Statistical Parsing of Dutch using Maximum Entropy Models with Feature Merging

Tony Mullen, Rob Malouf and Gertjan van Noord
Alfa-Informatica, University of Groningen
Groningen, Netherlands
{mullen,malouf,vannoord}@let.rug.nl

Abstract

In this project report we describe work in statistical parsing using the maximum entropy technique and the Alpino language analysis system for Dutch. A major difficulty in this domain is the lack of sufficient corpus data available for training. Among other problems, this sparseness of data increases the danger of the model overfitting the training data, making it particularly important that the selection of statistical features upon which to base the model be optimal. To this end we have adapted the notion of *feature merging*, a means of constructing equivalence classes of statistical features based upon common elements within them. In spite of promising preliminary results, subsequent tests have not enabled us to conclude whether this approach helps the kind of models we are working with.

1 Introduction

Recent years have seen a considerable amount of research in the field of maximum entropy-based "log linear" modeling for disambiguation (Berger et al., 1996; Della Pietra et al., 1997; Johnson et al., 1999). This is in large part due to the fact that such models are superior to others in modeling linguistic phenomena which contain internal dependencies, since the log linear modeling framework allows weights to be assigned to features without assuming independence of the features.

An important issue in any area of statistical NLP is the problem of insufficient amounts of training data. The maximum entropy framework of statistical modeling is theoretically sound, but like most other modeling methods its efficacy depends upon having a sufficient amount of informative training data. For parsing, this data is sometimes available in the form of hand-parsed corpora, or "treebanks". Such data is often difficult to come by. Several large treebanks exist for English, but the resources available for other languages are significantly less developed. For this reason, it is imperative to get the most out of what is available.

In the current experiments the goal is to maximally exploit the small but informative training data set we have by taking a more sophisticated view of the feature set itself. To this end we employ the method of *feature merging*, a means of constructing equivalence classes of statistical features based upon common elements within them. This technique allows the model to retain information which would otherwise be discarded in a simple frequency-based feature cutoff by producing new, generalized features which serve as a variant of backed-off features. We discuss work done in Alpino, a new language analysis system for Dutch (Bouma et al., 2001).

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 et al., 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.

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 250 rules, defined in terms of a few general rule structures and principles. The grammar covers the basic constructions of Dutch as well as a number of more idiosyncratic constructions (see Bouma et al. (2001) for details). 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 150 verbal subcategorization types.

The lexicon contains about 100,000 entries. In addition, lexical analysis is extended by a number of heuristics to treat unknown words.

2.2 Robust Parsing

The construction of a dependency structure 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, 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.

Note that best-first parsing methods proposed for stochastic context-free grammars (e.g. Caraballo and Charniak (1998)) cannot be used in this context. In attribute-value grammars, it might easily happen that the locally most promising sub-trees cannot be extended to global parses because of conflicting feature constraints.

In (Bouma et al., 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 expensive to construct all parse trees, we have implemented an algorithm which computes parse trees from the parse forest as an approximate 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.

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 et al., 2000), using dependency structures similar to those used for the German Negra corpus (Skut et al., 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. Control relations are encoded by means of co-indexing. Note that a dependency structure does not necessarily reflect (surface) syntactic constituency.

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 2500 sentences of *Eindhoven-cdbl*.

3 Maximum entropy modeling and Alpino

The maximum entropy (maxent) technique is an approach to statistical modeling using *log linear* distributions. In this framework, events are considered as multiplicities of weighted *features*. In training, the features' weights are derived based upon the distribution of the features in the training data, using an iterative algorithm such as *improved iterative scaling* (IIS) (Della Pietra et al., 1997). Weights are chosen to maximize the entropy of the model while minimizing the divergence between the model and the training data. In employing the model, events of a given context are evaluated by summing the weights of their representative features and normalizing over the context to obtain a probability distribution, as in equation 1, where $p(y|x)$ represents the probability of event y given context x, and λ_i represents the weight for feature f_i. The value of each function f_i reflects the number of times the feature is active for a given event.

$$p(y|x) = \frac{1}{Z(x)} \exp\left[\sum_i \lambda_i f_i(x,y)\right] \tag{1}$$

and $Z(x)$ is the normalization factor:

$$Z(x) = \sum_y \exp\left[\sum_i \lambda_i f_i(x,y)\right] \quad (2)$$

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, we consider a context 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 2.2. Following Johnson et al. (1999), the best-first search proceeds on the basis of the unnormalized conditional probabilities derived from equation 1 for each possible subtree.

4 The Features and Feature Merging

The model depends on the distribution of the 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, we employ several types of features 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.

Examples of these features may be seen below, where example 1 is a rule for creating a VP, 2 contains a valency frame for the noun *mens*, 3 describes a dependency triple between the noun *mens* and the adjective *modern*, and the direction of the modification, and finally example 4 contains lexical information about the word *modern* as it occurs in context.

1 `vp_arg_v(np)`
2 `noun(de):mens:[mod]`
3 `noun:mens:mod:left:adjective:modern`
4 `moderne:modern:adjective(e,adv)`

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 we say 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.

A number of approaches have been taken to smoothing exponential models, including imposing a Gaussian prior over the distribution (Chen and Rosenfeld, 1999) and building the feature set up by a process of induction to ensure that only maximally representative features are admitted into the model (Della Pietra et al., 1997). The most commonly employed and computationally inexpensive approach to reducing noise is to use a frequency-based feature cutoff (Ratnaparkhi, 1998), in which features which occur fewer times in the training data than some predetermined cutoff are eliminated. This has shown to be an effective way to improve results. Because of its simplicity and effectiveness, it is the approach we have focused on improving on in the present research. 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, we experiment with a method of *feature merging* similar to that introduced in Mullen and Osborne (2000). This approach 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 each of the features before merging. Thus information in features which otherwise might have been lost in a cutoff is retained in the form of a more general feature.

4.2 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 work, lexical items, both sentence forms and word stems, are considered as elements. 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

feature 1 $= A : B$ with count 2
feature 2 $= A : C$ 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.

feature 3 $= A : X$ with count 6

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

4.3 Composition of 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 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.

4.4 Why feature merging?

It is well known that many models benefit from a frequency-based feature cutoff. Using feature merging, we seek to take a more sophisticated view of the features themselves, allowing 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, we hope to discard noisy information from the model with a greater degree of control, maintaining useful information contained by features which would otherwise be lost.

5 Experiments

Experiments were performed using the features described above with a training set of 1,220 sentences, whose parses totaled 626,699 training events, initially with a held-out set of 131 sentences and subsequently on a test set of unseen sentences totaling 566 sentences. A merging threshold of 500 and a feature cutoff of 500 were determined by use of the 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.

5.1 Evaluation

For each sentence in the test set, the dependency structure of the highest scoring parse was extracted, and compared to the gold standard dependency structure in the treebank (Carroll et al., 1998). For each sentence, we calculate the *overlap*, the number of relations for which the system agrees with the gold standard parse, and the number of errors:

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

Model	CA	Φ
baseline	63.01	0.00
maxent	77.45	56.99
maxent+cutoff	79.36	64.52
maxent+cutoff+merge	80.01	67.08
best possible	88.35	100.00

Table 1: Preliminary results on held-out data

Model	CA	Φ
baseline	58.92	0.00
maxent	71.53	45.77
maxent+cutoff	73.67	53.54
maxent+cutoff+merge	73.26	52.05
best possible	86.47	100.00

Table 2: Results on unseen data

From the error, we can compute the *concept accuracy* (Boros et al., 1996; van Zanten et al., 1999) as

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

The results given below for each model are in terms of average per-sentence concept accuracy.

A second, related, metric is employed as well, 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-sentence concept accuracy obtained by arbitrary selection of a parse) and the best possible per-sentence concept accuracy. This latter 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, which we refer to as the *phi* measure, is shown in equation 3:

$$\Phi = 100 \times \frac{\text{CA} - \text{baseline}}{\text{possible} - \text{baseline}} \tag{3}$$

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

Preliminary results on held-out data to establish thresholds, shown in table 1, were promising, suggesting that incorporation of merging with a merge threshold of 500 elements performed somewhat better than the best possible feature cutoff of 500 elements alone.

Unfortunately, these preliminary results were not supported by experiments on unseen data. As can be seen in table 2, the averaged results of four folds of ten-fold cross validation, representing a total of 566 sentences, show that the use of merging at the threshold determined by the held-out data does not appear to benefit the model.

6 Conclusion

Results so far are inconclusive. The effectiveness of the merging technique appears to depend greatly on qualities of the feature set itself. It is hoped that further experiments with different types of features will shed light on circumstances in which feature merging may be most effectively employed as a tool to optimize model performance.

References

Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22.

M. Boros, W. Eckert, F. Gallwitz, G. Görz, G. Hanrieder, and H. Niemann. 1996. Towards understanding spontaneous speech: Word accuracy vs. concept accuracy. In *Proc. ICSLP '96*, volume 2, pages 1009–1012, Philadelphia, PA.

Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2001. Wide coverage computational analysis of dutch. In *Computational Linguistics in the Netherlands 2000 (CLIN 2000). note = To appear. Available from* `http://www.let.rug.nl/~vannoord/`.

Sharon A. Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2).

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: A survey and a new proposal. In *Proceedings of the first International Conference on Language Resources and Evaluation (LREC)*, pages 447–454, Granada, Spain.

Stanley Chen and Ronald Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, Carnegie Mellon University, Pittsburg, February.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393.

P. C. Uit den Boogaart. 1975. *Woordfrequenties in geschreven en gesproken Nederlands*. Oosthoek, Scheltema & Holkema, Utrecht. Werkgroep Frequentie-onderzoek van het Nederlands.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 535–541, College Park, Maryland.

Michael Moortgat, Ineke Schuurman, and Ton van der Wouden. 2000. CGN syntactische annotatie. internal report Corpus Gesproken Nederlands.

Tony Mullen and Miles Osborne. 2000. Overfitting avoidance for stochastic modeling of attribute-valued grammars. In *Proceedings of the Fourth Conference on Computational Natural Language Learning*, pages 49–54, Lisbon.

Nelleke Oostdijk. 2000. The Spoken Dutch Corpus: Overview and first evaluation. In *Proceedings of Second International Conference on Language Resources and Evaluation (LREC)*, pages 887–894.

Carl Pollard and Ivan Sag. 1994. *Head-driven Phrase Structure Grammar*. University of Chicago / CSLI.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

Ivan Sag. 1997. English relative clause constructions. *Journal of Linguistics*, 33(2):431–484.

Wojciech Skut, Brigitte Krenn, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, DC.

Gertjan van Noord, Gosse Bouma, Rob Koeling, and Mark-Jan Nederhof. 1999. Robust grammatical analysis for spoken dialogue systems. *Journal of Natural Language Engineering*, 5(1):45–93.

Gertjan van Noord. 1997. An efficient implementation of the head corner parser. *Computational Linguistics*, 23(3):425–456. cmp-lg/9701004.

Gert Veldhuijzen van Zanten, Gosse Bouma, Khalil Sima'an, Gertjan van Noord, and Remko Bonnema. 1999. Evaluation of the NLP components of the OVIS2 spoken dialogue system. In Frank van Eynde, Ineke Schuurman, and Ness Schelkens, editors, *Computational Linguistics in the Netherlands 1998*, pages 213–229. Rodopi Amsterdam.