Journal of Machine Learning Research 2 (2002) 595-613 Submitted 9/01; Published 3/02

Shallow Parsing using Specialized HMMs

Antonio Molina AMOLINA@DSIC.UPV.ES
Ferran Pla FPLA@DSIC.UPV.ES
Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València
Camí de Vera s/n, 46020 València (Spain)

Editors: James Hammerton, Miles Osborne, Susan Armstrong and Walter Daelemans

Abstract

We present a unified technique to solve different shallow parsing tasks as a tagging problem using a Hidden Markov Model-based approach (HMM). This technique consists of the incorporation of the relevant information for each task into the models. To do this, the training corpus is transformed to take into account this information. In this way, no change is necessary for either the training or tagging process, so it allows for the use of a standard HMM approach. Taking into account this information, we construct a Specialized HMM which gives more complete contextual models. We have tested our system on chunking and clause identification tasks using different specialization criteria. The results obtained are in line with the results reported for most of the relevant state-of-the-art approaches.

Keywords: Shallow Parsing, Text Chunking, Clause Identification, Statistical Language Modeling, Specialized HMMs.

1. Introduction

Shallow parsing has become an interesting alternative to full parsing. The main goal of a shallow parser is to divide a text into segments which correspond to certain syntactic units. Although the detailed information from a full parse is lost, shallow parsing can be done on non-restricted texts in an efficient and reliable way. In addition, partial syntactical information can help to solve many natural language processing tasks, such as information extraction, text summarization, machine translation and spoken language understanding.

Shallow parsing involves several different tasks, such as text chunking, noun phrase chunking or clause identification. *Text chunking* consists of dividing an input text into non-overlapping segments. These segments are non-recursive, that is, they cannot include other segments and are usually called *chunks* as defined by Abney (1991). Noun phrase chunking (*NP chunking*) is a part of the text chunking task, which consists of detecting only noun phrase chunks. The aim of the *Clause identification* task is to detect the start and the end boundaries of each clause (sequence of words that contains a subject and a predicate) in a sentence. For example, the sentence *"You will start to see shows where viewers program the program"* would be chunked as follows:

(NP You) (VP will start to see) (NP shows) (ADVP where) (NP viewers) (VP program) (NP the program) .[1]

The clauses in the sentence would be:

(S You will start to see shows (S where (S viewers program the program)) .)

Chunks and clause information in a sentence can also be represented by means of tags. In Tjong Kim Sang et al. (2000), there are several equivalent chunk tag sets for representing chunking. The *IOB2* set, which was previously used by Ratnaparkhi (1998), uses three kinds of tags: B-X for the first word of a chunk of type X; I-X for a non-initial word in an X chunk; O for a word outside of any chunk. For clause identification, each word can be tagged with the corresponding brackets if the word starts and/or ends a clause, or with a null tag if the word is not the start or the end of a clause. The above example can be represented using this notation as follows:

You	B-NP	(S*
'll	B-VP	*
start	I-VP	*
to	I-VP	*
see	I-VP	*
shows	B-NP	*
where	B-ADVP	(S*
viewers	B-NP	(S*
program	B-VP	*
the	B-NP	*
program	I-NP	*S)S)
.	O	*S)

Earlier approaches to solving this problem consisted of parsers which are based on a grammar of hand-coded rules. Abney (1996) developed the incremental partial parser *CASS* based on finite state methods for detecting chunks and clauses. Aït-Mokhtar and Chanod (1997) also built an incremental architecture of finite-state transducers that identifies chunks and detects subjects and objects. Voutilainen (1993) used a different formalism (constraint grammars) to detect NPs.

In the literature, you can find different learning methods which have been applied to perform shallow parsing: Transformation-based Learning, Memory-based Learning, Hidden Markov Models, Maximum Entropy, Support Vector Machines, etc. The first works focused mainly on NP detection. Ramshaw and Marcus (1995) used Transformation-based learning and put forward a standard training and testing data set, that has later been used to contrast other approaches. Memory-based learning was used by Daelemans et al. (1999) and Argamon et al. (1998). The main reference for text chunking is the shared task for CoNLL-2000[2] (Tjong Kim Sang and Buchholz, 2000). In Section 4, we will briefly review the different approaches to text chunking presented in this shared task.

Learning approaches for clause identification have recently been developed. Orasan (2000) applied memory-based learning techniques and corrected the output by applying

1. In the example NP is Noun Phrase, VP is Verbal Phrase, ADVP is Adverbial Phrase and S is a clause.
2. Chunking shared task is available at `http://lcg-www.uia.ac.be/conll2000/chunking/`

some rules. In the shared task for CoNLL-2001[3] (Tjong Kim Sang and Déjean, 2001) other approaches were presented (Hidden Markov Models, Memory-based Learning, Boosting, etc.)

In this paper, we present a unified technique to construct Specialized HMMs to be used for solving shallow parsing tasks. First, in Section 2, we formalize shallow parsing as a tagging problem and define the method for the specialization of the models. In Section 3, we present the results of the application of our approach to text chunking. We test different specialization criteria under the same experimental conditions of the CoNLL-2000 shared task. We achieved state-of-the-art performance, as we show in Section 4. We also apply this technique to solve the clause identification problem in Section 5. Finally, we present some concluding remarks.

2. Shallow parsing as HMM-based tagging

We consider shallow parsing to be a tagging problem. From the statistical point of view, tagging can be solved as a maximization problem.

Let $\mathcal{O}$ be a set of output tags and $\mathcal{I}$ the input vocabulary of the application. Given an input sentence $I = i_1, \ldots, i_T$, where $i_j \in \mathcal{I} : \forall j$, the process consists of finding the sequence of states of maximum probability on the model. That is, the sequence of output tags, $O = o_1, \ldots, o_T$, where $o_j \in \mathcal{O} : \forall j$. This process can be formalized as follows:

$$\begin{aligned}\widehat{O} &= \arg\max_{O} P(O|I) \\ &= \arg\max_{O} \left(\frac{P(O) \cdot P(I|O)}{P(I)} \right); \, O \in \mathcal{O}^T \end{aligned} \tag{1}$$

Due to the fact that this maximization process is independent of the input sequence, and taking into account the Markov assumptions, the problem is reduced to solving the following equation (for a second–order HMM):

$$\arg\max_{o_1 \ldots o_T} \left(\prod_{j:1 \ldots T} P(o_j | o_{j-1}, o_{j-2}) \cdot P(i_j | o_j) \right) \tag{2}$$

The parameters of equation 2 can be represented as a second-order HMM whose states correspond to a tag pair. Contextual probabilities, $P(o_j|o_{j-1}, o_{j-2})$, represent the transition probabilities between states and $P(i_j|o_j)$ represents the output probabilities.

This formalism has been widely used to efficiently solve part-of-speech (POS) tagging in (Church, 1988, Merialdo, 1994, Brants, 2000), etc. In POS tagging, the input vocabulary is composed of words and the output tags are POS or morphosyntactic tags. The segmentation produced by some different shallow parsing tasks, such as text chunking or clause identification, can be represented as a sequence of tags as we mentioned above. Therefore, these problems can also be carried out in a way similar to POS tagging. However, to successfully solve each task, it is necessary to deal with certain peculiarities.

On the one hand, you have to decide which available input information is really relevant to the task. POS tagging considers only *words* in the input. In contrast, chunking can

3. Clause identification shared task is available at `http://lcg-www.uia.ac.be/conll2001/clauses/`

$\mathcal{T}$			$\xrightarrow{f_s}$ $\widetilde{\mathcal{T}}$	
I		O	$\widetilde{I}$	$\widetilde{O}$
You	PRP	B-NP	PRP	PRP·B-NP
will	MD	B-VP	MD	MD·B-VP
start	VB	I-VP	VB	VB·I-VP
to	TO	I-VP	TO	TO·I-VP
see	VB	I-VP	VB	VB·I-VP
shows	NNS	B-NP	NNS	NNS·B-NP
where	WRB	B-ADVP	where·WRB	where·WRB·B-ADVP
viewers	NNS	B-NP	NNS	NNS·B-NP
program	VBP	B-VP	VBP	VBP·B-VP
the	DT	B-NP	DT	DT·B-NP
program	NN	I-NP	NN	NN·I-NP
.	.	O	.	.·O

Figure 1: Example of the result of applying specialization on a sentence.

consider *words* and *POS tags*, and clause identification can take into account *words*, *POS tags* and *chunk tags*. In this case, if all this input information is considered, the input vocabulary of the application could become very large, and the model would be poorly estimated.

On the other hand, the output tag set could be too generic to produce accurate models. The contextual model can be enriched by considering a more fine-grained output tag set by adding some kind of information to the output tags. For instance, in the chunking task we have enriched the chunk tags by adding to them POS information and selected words as we will show below. This aspect has also been tackled in POS tagging (Kim et al., 1999, Lee et al., 2000, Pla and Molina, 2001), by lexicalizing the models, that is, by incorporating words into the contextual model.

In this work, we propose a simple technique that permits us to encode the available information into the model, without changing the learning or the tagging processes. This method consists of modifying the original training data set in order to consider only the relevant input information and to extend the output tags with additional information.

This transformation is the result of applying a *specialization* function f_s on the original training set $\mathcal{T}$ to produce a new one $\widetilde{\mathcal{T}}$, that is:

$$f_s : \mathcal{T} \subset (\mathcal{I} \times \mathcal{O})^* \rightarrow \widetilde{\mathcal{T}} \subset (\widetilde{\mathcal{I}} \times \widetilde{\mathcal{O}})^*$$

This function transforms every training tuple $\langle i_j, o_j \rangle$ to a new tuple $\langle \widetilde{i}_j, \widetilde{o}_j \rangle$, and thus the original input and output sets to the new sets $\widetilde{\mathcal{I}}$ and $\widetilde{\mathcal{O}}$, by concatenating the selected information. This function has to be experimentally defined for each task, as we will discuss in Section 3.

Figure 1 shows an example of the application of this function on a sample of the training set used in the chunking task. In this example, we have considered POS tags and certain selected words as relevant input information. The output tags have also been enriched with this information. For example, the tuple ⟨You·PRP, B-NP⟩ is transformed to the new tuple ⟨PRP, PRP·B-NP⟩, considering only POS information. On the other hand, the tuple ⟨where·WRB, B-ADVP⟩, considering also lexical information, is transformed to the new tuple ⟨where·WRB, where·WRB·B-ADVP⟩.

From this new training set $\widetilde{\mathcal{T}}$, we can learn the Specialized HMM by maximum likelihood in the usual way. The tagging process is carried out by Dynamic Programming Decoding using the Viterbi algorithm. This decoding process is not modified, you simply consider the decisions taken into account in the specialization process. That is, to consider the relevant information as input and to map the sequence of output tags (which belongs to $\widetilde{\mathcal{O}}$) to the original output tags (which belongs to $\mathcal{O}$). This can be carried out in a direct way.

3. Chunking evaluation

We present a set of experiments in order to evaluate the chunking approach proposed in this work. We focus on the different specialization criteria considered to construct the specialized HMM. As we mentioned above, one of the advantages of our approach is that no change is needed for either the training or the decoding processes carried out when specialized HMMs are used. To confirm this, all the experimental work was conducted using the TnT[4] tagger developed by Brants (2000) without making any modification to its source code.

TnT is a very efficient statistical POS tagger based on HMMs. To deal with sparse problems, it uses linear interpolation as a smoothing technique to estimate the model. To handle unknown words, it uses a probabilistic method based on the analysis of the suffix of the words.[5] All the following experiments were done with TnT's default options using second-order HMMs.

We used the data defined in the shared task of CoNLL-2000. The characteristics of this task were described by Tjong Kim Sang and Buchholz (2000). It used the same Wall Street Journal corpus sections defined by Ramshaw and Marcus (1995), that is, sections 15-18 for training, and section 20 for testing. The set of chunks (NP, VP, ADVP, ADJP, PP, SBAR, CONJP, PRT, INTJ, LST, UCP) was derived from the full parsing taking into account certain assumptions and simplifications.[6] The POS tagging was obtained using the Brill tagger (Brill, 1995) without correcting the tagger output. We also compared our results to those obtained by the other approaches that participated in the shared task (see Section 4).

As we stated in Section 2, two decisions must be made in order to define a specialization function on the training set: which information from the available input is relevant and how to refine the output tags.

These decisions were tested experimentally. We used a development set (which was different from the test set) in order to select this information. We divided the original training set into two partitions: 90% for training and 10% for tuning (development set). To do that, we used nine consecutive sentences from the original training set for training

4. TnT is available at `http://www.coli.uni-sb.de/thorsten/tnt`
5. In our case, the suffix method is not the most suitable for handling unknown words because the input vocabulary is reduced to POS tags or concatenations of POS tags and words. In all the experiments performed, the vocabulary was always seen in the training set. Therefore, we did not consider necessary to study this problem.
6. The script to derive the chunks is available at `http://ilk.kub.nl/~sabine/chunklink/`

and we used the tenth sentence for testing. We tested different combinations of input and output information on the development set and we selected those that improved the results obtained with no specialized training data set.

The baseline system, which we called **BASIC**, considered training tuples such as $\langle p_i, ch_i \rangle$, that is, only POS tags (p_i) were taken into account as input vocabulary, and no changes were made in the chunk tag set (ch_i). This criterion gave a very poor performance, because the output tag set was too generic to construct accurate models. Therefore, we defined a specialization function f_s in order to extend the output vocabulary as follows:

$$f_s(\langle w_i \cdot p_i, ch_i \rangle) = \langle p_i, p_i \cdot ch_i \rangle$$

This criterion, denoted as **SP**, makes training tuples $\langle p_i, p_i \cdot ch_i \rangle$ in which only POS tags are considered as input and the output tags are enriched with the POS tag associated with the input words (w_i).

Finally, the **SP** criterion was tested by adding lexical information in both the input and the output. The problem was that adding all the words to the input and/or the output produced very large models and no improvements were observed. For this reason, we tested a selective lexicalization of the model. That is, only a set of certain relevant words ($\mathcal{W}_s$) were considered in the contextual language model. In this respect, we defined the following specialization function:

$$f_{s-Lex}(\langle w_i \cdot p_i, ch_i \rangle) = \begin{cases} \langle w_i \cdot p_i, w_i \cdot p_i \cdot ch_i \rangle \text{ if } w_i \in \mathcal{W}_s \\ \langle p_i, p_i \cdot ch_i \rangle \text{ if } w_i \notin \mathcal{W}_s \end{cases}$$

The main problem of this lexicalization is to define the set of words $\mathcal{W}_s$ to be considered. We defined some criteria in order to automatically extract the relevant words that improved the performance on the development set. These criteria are summarized below.

- **Lex-WCC** selects the words from the training set that belong to closed categories.[7]
- **Lex-WHF** selects the words whose frequency in the training set was higher than a certain *threshold*. In order to determine which threshold maximized the performance of the model (that is, the best set of words to specialize the model), we tuned it on the development partition with word sets of different sizes. The best performance was obtained by selecting the words whose frequency was higher than 100.
- **Lex-WTE** selects the words whose chunk tagging error rate was higher than a certain *threshold*. These words were extracted from the output provided by the tagger that uses the **SP** model. The best *threshold* obtained corresponds to the words whose error frequency was higher than 2 in the development set.
- **Lex-WCH** selects the words that belong to certain chunks such as SBAR, PP and VP with high frequency in the training set.
- **Lex-COM** selects the words corresponding to a combination of **Lex-WTE** and **Lex-WCH** criteria.

specialization criteria	precision	recall	$F_{\beta=1}$	$\lvert\widetilde{\mathcal{O}}\rvert$	$\lvert\mathcal{W}_s\rvert$
BASIC	84.16%	84.52%	84.34	22	0
SP	90.44%	89.96%	90.20	317	0
SP + Lex-WCC	91.99%	91.56%	91.77	830	154
SP + Lex-WHF	91.87%	92.14%	92.00	1,086	144
SP + Lex-WTE	92.22%	92.00%	92.11	592	38
SP + Lex-WCH	92.03%	92.25%	92.14	1,305	217
SP + Lex-COM	92.10%	92.35%	92.23	1,341	225

Table 1: Overall chunking results on the development data set using Specialized HMMs with different specialization criteria.

Table 1 shows the results of the tuning process on the development data set measured in terms of precision, recall and F_β rate. In addition, it shows the size of the output tag set ($\lvert\widetilde{\mathcal{O}}\rvert$) and the size of the selected word set ($\lvert\mathcal{W}_s\rvert$). It can be observed that all the specializations considered outperformed the **BASIC** model. Although each lexicalization criterion used a different set of selected words with a size that ranked between 38 and 225 words, **SP + Lex-WHF**, **SP + Lex-WTE** and **SP + Lex-WCH** criteria achieved similar results while **SP + Lex-WCC** criterion achieved somewhat worse results. We obtained the best performance with the combination criterion **SP + Lex-COM**, with a F_β improvement of 9.4% with respect to the **BASIC** model. The improvement of the **SP** model with respect to the **BASIC** model was about 7% and the use of lexicalization criteria incremented F_β about 2% with respect to the **SP** model.

We think that these statistical criteria can be improved by means of a linguistic study to determine which words are really relevant to this disambiguation problem. The experiments conducted have given us some clues about this, because we have observed that the effect of some words on the overall result is not significant. For example, the **SP + Lex-WTE** criterion which only provide a small set of 38 specialized words performed better than other criteria.

Once the system was tested on the development data set, new models were learnt using the original training data set (sections 15-18) with the best specialization parameters obtained in the tuning process. Next, the system was tested on a new unseen data set (section 20). Table 2 shows that the system had a similar behaviour for both development and test data sets. The best performance was also obtained by using the **SP + Lex-COM** criterion, which suggests that a model that includes these specialization parameters could be successfully applied to other unseen data.

The results for precision, recall and F_β rate for all the chunks considered using the **SP + Lex-COM** criterion are summarized in Table 3. These results outperformed the F_β rate of **BASIC** and **SP** for each chunk. The details of this improvement are shown in Figure 2. The highest improvement was achieved for SBAR and PRT chunks. This is

7. The closed categories considered are: *CC, DT, MD, POS, PP\$, RP, TO, WDT, WP\$, EX, IN, PDT, PRP, WP, WRB.*

specialization criteria	precision	recall	$F_{\beta=1}$	$\lvert\tilde{\mathcal{O}}\rvert$	$\lvert\mathcal{W}_s\rvert$
BASIC	84.31%	84.35%	84.33	22	0
SP	89.58%	89.55%	89.57	320	0
SP + Lex-WCC	91.50%	91.51%	91.51	846	154
SP + Lex-WHF	91.30%	91.76%	91.53	1,105	144
SP + Lex-WTE	91.65%	91.82%	91.73	601	38
SP + Lex-WCH	91.74%	92.12%	91.93	1,339	217
SP + Lex-COM	91.96%	92.41%	92.19	1,381	225

Table 2: Overall chunking results on the shared task using Specialized HMMs with different specialization criteria.

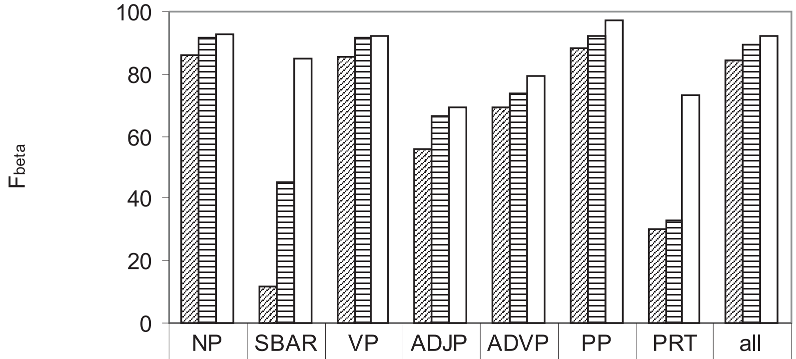
chunk	precision	recall	$F_{\beta=1}$
ADJP	71.19%	67.12%	69.10
ADVP	79.84%	79.10%	79.47
CONJP	38.46%	55.56%	45.45
INTJ	50.00%	50.00%	50.00
NP	92.30%	92.68%	92.49
PP	96.58%	97.40%	96.99
PRT	71.43%	75.47%	73.39
SBAR	85.50%	84.86%	85.18
VP	91.73%	92.81%	92.26
all	91.96%	92.41%	92.19

Table 3: Chunking results on the test set of the shared task using Specialized HMMs with the **SP + Lex-COM** criterion.

because the set of selected words that we considered included words that usually appear in these chunks.

Due to the fact that the number of parameters of the models increases, these models will be better estimated if the training data set is larger. To confirm this, we conducted an experiment increasing the size of the training data set. We chose training data from sections 00 to 19 of the WSJ corpus; the test data set was again section 20 and we used the same word set of the shared task ($|\mathcal{W}_s| = 225$) obtained using the **SP + Lex-COM** criterion to specialize the model. Figure 3 shows that F_β improves as the size of the training set increases, achieving a F_β rate of 93.25 with a training data set size of about 950,000 words and 1,960 output tags.

In Table 4 the results for each chunk using this large training data set are summarized. It can also be observed that all the F_β rates outperformed the results achieved with the small training data set. Although the overall F_β improvement is only about 1%, the best



	NP	SBAR	VP	ADJP	ADVP	PP	PRT	all
BASIC	86.09	11.46	85.66	55.91	69.55	88.37	30.14	84.33
SP	91.56	45.00	91.58	66.35	73.51	91.90	32.93	89.57
SP+Lex-COM	92.49	85.18	92.26	69.10	79.47	96.99	73.39	92.19

Figure 2: Improvement on F_β rate for certain chunks on the shared task using Specialized HMMs.

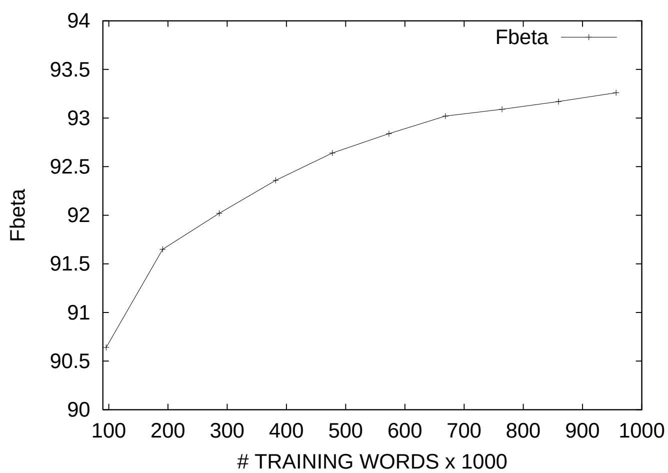


Figure 3: Evolution of the F_β rate using training sets of different size.

improvements were achieved for those chunks that include selected words (6.8% for PRT, 7.9% for ADJP and 3.8% for SBAR).

Finally, we would like to note that the efficiency of the system is not reduced even if $|\widetilde{\mathcal{O}}|$ increases when specialized models are used. For the model learnt with the **SP + Lex-COM** criterion the tagging speed holds around 30,000 words/second running on a Pentium

chunk	precision	recall	$F_{\beta=1}$
ADJP	78.54%	71.00%	74.58
ADVP	81.85%	79.68%	80.75
CONJP	40.00%	66.67%	50.00
INTJ	50.00%	50.00%	50.00
NP	93.52%	93.43%	93.48
PP	96.77%	97.73%	97.25
PRT	75.00%	82.08%	78.38
SBAR	88.70%	88.04%	88.37
VP	93.33%	93.73%	93.53
all	93.25%	93.24%	93.25

Table 4: Chunking results, on the test of the shared task with large training data set, using Specialized HMMs with **SP + Lex-COM** criterion.

500 Mhz. Although we have not made a comparative study of the efficiency with other approaches, we think that these performances are difficult to overcome by other systems.

4. Comparison with other chunking approaches

We compared our results with those systems that have presented results for the same training and test data used in the chunking shared task performed in CoNLL-2000. Each of these systems uses a specific learning method and different kinds of information. Following, we will briefly review each of these approaches. Finally, we compare the different kinds of information used for the different learning methods. Other comparisons, such as an efficiency comparison, cannot be done because the authors did not usually report it. Basically, these systems can be divided into the following groups: Rule-based systems, Memory-based systems, Statistical systems and Combined Systems.

Rule-based systems

The ALLiS system (Déjean, 2000) is based on theory refinement. It attempts to improve a previously learnt grammar using "contextualization" and "lexicalization" operators. The method only takes context into account if the confidence value for a certain POS tag is under a certain threshold. It calculates the ratio between the number of occurrences of a POS tag in a chunk and the number of occurrences of this POS tag in the training corpora. If this ratio is higher than a certain threshold, then the corresponding chunk tag is assigned. If not, it takes into account left and right context ("contextualization") and the current word ("lexicalization").

The system presented by Johansson (2000) picks the most likely chunk tag for a given context, assuming that a larger context (if it has been seen in the training data) overrides the label proposed by a smaller context. It obtains the best results for 5-context, that is, a context of the current POS tag, the two left POS tags and the two right POS tags.

Memory-based systems

Veenstra and Van den Bosch (2000) studied how the memory-based learning algorithm, implemented in TiMBL software, performs with different settings. Memory-based learning consists of storing the instances seen during learning in memory along with the corresponding categories. A new instance can be classified in a category by computing the distance between the new instance and the stored instances. The best results are obtained using IB1-IG algorithm (Daelemans et al., 1997) and applying the modified value difference metric to POS features.

Statistical systems

Osborne (2000) used Ratnaparkhi's Maximum Entropy-based tagger (Ratnaparkhi, 1996) to perform chunking. To do that, the input to the tagger is redefined to be a concatenation ("configuration") of the different contexts that are useful for chunking. It performs chunking in two steps. First, it guesses a default chunk tag by applying a model which has been learnt with configurations which consists of current POS tags and words. Second, it produces the definitive chunk tags by applying a model learnt with configurations which also takes into account the chunk tags previously guessed. Little improvement is achieved by incorporating suffix and prefix information to the configurations.

The approach of Koeling (2000) also builds a ME model which takes into account several individual features and complex features combining POS tags and chunk tags.

In our previous work (Pla et al., 2000b), we used a two level first-order HMM that performed tagging and chunking at the same time. The model was also refined by lexicalization to improve its performance.

Zhou, Su, and Tey (2000) incorporated contextual information into a bigram model by means of defining structured tags as input. These tags are composed of the current word (if it belongs to a certain category), the current POS tag, the previous POS tag, the descriptor of the phrase category and a structural relation tag that indicates if two adjacent words have the same parent. The model is refined by an error-learning technique that keeps only the words whose error rate decreases when they are incorporated into the model. Finally, a memory-based sequence learning is applied to incorporate chunk pattern probabilities achieving slight improvement.

Combined systems

These systems combine the output of different classifiers in order to improve the chunking performance.

Kudo and Matsumoto (2000) combined several Support Vector Machine classifiers. SVMs are a very suitable learning approach for solving two-class pattern recognition problems. Basically, SVMs are binary classifiers that can guess whether an instance (a token) belongs to a class or not. Pairwise classification is used to solve chunking (which is a multi-class task). This consists of training a classifier for each pair of different chunk tags (for K chunk tags, K*(K-1)/2 classifiers have to be learnt). The results of all the classifiers are combined by a dynamic programming algorithm. Later Kudo and Matsumoto (2001)

System	Method	w	w_{left}	w_{right}	p	p_{left}	p_{right}	c_{left}	F_β
[KM01]	SVM(Comb)	x	2	2	x	2	2	2	93.91
[ZDJ01]	Winnow	x	2	2	x	2	2	2	93.51
[Hal00]	WPDV(Comb)	x	1-5	1-5	x	3-5	3	2	93.32
[LR01]	Winnow								93.02
[TKS00]	MBL(Comb)	x	4	4	x	4	4		92.50
SP+Lex-COM	HMM	x	2		x	2		2	92.19
[ZST00]	HMM + MBL	x	1		x	1		1	92.12
[Dej00]	Rule-based	x			x	1	1		92.09
[Koe00]	ME	x	1	1	x	3	2	3	91.97
[Osb00]	ME	x	2	2	x	2	2	2	91.94
[VB00]	MBL	x	5	3	x	5	3		91.54
[PMP00]	HMM	x	1		x	1		1	90.14
[Joh00]	Rule-based				x	0-3	0-3		87.23

Table 5: F_β results of the different shallow parsing systems related and a comparison of the information used by them in the learning process.

introduced a weighted voting technique that improved the results on the same training and test data.

By majority voting, Tjong Kim Sang (2000) combined the results provided by five different memory-based learning classifiers (one classifier for each different chunk representation, that is, IOB1, IOB2, IOE1, IOE2 and C+O). In all cases, the combined system outperformed the individual systems. The system performs the task in two steps. First, it identifies chunk boundaries and second, it assigns the chunk tag. In the first step it considers POS and words as features and, in the second step, it adds the context of chunk tags guessed in the first phase.

Van Halteren (2000) presented another combined system that uses a more sophisticated combining technique called Weighted Probability Distribution Voting (WPDV). It combines the output of five classifiers: a memory-based learning classifier and four different WPDV classifiers. Due to the fact that output can present some systematic errors, these are also corrected using a WPDV model for each kind of error.

Recently, other systems based on the Winnow algorithm have been applied to chunking on the same data set. These systems learn different classifiers. Each predicts the start or the end of a kind of chunk. The output of these classifiers is combined in order to chunk the sentence satisfying some constraints such as non-overlapping constraints. The SNoW architecture, which is based on the Winnow algorithm, is used by Li and Roth (2001). The algorithm was modified by Zhang, Damerau, and Johnson (2001) to guarantee its convergence for linearly non-separable data and was successfully applied to text chunking.

Comparison

In Table 5 we have summarized the features that each approach takes into account and the F_β result reported. The table indicates whether a model uses some of the following features: the current word (w), current POS tag (p), the words to the left (w_{left}) and to the right (w_{right}), the POS tags to the left (p_{left}) and to the right (p_{right}), and the chunk tag (c_{left}) to the left. In addition, Osborne (2000) considers prefix and suffix word information, and the current chunk; Koeling (2000) also incorporates complex features by concatenating individual features; Zhou et al. (2000) include structural relations between words, the descriptor of the phrase category, and takes into account only certain words; Tjong Kim Sang (2000) also considers a context of two left and two right chunk tags guessed in a first level. Zhang et al. (2001) also include second-order features, such as POS-POS, chunk-chunk, word-POS and chunk-POS. Li and Roth (2001) do not report feature information.

It can be seen that combined systems perform better than the individual systems (only Winnow-based systems outperform some of the combined systems). There are six systems that produce very similar results (between 91.5% and 92.2%). The results of our system (Specialized HMMs) are slightly better than these individual classifiers. Only the Winnow-based systems perform better than Specialized HMMs. One conclusion that we can draw from Table 5 is that HMM approaches perform better than other systems and require encoding less information. This shows the importance not only of the feature selection but the importance of algorithm itself. It can be considered that the dynamic programming decoding algorithm used by HMM systems to solve the maximization equation implicitly takes into account information of the whole sentence.

We have found two approaches that use a technique similar to the one we describe in this paper: the Osborne (2000) Maximum Entropy approach and the HMM used by Zhou, Su, and Tey (2000). Like Specialized HMMs, both approaches consider structural tags or concatenations as input to the system. The differences lie in the underlying model used and the information which is taken into account in the input. Both approaches need to encode more information than Specialized HMMs, but they do not achieve better results: Osborne achieved a similar result (F_β=91.94), but Zhou et al. obtained lower results (F_β=89.57) when error correcting and memory-based techniques were not applied. Moreover, Osborne needed to learn two models that have to be applied sequentially: a first model that proposes an initial chunk tag, and a second one that takes into account the information provided by the first one.

Note that the performance of our preliminary system (Pla et al., 2000b) was very poor because it used bigram models instead of trigram models. In addition, it only took words as input in order to perform tagging and chunking at the same time, which decreased the performance as we reported in (Pla et al., 2000a).

5. Clause identification

Clause identification can be carried out in a way similar to text chunking, using the specialization technique previously presented. As in text chunking, the success of the method lies in the definition of the specialization function, that is, in an appropriate selection of the input information and the output tag set.

All the following experiments were conducted under the same conditions defined at the clause-splitting shared task of CoNLL-2001 (Tjong Kim Sang and Déjean, 2001). At this shared task, clause detection was divided into three parts: clause start-boundary detection, clause end-boundary detection and embedded clause detection. Here, we only report results

$\mathcal{T}$				$\xrightarrow{f_s}$ $\widetilde{\mathcal{T}}$	
I			O	$\widetilde{I}$	$\widetilde{O}$
You	PRP	B-NP	(S*	PRP·B-NP	PRP·(S1*
will	MD	B-VP	*	MD·B-VP	MD·*1
start	VB	I-VP	*	VB·I-VP	VB·*1
to	TO	I-VP	*	TO·I-VP	TO·*1
see	VB	I-VP	*	VB·I-VP	VB·*1
shows	NNS	B-NP	*	NNS·B-NP	NNS·*1
where	WRB	B-ADVP	(S*	WRB·B-ADVP	WRB·(S2*
viewers	NNS	B-NP	(S*	NNS·B-NP	NNS·(S3*
program	VBP	B-VP	*	VBP·B-VP	VBP·*3
the	DT	B-NP	*	DT·B-NP	DT·*3
program	NN	I-NP	*S)S)	NN·NN	NN·*S3)S2)
.	.	O	*S)	.·O	O·*1S)

Figure 4: Example of the result of applying specialization on a sample of the training set used in clause-splitting.

on the third part of the task (the rest of the results can be consulted in (Molina and Pla, 2001)). This shared task defined sections 15-18 from the WSJ corpus as training data set, section 20 as development set and section 21 as test set.

Clause-splitting can be seen as a step after chunking. Therefore, the available input information consists of words, POS tags and chunk tags. The output information are the clause tags. We performed clause-splitting in two phases: first, we found the best segmentation in clauses for the input sentence using a Specialized HMM; second, we corrected some balancing inconsistencies observed in the output by applying some rules.

In this case, we considered that the relevant input information for clause-detection was formed by POS tags and chunk tags. In order to produce more accurate models, the output tags were enriched with the POS tag. To avoid incorrectly balanced clauses in the output, we also added the number corresponding to the depth level of the clause in the sentence to the output tags. Thus, the specialization function f_s was defined as follows:

$$f_s(\langle w_i \cdot p_i \cdot ch_i, s_i \rangle) = \langle p_i \cdot ch_i, p_i \cdot s_i^{'} \rangle$$

where s_i is a clause tag and $s_i^{'}$ is the enumerated clause tag (each bracket in the clause tag is enumerated with the depth level of the corresponding clause). The effect of the application of this function (**SP** criterion) on a sentence can be seen in Figure 4.

Due to the fact that models have to be smoothed to guarantee a complete coverage of the language, this does not assure the correct balancing of the output. Therefore, we applied the following correcting rules to repair the inconsistencies in the output:

1. If the clause segmentation presents more *start* than *end* boundaries, we add the *end* boundaries that are needed to the last word in the sentence (just before the dot).

system	precision	recall	$F_{\beta=1}$
Carreras and Màrquez	84.82%	73.28%	78.63
SP+Lex-WHF	70.89%	65.57%	68.12
Tjong Kim Sang	76.91%	60.61%	67.79
SP	69.62%	64.17%	66.79
Patrick and Goyal	73.75%	60.00%	66.17
Déjean	72.56%	54.55%	62.77
Hammerton	55.81%	45.99%	50.42

Table 6: Clause-splitting results of the different systems presented in the shared task of CoNLL-2001.

2. If the clause segmentation presents more *end* than *start* boundaries, we add the *start* boundaries that are needed to the first word in the sentence.

3. If the sentence does not start with a *start* boundary or does not finish with an *end* boundary, we add these *start* and *end* tags.

An alternative solution would be to incorporate these rules into the model, but to do this it would be necessary to modify learning, decoding and smoothing processes, which is far from this preliminary approach to clause-detection.

Finally, we also tested the different specialization criteria obtaining slight improvements. The best results were achieved by incorporating some of the most frequent words into the model (**SP+Lex-WHF** criterion). These results, which can be seen in Table 6, are in line with those presented in the clause-splitting shared task of CoNLL-2001 by other systems (Tjong Kim Sang and Déjean, 2001). Our system achieved a performance which was slightly better than others based on boosting algorithms, neuronal networks, symbolic methods or memory-based learning. The best system used Ada-Boost learning combined with decision trees. In addition, the set of features was adapted to the task following linguistic criteria.

6. Concluding remarks

In this work, we have presented a technique that allows us to tackle different natural language disambiguation tasks as tagging problems. In particular, we have addressed the shallow parsing and the clause identification problems. Using this technique, the relevant information for each task can be determined. Thus, a specific task can be performed using a standard HMM-based tagger without modifying the learning and testing processes.

The results reported here show that the HMM approach performs in line with other approaches that use more sophisticated learning methods when an appropriate definition of the input and output vocabularies is provided. Moreover, this approach maintains the efficiency of the system throughout both the learning and the testing phases.

The specialization methods proposed are independent of the corpus and the language used. The lexicalization criteria presented provide sets of words that are very common, such as words that belong to closed categories or words that appear frequently in the corpus.

These selected words can also appear in other English corpora and, therefore, the chunking or clause identification problem could be successfully solved using this technique. Moreover, the criteria presented here are independent of the language. This has been contrasted in previous works for the POS tagging problem in English and Spanish corpora (Pla and Molina, 2001, Pla et al., 2001). Unfortunately, these aspects could not been tested for chunking and clause-detection due to the unavailability of other annotated corpora.

We think the method presented here can be improved in two aspects: the selection of the features that have to be included in the input and output vocabularies for each disambiguation task, and the selection of the words that really improve the performance of the system. To do this, it would be necessary to take into account not only statistical criteria, but linguistic criteria as well.

Finally, due to the fact that this technique does not need to change the learning and tagging processes, we think that the application of this technique using other taggers based on different paradigms could be of interest.

Acknowledgments

We would like to thank to the reviewers for their helpful comments. This work has been supported by the Spanish research projects CICYT TIC2000–0664–C02–01 and TIC2000–1599–C01–01.

References

S. Abney. *Parsing by Chunks*. R. Berwick, S. Abney and C. Tenny (eds.) Principle–based Parsing . Kluwer Academic Publishers, Dordrecht, 1991.

S. Abney. Partial Parsing via Finite-State Cascades. In *Proceedings of the ESSLLI'96 Robust Parsing Workshop*, Prague, Czech Republic, 1996.

S. Aït-Mokhtar and J.P. Chanod. Incremental Finite-State Parsing. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, Washington D.C., USA, 1997.

S. Argamon, I. Dagan, and Y. Krymolowski. A Memory–based Approach to Learning Shallow Natural Language Patterns. In *Proceedings of the joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics, COLING-ACL*, pages 67–73, Montréal, Canada, 1998.

Thorsten Brants. TnT – a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing (ANLP-2000)*, Seattle, WA, 2000.

E. Brill. Transformation–based Error–driven Learning and Natural Language Processing: A Case Study in Part–of–speech Tagging. *Computational Linguistics*, 21(4):543–565, 1995.

Xavier Carreras and Luís Màrquez. Boosting trees for clause splitting. In Walter Daelemans and Rémi Zajac, editors, *Proceedings of CoNLL-2001*, pages 73–75. Toulouse, France, 2001.

K. W. Church. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Proceedings of the 1st Conference on Applied Natural Language Processing, ANLP*, pages 136–143. ACL, 1988.

W. Daelemans, S. Buchholz, and J. Veenstra. Memory-Based Shallow Parsing. In *Proceedings of EMNLP/VLC-99*, pages 239–246, University of Maryland, USA, June 1999.

W. Daelemans, Antal Van den Bosch, and T. Weijters. *IGTree: Using Trees for Compression and Classification in Lazy Learning Algorithms*. D. Aha (ed.), Artificial Intelligence Review 11, Special issue on Lazy Learning. Kluwer Academic Publishers, 1997.

Hervé Déjean. Learning Syntactic Structures with XML. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, September 2000.

Hervé Déjean. Using allis for clausing. In Walter Daelemans and Rémi Zajac, editors, *Proceedings of CoNLL-2001*, pages 64–66. Toulouse, France, 2001.

James Hammerton. Clause identification with long short-term memory. In Walter Daelemans and Rémi Zajac, editors, *Proceedings of CoNLL-2001*, pages 61–63. Toulouse, France, 2001.

Christer Johansson. A Context Sensitive Maximum Likelihood Approach to Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, September 2000.

J.D. Kim, S.Z. Lee, and H.C. Rim. HMM Specialization with Selective Lexicalization. In *Proceedings of the join SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP-VLC-99)*, 1999.

Rob Koeling. Chunking with Maximum Entropy Models. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, September 2000.

Taku Kudo and Yuji Matsumoto. Use of Support Vector Learning for Chunk Identification. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, September 2000.

Taku Kudo and Yuji Matsumoto. Chunking with Support Vector Machines. In *Proceedings of NAACL 2001*, Pittsburgh, USA, 2001. Morgan Kaufman Publishers. http://cactus.aist-nara.ac.jp/˜taku-ku/publications/naacl2001.ps.

Sang-Zoo Lee, Juni ichi Tsujii, and Hae-Chang Rim. Lexicalized Hidden Markov Models for Part-of-Speech Tagging. In *Proceedings of 18th International Conference on Computational Linguistics*, Saarbrucken, Germany, August 2000.

Xin Li and Dan Roth. Exploring Evidence for Shallow Parsing. In *Proceedings of the 5th Conference on Computational Natural Language Learning (CoNLL-2001)*, Toulouse, France, July 2001.

B. Merialdo. Tagging English Text with a Probabilistic Model. *Computational Linguistics*, 20(2):155–171, 1994.

Antonio Molina and Ferran Pla. Clause detection using HMM. In *Proceedings of the 5th Conference on Computational Natural Language Learning (CoNLL-2001)*, Toulouse, France, July 2001.

Constantin Orasan. A hybrid method for clause splitting in unrestricted English texts. In *Proceedings of ACIDCA'2000*, Monastir, Tunisia, 2000.

Miles Osborne. Shallow Parsing as Part-of-Speech Tagging. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, September 2000.

Jon D. Patrick and Ishaan Goyal. Boosted decision graphs for nlp learning tasks. In Walter Daelemans and Rémi Zajac, editors, *Proceedings of CoNLL-2001*, pages 58–60. Toulouse, France, 2001.

Ferran Pla and Antonio Molina. Part-of-Speech Tagging with Lexicalized HMM. In *proceedings of International Conference on Recent Advances in Natural Language Processing (RANLP2001)*, Tzigov Chark, Bulgaria, September 2001.

Ferran Pla, Antonio Molina, and Natividad Prieto. Tagging and Chunking with Bigrams. In *Proceedings of the COLING–2000*, Saarbrücken, Germany, August 2000a.

Ferran Pla, Antonio Molina, and Natividad Prieto. Improving Chunking by means of Lexical-Contextual Information in Statistical Language Models. In *Proceedings of ConNLL–2000*, Lisbon, Portugal, September 2000b.

Ferran Pla, Antonio Molina, and Natividad Prieto. Evaluación de un etiquetador morfosintáctico basado en bigramas especializados para el castellano. *Revista para el Procesamiento del Lenguaje Natural*, 2001.

L. Ramshaw and M. Marcus. Text Chunking Using Transformation-Based Learning. In *Proceedings of third Workshop on Very Large Corpora*, pages 82–94, June 1995. ftp://ftp.cis.upenn.edu/pub/chunker/wvlcbook.ps.gz.

A. Ratnaparkhi. A Maximum Entropy Part–of–speech Tagger. In *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing, EMNLP*, 1996.

A. Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Phd. Thesis, University of Pennsylvania, 1998. http://www.cis.upenn.edu/~adwait.

Erik F. Tjong Kim Sang. Text Chunking by System Combination. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, September 2000.

Erik F. Tjong Kim Sang. Memory-based clause identification. In Walter Daelemans and Rémi Zajac, editors, *Proceedings of CoNLL-2001*, pages 67–69. Toulouse, France, 2001.

Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, September 2000.

Erik F. Tjong Kim Sang, Walter Daelemans, Herv Djean, Rob Koeling, Yuval Krymolowsky, Vasin Punyakanok, and Dan Roth. Applying System Combination to Base Noun Phrase Identification. In *Proceedings of 18th International Conference on Computational Linguistics COLING'2000*, pages 857–863, Saarbrücken, Germany, August 2000. Morgan Kaufman Publishers. http://lcg-www.uia.ac.be/~erikt/papers/coling2000.ps.

Erik F. Tjong Kim Sang and Hervé Déjean. Introduction to the CoNLL-2001 shared task: Clause identification. In *Proceedings of the 5th Conference on Computational Natural Language Learning (CoNLL-2001)*, Toulouse, France, July 2001.

Hans Van Halteren. Chunking with WPDV Models. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, September 2000.

Jorn Veenstra and Antal Van den Bosch. Single-Classifier Memory-Based Phrase Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, September 2000.

Atro Voutilainen. NPTool, a Detector of English Noun Phrases. In *Proceedings of the Workshop on Very Large Corpora*. ACL, June 1993.

Tong Zhang, Fred Damerau, and David Johnson. Text chunking using regularized Winnow. In *proceedings of the Joint EACL-ACL Meeting (ACL2001)*, Toulouse, France, July 2001.

GuoDong Zhou, Jian Su, and TongGuan Tey. Hybrid Text Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, September 2000.