to improve matters beyond the use of the cutoff. An-
other approach to features was attempted, employing the multiple rule features
described in section 7.4.5. It was hoped that taking this different and somewhat
unconventional route with modeling might bring about some improvement over
the models employing the lexical features.

On the contrary, preliminary tests on the same held-out set as previously
showed considerably worse performance than the lexicalized model, as can be
seen in table 7.3; the multiple rule features do not appear to contribute in a
positive way to the modeling. Number of features in original set was 71,555. Merging, on its own showed some improvement, resulting in CA of 73.84. Once again, a cutoff of 500 was found to yield the best results, although in this case the improvement was considerably less marked, increasing only from a CA score of 72.34 to 75.05.

No promise of further improvement through combining merging with a cutoff was found. No better merging threshold was found than 500, which, combined with the cutoff, yielded slightly worse results than the cutoff alone, a concept accuracy of 74.36. Given the size of the data set, this is not a significant difference, but it certainly does not suggest that an improvement is to be gained by merging.

## 7.7 Conclusion

Given the sparseness of the data available to training the Alpino models, the feature merging technique, shown in chapter 6 to be helpful in cases where noise presents a major problem for models, seemed like a promising approach to improving modeling. Indeed, preliminary investigations with held-out data to find optimal cutoffs and merge thresholds seemed to offer reason for optimism. On a small held-out dataset, for which the threshold and cutoff levels could be tweaked and adjusted until optimal, the best results were indeed reached using a combination of merging and cutoff approaches. Unfortunately, tests on unseen data, where the cutoff and merge threshold levels were held fixed, showed no benefit gained by merging prior to the cutoff. The improvements seen in the preliminary experiments with lexicalized features were only barely statistically significant, and seem likely to have arisen due to fortuitous selection of the test data. Further experiments, using somewhat more exotic feature types, showed no such promise, even tested initially on the same data. Further investigation into feature merging for parsing in the Alpino environment does not appear to be warranted.

<table>
<thead>
<tr>
<th>Model</th>
<th>CA</th>
<th>$\Phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>63.01</td>
<td>0.00</td>
</tr>
<tr>
<td>maxent</td>
<td>72.34</td>
<td>36.82</td>
</tr>
<tr>
<td>maxent+cutoff</td>
<td>75.05</td>
<td>47.51</td>
</tr>
<tr>
<td>maxent+merge only</td>
<td>73.84</td>
<td>42.74</td>
</tr>
<tr>
<td>maxent+cutoff+merge</td>
<td>74.36</td>
<td>44.79</td>
</tr>
<tr>
<td>best possible</td>
<td>88.35</td>
<td>100.00</td>
</tr>
</tbody>
</table>

Table 7.3: Results on a held-out set, with multiple rule features