Cascaded Grammatical Relation Assignment

Sabine Buchholz and Jorn Veenstra and Walter Daelemans

ILK, Computational Linguistics, Tilburg University
PO Box 9013, 5000 LE Tilburg, The Netherlands
[buchholz, veenstra, daelemans]@kub.nl

Abstract

In this paper we discuss cascaded Memory-Based grammatical relations assignment. In the first stages of the cascade, we find chunks of several types (NP, VP, ADJP, ADVP and adverbial functions) and label them with their adverbial function (e.g. local, temporal). In the last stage, we assign grammatical relations to pairs of chunks. We studied the effect of adding several levels to this cascade and the effect of using different modules in each level. We found that even the least performing chunkers enhanced the performance of the relation finder.

Introduction

When dealing with large amounts of text, it is often useful to perform shallow processing on large amounts of text. However, when dealing with natural language text, it is often necessary to perform more complex processing on smaller amounts of text. This is especially true when dealing with text that is in a natural language.

In the rest of this paper we will discuss some Memory-Based shallow parsing techniques to find labeled chunks and grammatical relations in a sentence. Several MB modules have been developed in previous work, such as: a POS tagger (Daelemans et al., 1996), a chunker (Veenstra, 1998; Tjong Kim Sang and Veenstra, 1999) and a grammatical relation assigner (Buchholz, 1999). The questions we will answer in this paper are: Can we reuse these modules in a cascade? What is the effect of cascading? Will errors at a lower level percolate to higher modules?

Recently, many people have looked at cascaded and/or shallow parsing and GR assignment. Abney (1991) is one of the first to propose to split up parsing into several cascades. He suggests to first find the chunks and then the dependencies between these chunks. Grefenstette (1996) describes a cascade of finite-state transducers, which first finds noun and verb groups, then their heads, and finally syntactic functions. Brants and Skut (1998) describe a partially automated annotation tool which constructs a complete parse of a sentence by recursively adding levels to the tree. (Collins, 1997; Ratnaparkhi, 1997) use cascaded processing for full parsing with good results. Argamon et al. (1998) applied Memory-Based Sequence Learning (MBSL) to NP chunking and subject/object identification. However, their subject and object finders are independent of their chunker.

In this paper we will explicitly study the effect of adding steps to the grammatical relations assignment cascade. Through experiments with cascading several classifiers, we will show that compared to using independent classifiers, the cascaded classifier performs several steps better than the classifier on the single classifier.

In the rest of this paper, we will first briefly describe Memory-Based Learning in Section 2. Then, we will describe the cascaded classifier in Section 3. Finally, we will present our experiments and results in Section 4.
In the first experiment described in this section, we discuss the chunking classifiers that we later use as steps in the cascade. Section 3.2 describes the basic CR classifier. Section 3.3 presents the architecture and results of the cascaded CR assignment experiments. We discuss the results in Section 4 and conclude with Section 5.

Memory-Based Learning

Memory-Based Learning (MBL) keeps all training data in memory and only abstracts at classification time by extrapolating a class from the most similar item(s) in memory. In recent work Daelemans et al. (1999b) have shown that for typical natural language processing tasks, this approach is advantageous because it also remembers "exceptional," low-frequency cases which are useful to extrapolate from. Moreover, automatic feature weighting in the similarity metric of an MB learner makes the approach well-suited for domains with large numbers of features from heterogeneous sources, as it embodies a smoothing-by-similarity method when data is sparse (Zavrel and Daelemans, 1997).

We have used the following MBL algorithms:
- IB1: A variant of the k-nearest neighbor (k-NN) algorithm. The distance between a test item and each memory item is defined as the number of features for which they have a different value (overlap metric).
- IB1-IG: IB1 with information gain (an information-theoretic notion measuring the reduction of uncertainty about the class to be predicted when knowing the value of a feature) to weight the cost of a feature value mismatch during comparison.
- IGT tree: In this variant, a decision tree is created with features as tests, and ordered according to the information gain of the features, as a heuristic approximation of the computationally more expensive IB1 variants.

For more references and information about these algorithms we refer to (Daelemans et al., 1999b; Daelemans et al., 1998b).

For the experiments described in this paper we have used TiMBL, an MBL software package developed in the ILK-group (Daelemans et al., 1998). TiMBL is available from: http://ilk.kub.nl/.

We have used the following MBL algorithms:
- IB1: With information gain (can have a different value (overlap metric)).
- IB1-IG: With information gain (can have a different value (overlap metric)).

We have used the following MBL algorithms:
- IB1: With information gain (can have a different value (overlap metric)).
- IB1-IG: With information gain (can have a different value (overlap metric)).

We have used the following MBL algorithms:
- IB1: With information gain (can have a different value (overlap metric)).
- IB1-IG: With information gain (can have a different value (overlap metric)).

We have used the following MBL algorithms:
- IB1: With information gain (can have a different value (overlap metric)).
- IB1-IG: With information gain (can have a different value (overlap metric)).

We have used the following MBL algorithms:
- IB1: With information gain (can have a different value (overlap metric)).
- IB1-IG: With information gain (can have a different value (overlap metric)).
In grammatical relation assignment we assign a GR to pairs of words in a sentence. In our approach for PP-chunking and ADVFUNCs we use a local context information, denoted by windowing features, determined by validation experiments. The classes are the last chunking-labeling step, we assign adjectives to GR assignment (see Section 3.2). In the present case, elements are words or punctuation marks, as shown in Figure 1 and we wanted the algorithm to decide whether, and if so how, to extract the disambiguated chunks. The disambiguated chunks are the resolved chunks from the treebank. A prep position is the head of a PP, the second feature contains the number of intervening elements between the verb and the focus, a noun is the head of the phrase which is after the focus, and the focus is to the left of the noun. It also allows similar features for the POS of both the focus and the head of the PP chunks. When we have an NP as the head of a phrase, we extract a set of feature values from the sentence. The instance contains superfluous signs. In addition to the lexical and the local context information, we include superficial information. The frames are words or punctuation marks. Table 1: Results of chunking/chunking expert.

<table>
<thead>
<tr>
<th>Type</th>
<th>Results of chunking/chunking expert</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADVP</td>
<td>0.70</td>
</tr>
<tr>
<td>ADJP</td>
<td>0.96</td>
</tr>
<tr>
<td>ADVP</td>
<td>0.97</td>
</tr>
<tr>
<td>ADJP</td>
<td>0.96</td>
</tr>
<tr>
<td>ADVP</td>
<td>0.97</td>
</tr>
<tr>
<td>ADJP</td>
<td>0.96</td>
</tr>
<tr>
<td>ADVP</td>
<td>0.97</td>
</tr>
<tr>
<td>ADJP</td>
<td>0.96</td>
</tr>
</tbody>
</table>

As observed, the features were chosen by manual optimization and the third feature is the number of intervening elements between the verb and the noun.
By their position in the context, features can be differentiated into one of the two types. Features 1-10 are from the context words. Features 11-12 are from the POS. Feature 11 is the POS of the word or chunk. Feature 12 is the word or chunk itself.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Word/POS</th>
<th>Context 1-7</th>
<th>Context 8-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>POS</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Word/POS</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>POS</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Word/POS</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>POS</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Word/POS</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>POS</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>Word/POS</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>POS</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>Word/POS</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2: The first instance for the sentence in Figure 1. Features 1-2 are the features for an example sentence annotated with POS.
3.3 Cascaded Experiments

Recall the treebank list versus the classified list yields pairs given the prediction of the classifier. Grouping instances extracted directly from the treebank, this set contains predictions with a grammatical function label added. The second task is to determine whether the same verb-focus instance that was classified with a grammatical function label is also found in the test data. Then, we can then calculate precision and recall of the grammatical relations get: precision and recall of the grammatical relations.

The classification task was always that of finding grammatical relations. The amount of structure in the input data varied. Table 4 shows the results of the experiments. The amount of structure in the input data varied. Table 4 shows the results of the experiments. This restriction allowed, for example, for another verb/VP chunk between the verb and the focus, in case the focus precedes the verb, and no other verb in case the verb precedes the verb. Therefore, we ran a series of experiments.

The amount of structure in the input data varied. Table 4 shows the results of the experiments. The amount of structure in the input data varied. Table 4 shows the results of the experiments. This restriction allowed, for example, for another verb/VP chunk between the verb and the focus, in case the focus precedes the verb, and no other verb in case the verb precedes the verb. Therefore, we ran a series of experiments.
There are three ways how two cascaded modules can interact:

1. The first module can add information on which the later module can (partially) base its decisions. This is the case between the adversarial functions finder and the relations finder. The former adds an extra informative feature to the instances of the latter (Feature 16 in Table 3). Cf. column two of Table 4.

2. The first module can restrict the number of decisions to be made by the second one. This is the case in the combination of the chunking steps and the relations finder. Without the chunker, the relations finder would have to decide for every word, whether it is the head of a constituent that bears a relation to the verb. With the chunker, the relations finder has to make this decision for fewer words, namely only for those that are the last word in a chunk resp. the prepositional phrase of a PP chunk. Practically, this reduction of the number of decisions (which translates into a reduction of instances) as can be seen in the third column of Table 4.

3. The first module can reduce the number of elements used for the instances by counting one chunk as just one context element. We can see the effect in the feature that indicates the distance in elements between the focus and the verb. The more chunks are used, the smaller the average absolute distance (see column four of Table 4).

The PP chunker reduces the number of decisions for the relations finder (instead of one instance for the preposition and one for the NP chunk, we get only one instance for the PP chunk), introduces an extra feature (Feature 12 in Table 3), and changes the context (instead of a preposition and an NP, context can now be a preposition and a PP chunk). Without the chunker, the context can only consist of the PP chunk. We can see the effect in Table 4.

As we already noted above, precision and recall increase monotonically with the number of structure. However, we note large differences, such as NP chunks which increase F by more than 10% and PP chunks which increase F by 6%/8%.

For a detailed comparison of the two methods on the same task see (Daelemans et al., 1999a). That paper also shows that the chunker reduces the number of decisions to be made by the second module, without the chunker, the relations finder would have to decide for every word, whether it is the head of a constituent that bears a relation to the verb. Practically, this reduction of the number of decisions (which translates into a reduction of instances) as can be seen in the third column of Table 4.

The results show that the first module can reduce the number of decisions made by the second module.
5. Conclusion and Future Research

Table 5: Results of grammatical relation assignment with more and more structure in the test data added by earlier modules in the cascade. Columns show the number of features in the instances, the number of instances constructed from the test input, the average distance between the verb and the focus element, precision, recall, and \( F_{1} \) score over all relations and over some selected relations.

Table 4: Results of grammatical relation assignment with more and more structure in the test data added by earlier modules in the cascade. Columns show the number of features in the instances, the number of instances constructed from the test input, the average distance between the verb and the focus element, precision, recall, and \( F_{1} \) score over all relations and over some selected relations.

Table 3: Comparison of performance of several modules on realistic input (structurally enriched by previous modules in the cascade) vs. on "perfect" input (enriched with partial treebank annotation).

Table 2: Comparison of performance of several modules on realistic input (structurally enriched by previous modules in the cascade) vs. on "perfect" input (enriched with partial treebank annotation).

Table 1: Comparison of performance of several modules on realistic input (structurally enriched by previous modules in the cascade) vs. on "perfect" input (enriched with partial treebank annotation).

Table 5: Results of grammatical relation assignment with more and more structure in the test data added by earlier modules in the cascade. Columns show the number of features in the instances, the number of instances constructed from the test input, the average distance between the verb and the focus element, precision, recall, and \( F_{1} \) score over all relations and over some selected relations.

Table 4: Results of grammatical relation assignment with more and more structure in the test data added by earlier modules in the cascade. Columns show the number of features in the instances, the number of instances constructed from the test input, the average distance between the verb and the focus element, precision, recall, and \( F_{1} \) score over all relations and over some selected relations.

Table 3: Comparison of performance of several modules on realistic input (structurally enriched by previous modules in the cascade) vs. on "perfect" input (enriched with partial treebank annotation).

Table 2: Comparison of performance of several modules on realistic input (structurally enriched by previous modules in the cascade) vs. on "perfect" input (enriched with partial treebank annotation).

Table 1: Comparison of performance of several modules on realistic input (structurally enriched by previous modules in the cascade) vs. on "perfect" input (enriched with partial treebank annotation).
Adv. verbal function assignment will lead to better GR assignment.

Finally, since cascading proven effective for GR assignment we intend to study the effect of cascading different types of XP chunkers on chunking performance. We might, e.g., first find ADJP chunks, then use that chunker’s output as additional input for the NP chunker, then use the combined output as input to the VP chunker and so on. Other chunker orderings are possible, too. Likewise, it might be better to find different grammatical relations subsequently, instead of simultaneously.

References


