1 Introduction
In this paper I discuss a first attempt to create a
text chunker using a Maximum Entropy model.
The first experiments, implementing classifiers
that tag every word in a sentence with a phrase-
tag using very local lexical information, part-
of-speech tags and phrase tags of surrounding
words, give encouraging results.

2 Maximum Entropy models
Maximum Entropy (MaxEnt) models (Jaynes,
1957) are exponential models that implement
the intuition that if there is no evidence to
favour one alternative solution above another,
both alternatives should be equally likely. In
order to accomplish this, as much information
as possible about the process you want to model
must be collected. This information consists
of frequencies of events relevant to the process.
The frequencies of relevant events are consid-
ered to be properties of the process. When
building a model we have to constrain our at-
tention to models with these properties. In
most cases the process is only partially de-
scribed. The MaxEnt framework now demands
that from all the models that satisfy these con-
straints, we choose the model with the flattest
probability distribution. This is the model with
the highest entropy (given the fact that the con-
straints are met). When we are looking for a
conditional model $P(w|h)$, the MaxEnt solution
has the form:

$$ P(w|h) = \frac{1}{Z(h)} \cdot \sum_{i} \lambda_i f_i(h,w) $$

where $f_i(h,w)$ refers to a (binary valued) fea-
ture function that describes a certain event; $\lambda_i$
is a parameter that indicates how important fea-
ture $f_i$ is for the model and $Z(h)$ is a normali-
sation factor.

3 An MaxEnt chunker
3.1 Attributes used
First need to be decided which information
sources might help to predict the chunk tag. We
need to work with the information that is in-
cluded in the WSJ corpus, so the choice is first
limited to:

- Current word
- POS tag of current word
- Surrounding words
- POS tags of surrounding words

All these sources will be used, but in case of the
information sources using surrounding words we
will have to decide how much context is taken
into account. I did not perform exhaustive tests
on finding the best configuration, but following
(Tjong Kim Sang and Veenstra, 1999; Ratnaparkhi,
1997) I only used very local context. In
these experiments I used a left context of three
words and a right context of two words. Exper-
iments described in (Munoz et al., 1999) suc-
cessfully used larger contexts, but the few tests
that I performed to confirm this did not give
evidence that we could benefit significantly by
extending the context. Apart from information
given by the WSJ corpus, information generated by the model itself will also be used:

- Chunk tags of previous words

It would of course sometimes be desirable to use the chunk tags of the following words also, but these are not instantly available and therefore we will need a cascaded approach. I have experimented with a cascaded chunker, but I did not improve the results significantly.

In order to use previously predicted chunk tags, the evaluation part of the Maccent software had to be modified. The evaluation program needs a previously created file with all the attributes and the actual class, but the chunk tag of the previous two words cannot be provided beforehand as they are produced in the process of evaluation. A cascaded approach where after the first run the predicted tags are added to the file with test data is also not completely satisfactory as the provided tags are then predicted on basis of all the other attributes, but not the previous chunk tags. Ideally the information about the tags of the previous words would be added during evaluation. This required some modification of the evaluation script.

3.2 Results

The experiments are evaluated using the following standard evaluation measures:

- Tagging accuracy = \[
\frac{\text{Number of correct tagged words}}{\text{Total number of words}}
\]

- Recall = \[
\frac{\text{Number of correct proposed baseNP's}}{\text{Number of correct baseNP's}}
\]

- Precision = \[
\frac{\text{Number of correct proposed baseNP's}}{\text{Number of proposed baseNP's}}
\]

- \(F_\beta\)-score = \[
\frac{(\beta^2 + 1) \cdot \text{Recall} \cdot \text{Precision}}{\beta^2 \cdot \text{Recall} + \text{Precision}}
\] (\(\beta = 1\) in all experiments.)

In all the experiments a left context of 3 words and a right context of 2 words was used. The part of speech tags of the surrounding words and the word itself were all used as atomic features. The lexical information used consisted of the previous word, the current word and the next word. The word \(W_{-2}\) was omitted because it did not seem to improve the model. Using only these atomic features, the model scored an tagging accuracy of about 95.5\% and a F-score of about 90.5\%. Well below the reported results in the literature. Adding features combining POS tags improved the results significantly to just below state of the art scores. Finally 2 complex features involving NP chunk tags predicted for previous words were added. The most successful set of features used in our experiments is given in figure 1. It is not claimed that this is the best

| Template        | Meaning                           |
|-----------------|-----------------------------------|
| \(TAG_{-3}\)    | Base-NP tag of \(W_{-3}\)         |
| \(POS_{-3}\)    | Part of Speech tag of \(W_{-3}\)  |
| \(POS_{-3}/POS_0\) |                                |
| \(POS_{-3}/POS_{-2}\) |                       |
| \(TAG_{-3}/TAG_{-2}/TAG_{-1}/POS_0\) |                     |
| \(TAG_{-2}\)    |                                    |
| \(POS_{-2}\)    |                                    |
| \(W_{-1}\)      | Previous word                      |
| \(TAG_{-1}\)    |                                    |
| \(POS_{-1}\)    | Current word                        |
| \(W_0\)         |                                    |
| \(POS_0\)       |                                    |
| \(W_{+1}\)      |                                    |
| \(POS_{+1}\)    |                                    |
| \(POS_{-2}/POS_0\) |                                    |
| \(POS_{-2}/POS_{-1}\) |                           |
| \(TAG_{-2}/TAG_{-1}/POS_0\) |                     |
| \(POS_{-1}/TAG_{-1}/POS_0\) |                     |
| \(POS_{-1}/POS_0/POS_{+1}\) |                           |
| \(POS_{-2}/POS_{-1}/POS_0/POS_{+1}\) |                         |
| \(POS_0/POS_{+1}\) |                                    |
| \(POS_{+2}\)    |                                    |
| \(POS_0/POS_{+2}\) |                                    |
| \(POS_0/POS_{+1}/POS_{+2}\) |                     |

Figure 1: Feature set-up for best scoring experiment

set of features possible for this task. Trying new feature combinations, by adding them manually and testing the new configuration is a time consuming and not very interesting activity. Especially when the scores are close to the best published scores, adding new features have little impact on the behaviour of the model. An algorithm that discovers the interaction between features and suggests which features could be combined to improve the model would be very helpful here. I did not include any complex features involving lexical information. It might be
useful to include more features with lexical information if more training data is available (for example the full R&M data set consisting of section 2-21 of WSJ).

For feature selection a simple count cut-off was used. I experimented with several combinations of thresholds and the number of iterations used to train the model. When the threshold was set to 2, unique contexts (can be problematic during training of the model; see (Ratnaparkhi, 1998)) did not occur very frequently anymore and an upper bound on the number of iterations did not seem to be necessary. It was found that (using a threshold of 2 for every single feature) after about 100 iterations the model did not improve very much anymore. Using the feature setup given in figure 1 a threshold of 2 for all the features and allowing the model to train over 100 iterations, the scores given in table 1 were obtained.

4 Concluding remarks

The first observation that I would like to make here, is the fact that it was relatively easy to get results that are comparable with previously published results. Even though some improvement is to be expected when more detailed features, more context and/or more training data is used, it seems to be necessary to incorporate other sources of information to improve significantly on these results. Further, it is not satisfactory to find out what attribute combinations to use by trying new combinations and testing them. It might be worth to examine ways to automatically detect which feature combinations are promising (Mikheev, forthcoming; Della Pietra et al., 1997).

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1).

Luc Dehaspe. 1997. Maximum entropy modeling with clausal constraints. In *Proceedings of the 7th International Workshop on Inductive Logic Programming*.

S. Della Pietra, V. Della Pietra, and J. Lafferty. 1997. Inducing features from random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4).

| test data | precision | recall | \(F_{\beta=1}\) |
|-----------|-----------|--------|----------------|
| ADJP      | 65.53 %   | 75.33 % | 70.09          |
| ADVP      | 78.98 %   | 78.08 % | 78.53          |
| CONJP     | 55.56 %   | 45.45 % | 50.00          |
| INTJ      | 50.00 %   | 100.00 %| 66.67          |
| LST       | 0.00 %    | 0.00 %  | 0.00           |
| NP        | 93.18 %   | 92.84 % | 93.01          |
| PP        | 97.05 %   | 93.60 % | 95.30          |
| PRT       | 58.49 %   | 73.81 % | 65.26          |
| SBAR      | 63.36 %   | 82.68 % | 71.75          |
| VP        | 93.22 %   | 92.54 % | 92.88          |
| all       | 92.08 %   | 91.86 % | 91.97          |

Table 1: Results

E.T. Jaynes. 1957. Information theory and statistical mechanics. *Physical Review*, 108:171-190.

Andrei Mikheev. forthcoming. Feature lattices and maximum entropy models. *Journal of Machine Learning*.

Marcia Muñoz, Vasin Punyakanok, Dan Roth, and Dav Zimak. 1999. A learning approach to shallow parsing. In *Proceedings of EMNLP-WVLC’99*.

Adwait Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Brown University, Providence, Rhode Island.

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

Sven Eric Ristad. 1998. Maximum entropy modelling toolkit. Technical report.

Ronald Rosenfeld. 1994. *Adaptive Statistical Language Modelling: A Maximum Entropy Approach*. Ph.D. thesis, Carnegie Mellon University.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Strategy and tactics: a model for language production. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, University of Bergen, Bergen, Norway.