Feature-Rich Memory-Based Classification for Shallow NLP and Information Extraction

Jakub Zavrel[1] and Walter Daelemans[2]

[1] Textkernel BV, Nieuwendammerkade 28/a17, 1022 AB, Amsterdam, The Netherlands
zavrel@textkernel.nl
[2] CNTS, University of Antwerp, Universiteitsplein 1, Building A, B-2610 Antwerpen, Belgium
walter.daelemans@uia.ua.ac.be

Abstract. Memory-Based Learning (MBL) is based on the storage of all available training data, and similarity-based reasoning for handling new cases. By interpreting tasks such as POS tagging and shallow parsing as classification tasks, the advantages of MBL (implicit smoothing of sparse data, automatic integration and relevance weighting of information sources, handling exceptional data) contribute to state-of-the-art accuracy. However, Hidden Markov Models (HMM) typically achieve higher accuracy than MBL (and other Machine Learning approaches) for tasks such as POS tagging and chunking. In this paper, we investigate how the advantages of MBL, such as its potential to integrate various sources of information, come to play when we compare our approach to HMMs on two Information Extraction (IE) datasets: the well-known Seminar Announcement data set and a new German Curriculum Vitae data set.

1 Memory-Based Language Processing

Memory-Based Learning (MBL) is a supervised classification-based learning method. A vector of feature values (an instance) is associated with a class by a classifier that *lazily* extrapolates from the most similar set (*nearest neighbors*) selected from all stored training examples. This is in contrast to *eager* learning methods like decision tree learning [26], rule induction [9], or Inductive Logic Programming [7], which abstract a generalized structure from the training set beforehand (forgetting the examples themselves), and use that to derive a classification for a new instance.

In MBL, a distance metric on the feature space defines what are the nearest neighbors of an instance. Metrics with feature weights based on information-theory or other relevance statistics allow us to use rich representations of instances and their context, and to balance the influences of diverse information sources in computing distance.

Natural Language Processing (NLP) tasks typically concern the mapping of an input representation (e.g., a series of words) into an output representation (e.g., the POS tags corresponding to each word in the input). Most NLP tasks can therefore easily be interpreted as sequences of classification

tasks: e.g., given a word and some representation of its context, decide what tag to assign to each word in its context. By creating a separate classification instance (a "moving window" approach) for each word and its context, shallow syntactic or semantic structures can be produced for whole sentences or texts. In this paper, we argue that more semantic and complex input-output mappings, such as Information Extraction, can also effectively be modeled by such a Memory-based classification-oriented framework, and that this approach has a number of very interesting advantages over rivalling methods, most notably that each classification decision can be made dependent on a very rich and diverse set of features.

The properties of MBL as a lazy, similarity-based learning method seem make a good fit to the properties of typical disambiguation problems in NLP:

- **Similar input representations lead to similar output.** E.g., words occurring in a similar context in general have the same POS tag. Similarity-based reasoning is the core of MBL.
- **Many sub-generalizations and exceptions.** By keeping in memory all training instances, exceptions included, an MBL approach can capture generalization from exceptional or low-frequency cases according to [12].
- **Need for integration of diverse types of information.** E.g., in Information Extraction, lexical features, spelling features, syntactic as well as phrasal context features, global text structure, and layout features can potentially be very relevant.
- **Automatic smoothing in very rich event spaces.** Supervised learning of NLP tasks regularly runs into problems of *sparse data*; not enough training data is available to extract reliable parameters for complex models. MBL incorporates an implicit robust form of smoothing by similarity [33].

In the remainder of this Section, we will show how a memory-, similarity-, and classification-based approach can be applied to shallow syntactic parsing, and can lead to state-of-the-art accuracy. Most of the tasks discussed here can also easily be modeled using Hidden Markov Models (HMM), and often with surprising accuracy. We will discuss the strengths of the HMMs and draw a comparison between the classification-based MBL method and the sequence-optimizing HMM approach (Section 1.2).

1.1 Memory-Based Shallow Parsing

Shallow parsing is an important component of most text analysis systems in Text Mining applications such as information extraction, summary generation, and question answering. It includes discovering the main constituents of sentences (NPs, VPs, PPs) and their heads, and determining syntactic relationships like subject, object, adjunct relations between verbs and heads of other constituents. This is an important first step to understanding the who, what, when, and where of sentences in a text.

In our approach to memory-based shallow parsing, we carve up the syntactic analysis process into a number of classification tasks with input vectors representing a focus item and a dynamically selected surrounding context. These classification tasks can be segmentation tasks (e.g., decide whether a focus word or tag is the start or end of an NP) or disambiguation tasks (e.g., decide whether a chunk is the subject NP, the object NP or neither). Output of some memory-based modules is used as input by other memory-based modules (e.g., a tagger feeds a chunker and the latter feeds a syntactic relation assignment module). Similar ideas about cascading of processing steps have also been explored in other approaches to text analysis: e.g., finite state partial parsing [1,18], statistical decision tree parsing [23], and maximum entropy parsing [30]. The approach briefly described here is explained and evaluated in more detail in [10,11,6] [1].

Chunking The phrase chunking task can be defined as a classification task by generalizing the approach of [28], who proposed to convert NP-chunking to tagging each word with **I** for a word inside an NP, **O** for outside an NP, and **B** for between two NPs). The decision on these so called IOB tags for a word can be made by looking at the Part-of-Speech tag and the identity of the focus word and its local context. For the more general task of chunking other non-recursive phrases, we simply extend the tag set with IOB tags for each type of phrase. To illustrate this encoding with the extended IOB tag set, we can tag the sentence:

```
But/CC [NP the/DT dollar/NN NP] [ADVP later/RB ADVP]
[VP rebounded/VBD VP] ,/, [VP finishing/VBG VP] [ADJP slightly/RB
higher/R ADJP] [Prep against/IN Prep] [NP the/DT yen/NNS NP]
[ADJP although/IN ADJP] [ADJP slightly/RB lower/JJR ADJP]
[Prep against/IN Prep] [NP the/DT mark/NN NP] ./.
```

as:

$$\begin{aligned}&\texttt{But/CC}_{O}\ \texttt{the/DT}_{I-NP}\ \texttt{dollar/NN}_{I-NP}\ \texttt{later/RB}_{I-ADVP}\\&\texttt{rebounded/VBD}_{I-VP}\ \texttt{,/,}_{O}\ \texttt{finishing/VBG}_{I-VP}\ \texttt{slightly/RB}_{I-ADVP}\\&\texttt{higher/RBR}_{I-ADVP}\ \texttt{against/IN}_{I-Prep}\ \texttt{the/DT}_{I-NP}\ \texttt{yen/NNS}_{I-NP}\\&\texttt{although/IN}_{I-ADJP}\ \texttt{slightly/RB}_{B-ADJP}\ \texttt{lower/JJR}_{I-ADJP}\\&\texttt{against/IN}_{I-Prep}\ \texttt{the/DT}_{I-NP}\ \texttt{mark/NN}_{I-NP}\ \texttt{./.}_{O}\end{aligned}$$

Table 1 (from [6]) shows the accuracy of this memory-based chunking approach when training and testing on Wall Street Journal material. We report Precision, Recall, and $F_{\beta=1}$ scores, a weighted harmonic mean of Recall and Precision ($F_\beta = \frac{(\beta^2+1)*P*R}{\beta^2*P+R)}$).

[1] An online demonstration of the Memory-Based Shallow Parser can be found at `http://ilk.kub.nl` .

type	precision	recall	$F_{\beta=1}$
NPchunks	92.5	92.2	92.3
VPchunks	91.9	91.7	91.8
ADJPchunks	68.4	65.0	66.7
ADVPchunks	78.0	77.9	77.9
Prepchunks	95.5	96.7	96.1
PPchunks	91.9	92.2	92.0
ADVFUNCs	78.0	69.5	73.5

Table 1. Results of chunking–labeling experiments. Reproduced from [6].

Grammatical Relation Finding After POS tagging, phrase chunking and labeling, the last step of the shallow parsing consists of resolving the (types of) attachment between labeled phrases. This is done by using a classifier to assign a grammatical relation (GR) between pairs of words in a sentence. In our approach, one of these words is always a verb, since this yields the most important GRs. The other word (focus) is the head of the phrase which is annotated with this grammatical relation in the treebank (e.g., a noun as head of an NP).

An instance for such a pair of words is constructed by extracting a set of feature values from the sentence. The instance contains information about the verb and the focus: a feature for the word form and a feature for the POS of both. It also has similar features for the local context of the focus. Experiments on the training data suggest an optimal context width of two words to the left and one to the right. In addition to the lexical and the local context information, superficial information about clause structure was included: the distance from the verb to the focus, counted in words. A negative distance means that the focus is to the left of the verb. Other features contain the number of other verbs between the verb and the focus, and the number of intervening commas. These features were chosen by manual "feature engineering". Table 2 shows some of the feature-value instances corresponding to the following sentence (POS tags after the slash, chunks denoted with square and curly brackets, and adverbial functions after the dash):

[**ADVP** *Not*/RB *surprisingly*/RB **ADVP**] ,/, [**NP** *Peter*/NNP *Miller*/NNP **NP**] ,/, [**NP** *who*/WP **NP**] [**VP** *organized*/VBD **VP**] [**NP** *the*/DT *conference*/NN **NP**] **{PP-LOC** [**Prep** *in*/IN **Prep**] [**NP** *New*/NNP *York*/NNP **NP**] **PP-LOC}** ,/, [**VP** *does*/VBZ *not*/RB *want*/VB *to*/TO *come*/VB **VP**] **{PP-DIR** [**Prep** *to*/IN **Prep**] [**NP** *Paris*/NNP **NP**] **PP-DIR}** [**Prep** *without*/IN **Prep**] [**VP** *bringing*/VBG **VP**] [**NP** *his*/PRP$ *wife*/NN **NP**].

Table 3 shows the results of the experiments. In the first row, only POS tag features are used. Other rows show the results when adding several types of chunk information as extra features. The more structure is added, the better

Struct.			Verb		Context -2			Context -1			Focus					Context +1			Class
					word	pos	cat	word	pos	cat	pr	word	pos	cat	adv	word	pos	cat	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
-5	0	2	org.	vbd	-	-	-	-	-	-	-	surpris.	rb	advp	-	,	,	-	-
-3	0	1	org.	vbd	surpris.	rb	advp	,	,	-	-	Miller	nnp	np	-	,	,	-	-
-1	0	0	org.	vbd	Miller	nnp	np	,	,	-	-	who	wp	np	-	org.	vbd	vp	np-sbj
1	0	0	org.	vbd	who	wp	np	org.	vbd	vp	-	conf.	nn	np	-	York	nnp	pp	np
2	0	0	org.	vbd	org.	vbd	vp	conf.	nn	np	in	York	nnp	pp	loc	,	,	-	-

Table 2. The first five instances for the sentence in the text. Features 1–3 are the features for distance and intervening VPs and commas. Features 4 and 5 show the verb and its POS. Features 6–8, 9–11 and 17–19 describe the context words/chunks, Features 12–16 the focus chunk. Empty contexts are indicated by the "-" for all features.

the results: precision increases from 60.7% to 74.8%, recall from 41.3% to 67.9%. This in spite of the fact that the added information is not always correct, because it was predicted for the test material on the basis of the training material by the chunking classifiers.

				All			Subj.	Obj.	Loc.	Temp.
Structure in input	Feat.	# Inst.	Δ	Prec	Rec	$F_{\beta=1}$	$F_{\beta=1}$	$F_{\beta=1}$	$F_{\beta=1}$	$F_{\beta=1}$
words & POS only	13	350091	6.1	60.7	41.3	**49.1**	52.8	49.4	34.0	38.4
+NP chunks	17	227995	4.2	65.9	55.7	60.4	64.1	75.6	37.9	42.1
+VP chunks	17	186364	4.5	72.1	62.9	67.2	78.6	75.6	40.8	46.8
+ADVP/ADJP chunks	17	185005	4.4	72.1	63.0	67.3	78.8	75.8	40.4	46.5
+Prep chunks	17	184455	4.4	72.5	64.3	68.2	81.2	75.7	40.4	47.1
+PP chunks	18	149341	3.6	73.6	65.6	69.3	81.6	80.3	40.6	48.3
+ADVFUNCs	19	149341	3.6	74.8	67.9	**71.2**	81.8	81.0	46.9	63.3

Table 3. Results of grammatical relation assignment with increasing levels of 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 $F_{\beta=1}$ over all relations, and $F_{\beta=1}$ over some selected relations.

1.2 Hidden Markov Models

Except for the grammatical relation finder described in the previous section, all components of the memory-based shallow parser could also be modeled using Hidden Markov Models. An HMM is a finite state automaton with probabilities attached to state transitions and to symbol emissions [27]. The models are called 'Hidden' because we can not uniquely deduce which path the automaton took through its state space from the observation of an emitted symbol sequence. Tagging can be represented in an HMM by modeling tags as states, so that the transition probability $P(t_i|t_{i-1})$ corresponds to the

conditional probability of seeing tag t_i after tag t_{i-1}. The emission probabilities $P(w_j|t_i)$ correspond to the chance of seeing a particular word w_j when being in state t_i. These probabilities can be estimated from relative frequencies in a tagged corpus, and are usually smoothed to accommodate for sparse data and supplemented with a separate unknown word 'guesser' module to provide lexical probabilities for out of vocabulary words. A sequence of symbols is tagged by finding the most probable path through the state space, given the model's parameters and the input sequence.

A tag bigram model can in some sense only look at the previous tag in the sequence and the lexical probabilities of the current word as an information source for disambiguation. Richer information sources (i.e., features) can be modeled by making the state space more complex, e.g., by having states for every pair of tags, as is the case in typical trigram POS taggers [8,14,4]. An HMM can also be interpreted as a sequence of classification decisions, namely each step through the model classifies the present word given the previous decision (or the previous two decisions in the case of trigram taggers) and the word itself (the features). However, in contrast to e.g. the memory-based classification models described above, HMMs do not commit to the previous decision and move on. The search for the most probable path takes each 'classification' at step $i-1$ into account, together with its probability, so that all possible tag sequences are considered. In this way, a locally good classification (based on only two features 'word' and 'tag-1' can be overridden because it leads to a less likely path later in the sequence.

Due to this property, the HMM's information horizon can effectively be much larger that the explicit number of features it uses. In combination with effective smoothing and unknown word guessing techniques, and the fact that the small number of features allows for robust parameter estimation from even very modest amounts of training data, this makes a good implementation of a trigram HMM difficult to beat for POS tagging.

	LOB	WSJ	Wotan	WotanLite
Transformation-Based	96.37	96.28	–	94.63
Memory-Based	97.06	96.41	89.78	94.92
Maximum-Entropy	97.52	**96.88**	91.72	**95.56**
TnT	**97.55**	96.63	**92.06**	95.26

Table 4. The accuracy of feature-rich vs. HMM taggers on four POS datasets.

Table 4, shows how Trigrams 'n Tags (TnT) [4], rivals three other, feature-rich and classification-based algorithms [2] on a number of corpora.

[2] The Maximum Entropy-based tagger MXPOST [29], the Transformation-based tagger of [5], and the Memory-based tagger MBT [10] (adapted from [19].

Many modern approaches to Information Extraction also make use of Hidden Markov Models or variants thereof [3,16,24]. An Information Extraction task is typically modelled with an HMM that has a background state (or empty tag) for all words except a phrase to be extracted and a filler state (or extract tag) for the phrase that is a filler for a field to be extracted. Several variations on this setup have shown to be surprisingly accurate in comparison to various rule-learning methods. There are however, two important issues with this straightforward use of HMMs in Information Extraction. First, the HMMs are fundamentally short-sighted to the larger lexical context. In semi-structured text, an instance of a filler string is often preceded by a quasi-regular context. E.g., in the seminar announcement data set discussed below, a speaker instance is often preceded by phrases like: "Who :", "featuring", or followed by phrases like "will give a talk", "will discuss", etc. An HMM with only states for background and filler tags will never be able to make use of such information: The typical left context of a phrase of interest will be generated from the background state, just as other irrelevant parts of the document. The transition from the background state to the filler state will not become more likely by the occurrence of the informative cue phrase. In the works cited above, this has been dealt with, either by modifying the standard conditioning of the state transition probability from $P(t_i|t_{i-1})$ to $P(t_i|t_{i-1}, w_{i-1})$ [3], or by modifying the state space of the HMM so that special *prefix* and *suffix* states are reserved to tag the left and right context of the filler [16,17]. These modifications have shown to work very well in experimental comparisons, making HMMs the state-of-the-art method for Information Extraction. A second problem in the HMM framework is how to use diverse information sources. In many interesting IE domains success may depend on knowledge of the syntactic structure, or on the paragraph, discourse, or layout structure of the whole text. It is possible to incorporate these types of information in the model's state space as well. However, the room for maneuvering is limited. Each additional feature that is factored into the state space increases its size multiplicatively. This leads to an exponential slowdown of the HMM, whose tagging speed is quadratic in the size of the state space, and to exponential increase of data sparsity. The effects of data sparsity can be mitigated by effective smoothing techniques, such as deleted interpolation, or shrinkage [16]. Shrinkage, for example, interpolates the probability distributions between complex states in a model and similar states in simpler versions of the model. However, the need to define an explicit back-off ordering from specific complex states to similar but more general states leads to a combinatorial explosion as well [33]. In contrast, Memory-Based Learning, and other classification-based frameworks, can provide better solutions for these complex modeling problems by factoring the fusion of information sources into the feature space rather than into the state space.

2 Memory-Based Information Extraction

The task of Information Extraction (IE), arguably the core activity in the field of text mining, is to extract specific pre-defined types of information from unrestricted text; i.e. finding fillers in a text for predefined slots in some template. In current approaches to IE using supervised machine learning, extraction patterns are learned on the basis of linguistically enriched corpora, and the patterns are, possibly after manual post-processing, used in IE systems [31,20,32,7].

However, it is also possible to interpret IE as a classification task, similar to NLP tasks like POS tagging and chunking. The input representation is a word and (information about) its context, and the output is either the background class *none*, or one of the slot filler classes. For example, in a hypothetical IE system for terrorist attacks, the following mapping can be learned as a supervised classification-based task.

> John/**I-victim** Doe/**I-victim** , minister/*none* of/*none* language/*none* technology/*none* was/*none* killed/*none* by/*none* a/*none* car/**I-weapon** bomb/**I-weapon** yesterday/**I-time** evening/**I-time** ./*none*

From this tagged sentence the following template could be extracted:

Victim:	John Doe
Weapon:	car bomb
Time:	yesterday evening

We can also choose to assign, in one or several steps of classification, much more structure to the input than just the phrases to be extracted. For example, the title and author information of this paper might be tagged with section information and shallow syntactic structure as follows:

```
Feature-Rich    JJ:NP[:I-title
Memory-Based    JJ:NP:I-title:I-topic
Classification  NN:NP]:I-title:I-topic
for IN:PP[:I-title
Information NN:PP[:NP[:I-title:I-topic
Extraction  NN:PP]:NP]:I-title:I-topic}
XXXBLANKXXX none
Jakub   NNP:NP[:I-authors:I-name
Zavrel  NNP:NP]:I-authors:I-name
and CC:I-authors
Walter  NNP:NP[:I-authors:I-name
Daelemans   NNP:NP]:I-authors:I-name
```

```
Textkernel  NNP:NP[:I-authors:I-organisation
BV  NNP:NP]:I-authors:I-organisation
,   ,:I-authors
Nieuwendammerkade   NNP:NP[:I-authors:I-address
28  a17 SYM:NP]:I-authors:I-address
,   ,:I-authors:I-address
1022    CD:NP[:I-authors:I-address
AB  NNP:NP]:I-authors:I-address
,   ,:I-authors:I-address
Amsterdam   NNP:NP[]:I-authors:I-address
,   ,:I-authors:I-address
The DT:NP[:I-authors:I-address
Netherlands NNP:NP]:I-authors:I-address
CNTS    SYM:NP[]:I-authors:I-organisation
,   ,:I-authors:I-organisation
University  NNP:NP[:I-authors:I-organisation
of  NNP:NP[:I-authors:I-organisation
Antwerp NNP:NP[:I-authors:I-organisation
,   ,:I-authors
Universiteitsplein  NNP:NP[:I-authors:I-address
1   CD:NP]:I-authors:I-address
,   ,:I-authors:I-address
Building    NN:NP[:I-authors:I-address
A   SYM:NP:I-authors:I-address
,   ,:I-authors:I-address
B-2610  SYM:NP[]:I-authors:I-address
Antwerpen   NNP:NP[]:I-authors:I-address
,   ,:I-authors:I-address
Belgium NNP:NP[]:I-authors:I-address
```

Leading to the following XML structure (leaving POS tags aside):

```
<title>
<np>Feature-Rich <topic>Memory-Based Classification</topic></np>
<pp>for <np><topic>Information Extraction</topic></np></pp>
</title>
<authors>
<name><np>Jakub Zavrel</np></name> and
<name><np>Walter Daelemans</np></name>
<organisation><np>Textkernel BV</np></organisation>,
<address><np>Nieuwendammerkade 28/a17</np>, <np>1022 AB</np>,
<np>Amsterdam</np>, <np>The Netherlands</np></address>
<organisation><np>CNTS</np>, <np>University of Antwerp</np>,
<address><np>Universiteitsplein 1</np>, <np>Building A</np>,
<np>B-2610</np> Antwerpen, Belgium</np></address>
</authors>
```

In an HMM approach, such codings will lead to an explosion of the state space, or a necessary decomposition of the task into a sequence of many small sub-problems. However, from an informational point of view, there are probably many interesting dependencies between decisions at various levels of structure. A feature-rich classification based approach, such as Memory-Based Learning, allows a representation of such tasks as either monolithic classification tasks, or decomposed cascades or ensembles of tasks, whatever is best for accuracy and speed. When cascading Memory-Based classifiers, we can always train a higher level of classification to use the outputs of lower levels as input features.

2.1 TK_SemTagger

We have implemented a general environment called **TK_SemTagger** as part of Textkernel's Textractor toolkit in order to allow experimentation with the Memory-Based tagging approach to IE. In Textractor, we use this component side by side with HMMs, induced extraction rules and Shallow NLP preprocessing to benefit from the complementary strengths of all methods.

Architecture During training, the tagger reads in the training corpus, and constructs a corpus based lexicon (see below). If needed it also reads in other information sources, such as a domain lexicon, and connects to external NLP pre-processors. Then the training corpus is converted into training instances (feature vectors) according to a flexible feature set specification. These instances are used to train a Memory-based classifier (TiMBL [13], an efficient implementation of MBL). The manner in which cases are retrieved from memory, the metric used for computing similarity, the way features are weighted, the number of nearest neighbors used, and the weighting of neighbors during extrapolation are all parameterized. For a full discussion of TiMBL's parameters we refer the reader to [13].

After training the classifier, test data is processed given the same feature set specification, and each token is classified. The test data is processed sentence by sentence (for the IE tasks described below, a 'sentence' is a whole document). Each sentence is processed from left to right, so that previous classification decisions are propagated to the left context (for use as features in subsequent decisions). `TK_SemTagger` also allows the use of an *ensemble* of classifiers (e.g., with different features or different parameters), which can have dependencies between each other. The ensembles are combined by stacking (two levels: L1 and L2). The classifiers in level L2 are applied *after* the L1 classifiers have tagged the entire sentence, and hence can refer to decisions made by L1 classifiers both to the left and the right of the word to be tagged. In the present study, however, no exploration was made of ensemble systems.

Lexical information Tagging is an exercise in the satisfaction of two simultaneous constraints: lexical possibilities of a word and contextual possibilities. In HMM-based taggers, the lexicon (or the unknown word guesser) typically proposes a selection of tags for the present word, and the tags that are most compatible with the present context are selected. In classification-based tagging, in contrast, the lexical representations of a word are seen as yet another symbolic feature. This gives us greater freedom, as we can also assign a tag that is not in a word's lexical representation (and it is well-known that lexicons are seldom complete). We can even use a lexicon that contains tags from a completely different tag set [34].

The most important lexical information in our Memory-Based IE approach is compiled directly from the training corpus. We record the number of times each word has been used with each tag. To convert this frequency information into a symbolic feature, the set of tags for a word is sorted by frequency, and the tags that fall below a percentage threshold are pruned away. An empirically derived good default value for this threshold is 5%. In addition to this, words that occur very often (more than 25 times) and have no ambiguous occurrences are labeled 'sure thing'. This means that during tagging, their lexical tag is assigned without looking at the context. Since the lexical ambiguity class of a word is just another feature, we are able to use any number of different lexicons, such as e.g. domain-specific ontologies or gazetteers. However, in the present experiments this option was not used.

Features A case for classification can be represented using a multitude of features. The features are specified as a template of positions and types. The best setting depends on the task at hand, the selected features and their representation, the amount of data, and the used TiMBL parameters. The following types are available at each position i.

- **word**: gives the string of the word at position i.
- **wordf***expr*: where *expr* is a constraint on the frequency of the word to be included as a feature (e.g., 'wordf¿5' or 'wordf¡=10').
- **tag**: gives the full tag of the word at position i. This feature refers to a decision of the current classifier on an earlier word in the sentence, and can therefore only be used on positions in the left context.
- **known**: has a value of 'KNOWN' for words that have been seen in the training corpus, and a value of 'UNKNOWN' for words that did not occur in it.
- **lex**: gives the lexical representation (ambiguity class) of the word, as found in the lexicon constructed from the training corpus (s. Section 2.1).
- **domlex**: gives the lexical representation (ambiguity class) of the word as found in an externally supplied lexicon with domain knowledge.
- **exttag**: gives the tag given to a position by an external tagger, chunker or other syntactic pre-processor.

- **prev:***tag*2: this is a binary feature. It is on if *tag*2 (a second level tag) is present somewhere in the left history of previous tagging decisions. I.e. this feature means 'have seen *tag*2 before'. This type of feature can only be used for the position of the word that is to be tagged, not in left and right context positions.
- **suf***n*: a family of suffix features. 'suf*n* gives the *n*'th letter from the end of the word (e.g., 'suf1' of "bread" is "d", and 'suf3' is "e"). When *n* is given as a range, e.g., 'suf1-3', the value is the suffix between first and last number in the range (e.g., 'suf1-3' of "bread" is "ead").
- **pref***n*: a family of prefix features. Works analogously to 'suf'.
- **num**: a binary feature that indicates the presence of a numeric character in the word.
- **allnum**: a binary feature that indicates whether the token consist of numeric symbols, and possibly punctuation, only.
- **hyp**: a binary feature that indicates the presence of a hyphen in a word.
- **at**: a binary feature that indicates the presence of an 'at' sign ("") in a word.
- **und**: a binary feature that indicates the presence of an underscore in a word.
- **cap**: a binary feature that indicates the presence of a capital letter in a word.
- **allcap**: a binary feature that indicates whether the token is fully capitalized.
- **L1-***name*: a feature that refers to a decision of another classifier (named L1-*name*), that has been applied before the present one. When L1- *name* is the same as that of the current classifier, the result is the same as that of the 'tag' feature. This feature is used to define dependencies between classifiers in an ensemble. It was not used in the present experiments.

The feature template represents a window of word positions to the left (negative numbers) and to the right (positive numbers) of the focus word (the word to be tagged, i.e. position '0'). If any of the specified features refer to the output of another classifier, i.e. when a bootstrap dependency exists, this is solved by using features produced by cross-validating the other classifier on the training set. Figure 1 shows an example of a feature set for one classifier.

3 Data

We performed experiments comparing the TnT implementation of HMMs and the feature rich **TK_SemTagger** approach on two IE tasks. The first, the **Seminar Announcement** data set consists of 485 usenet postings concerning academic seminar announcements. The collection was collected and manually annotated by [15] [3] The set of documents is labeled with four fields

[3] Available from the RISE repository at `http://www.isi.edu/~muslea/RISE/repository.html`.

```
classifier: L1-A
features:
-4 tag wordf>4
-3 tag wordf>4
-2 tag wordf>4
-1 tag wordf>4
0 wordf>4 lex suf1 suf2 pref1 initonlycap initial allcap cap hyp num at
1 lex wordf>4
2 lex wordf>4
3 lex wordf>4
4 lex wordf>4
parameters: -mM -k3 -w4 -dIL
```

Fig. 1. Example of a feature configuration (the best configuration on the Seminar Announcement data set). The Timbl parameters shown are: MVDM; Shared variance weighting; 3 nearest neighbors; Dudani distance weighted voting.

to be extracted: *speaker*, *location*, *stime* (start time), and *etime* (end time). In the original work on this dataset, a 'One Best per Document' (OBD) scoring method was used. This means that if the extraction method can fill an entity slot with any one of the occurrences of that entity in the text, this is counted as correct. However, in the data, all occurrences are marked. In our experiments we have taken a slightly more challenging scoring metric: *Each occurrence* of an entity must be labeled with exactly matching boundaries by the tagger. We used the first half of the data set for training, the first 40 documents from the second half for validation of parameters, and documents 40 through 80 from the second half for testing.

The second task we report results on is the task of populating a database from a set of CVs. The data is a set of 103 **German Curriculum Vitae** that were spidered from the web, and manually annotated with XML structure. Both section information (personal information, work experience, education, other), and extractable entity information was annotated. The following fields are to be extracted: *name address*, *phone*, *email*, *birthday*, *marstat* (marital status), *military* (military service), *compskill* (computer skills), *degree*, *degreedate*, *experience*, *experiencedate*, *langskill* (language skill), and *langprofi* (language proficiency). Since the data set is so small (although it leads to more than 50000 token classification cases), we used the first 10 documents for validation, and the rest for training during tuning, and we report ten-fold cross validation results over the whole dataset for comparing algorithms. Arguably, this puts the more tuned system, (Timbl has four parameters in this experiment, whereas TnT only has one) in a slightly more favorable position. For both datasets, we report $F_{\beta=1}$ scores.

4 Experiments

4.1 HMM results

With the TnT HMM model we tested two variants. The first variant, TnT Simple, has a state for each filler type, and a background state for the re-

mainder of the text. The second variant, TnT Context, has a set of prefix and suffix states, one of each for each type of filler, that is applied to respectively two tokens before, and two tokens after the filler. As can be seen in Figures 5 and 6, the TnT Context model clearly outperforms the TnT Simple model on average. Interestingly, this does not happen on the *email*, *langskill*, and *military* fields of the German CV task, showing that for these fields the lexical content of the filler, rather than some cue phrases typically preceding or following it, is a strong enough feature to accurately make the decision. The TnT Context model suffers in these places from the additional data sparseness introduced by the prefix and suffix states. Increasing the context size for the prefix and suffix states did not further improve performance.

	speaker	location	stime	etime
TnT Context	**0.66**	**0.70**	**0.85**	**0.90**
TnT Simple	0.51	0.62	0.36	0.87

Table 5. Results on the Seminar Announcement test data with two variants of the TnT HMM tagger. This and subsequent figures refer to F1 scores.

	Simple	Context
address	0.00	0.00
birthday	0.46	**0.55**
compskill	0.36	**0.37**
degree	0.42	0.42
degreedate	0.39	**0.42**
email	**0.40**	0.33
experience	0.30	**0.32**
experiencedate	0.46	**0.48**
langprofi	0.63	**0.80**
langskill	**0.90**	0.87
marstat	1.00	1.00
military	0.44	**0.57**
name	**0.71**	0.43
phone	0.00	0.00

Table 6. Results on the German CV test data with two variants of the TnT HMM tagger.

4.2 TK_SemTagger results

For the feature set of TK_SemTagger, we have experimented with variants of the feature set shown in Figure 1. This feature template has previous 'tag'

decisions and 'words' in the left context. The right context and the focus word have word features and 'lex' features (ambiguity classes). Moreover, the focus position has a number of wordform-oriented features which allow us to classify unknown words. The weights of these features are shown in Figure 2.

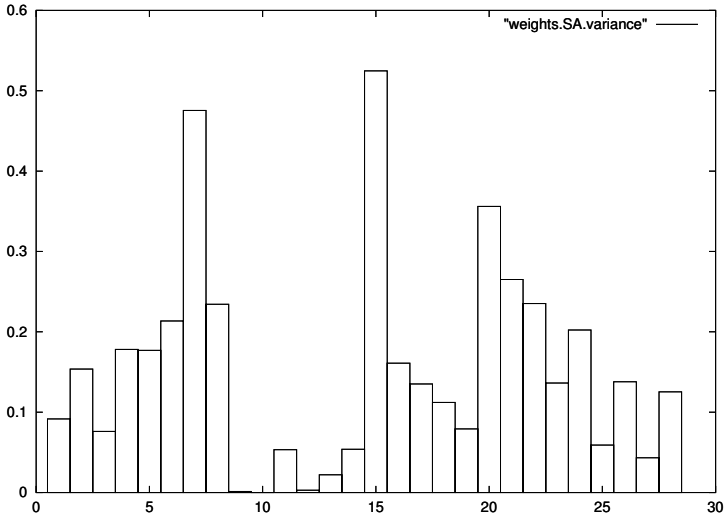

Fig. 2. The feature weights (using the *Shared Variance* statistic in the best setting for the Seminar Announcement data set. The features are the ones shown in Figure 1, ordered per position slot, and sorted alphabetically per slot. The highest weighted feature is the 'lex' feature (15) for the focus word, then the 'tag' of the previous word (7), and then the 'word' in focus position (20).

Context size A first set of experiments on the validation set was performed to test the influence of the number of context positions in the feature template. The results for the Seminar Announcement data shown in Table 7. A context size of zero means that only features of the focus word were used, a size of one means that one position to the left and one to the right were used (with both 'word' and 'tag' or 'lex' features).

Without context features (0) the model is quite helpless. One would expect that with an ideal feature weighting method, more context would simply improve performance. This is, unfortunately, not the case, which points to a serious problem of the feature weighting methods used. Because each feature receives a weight, independently of the others, on the basis of its predictiveness for the category, an unwanted effect takes place: highly redundant

context size	speaker	location	stime	etime
0	0.05	0.69	0.01	0.00
1	0.45	0.87	0.92	0.96
2	0.64	0.84	0.91	0.84
3	0.67	0.83	0.93	0.89
4	0.65	0.84	0.93	0.89
5	0.57	0.82	0.95	0.96

Table 7. Experiments on the Seminar Announcement validation data with different context sizes ('word' and 'tag' features; frequency threshold is 4).

features will tend to overwhelm independent but useful features. If we increase the context size, then even when far away context receives much lower weight than close context, the influence of the focus features on the similarity metric will be diluted. In the remainder of the experiments, we use a context size of 4.

Frequency threshold Low frequent words can sometimes cause the similarity metric to overestimate the importance of unreliable information. In Table 8, we show the effect of including values for the 'word' feature only for words above a certain frequency threshold. The threshold 4 was chosen for further experiments.

frequency threshold	speaker	location	stime	etime
1	0.41	0.84	0.93	0.89
2	0.55	0.82	0.93	0.89
3	0.65	0.81	0.93	0.89
4	0.65	0.84	0.93	0.89
5	0.63	0.86	0.93	0.89
10	0.48	0.83	0.93	0.89

Table 8. Experiments on the Seminar Announcement validation data with different frequency thresholds for the inclusion of the 'word' feature values.

Decision propagation Unlike the HMM, which examines all possible paths through the state space, the current TK_SemTagger has a greedy search strategy. The previous decisions of the tagger are propagated to the left context features. These decisions, however, may be incorrect. However, the model's training set contains correct left context features. Table 9 shows the results of experiments that examine the influence of this effect.

The top row of Table 9, labeled **tags**, shows the normal propagation of tag decisions to the left context. In the second row, labeled **lex**, the left

	speaker	location	stime	etime
tags	0.65	0.84	0.93	0.89
lex	0.43	0.85	0.71	0.29
tags+lex	0.51	0.86	0.93	0.93
bootstrap	0.44	0.83	0.67	0.44
bootstrap+focus	0.42	0.83	0.67	0.44

Table 9. Experiments on the validation set with various propagation types for the left 'tag' context (context size is 4; frequency threshold is 4).

context was constructed using 'lex' features, so no erroneous information is propagated from the classifiers decisions, and the left context patterns are from the same distribution during training and testing. The performance, however, seriously drops. The disambiguated left context is clearly the superior set of predictors, despite the noise. The bad effect of feature-redundancy on the effectivity of weighting is again seen in row three, labeled **tags+lex**. The bottom two rows test the use of a second classifier, identical to the initial one, whose cross-validated output on the training set was used to replace the 'tag' features (**bootstrap**) or to replace the 'tag' features in the context and add a new 'tag' feature for the focus word. Accuracy does not benefit from either of these manipulations.

The categories, or states of the classification-based model were only 'background' and one for each of the types of filler. We experimented briefly with adding prefix and suffix categories like in the TnT Context model, but this only deteriorated performance.

Comparison TK_SemTagger vs TnT Tables 10 and 11 show the best settings of TnT and TK_SemTagger respectively on the test set of the Seminar Announcements data, and in a ten fold cross validation on the German CV data. The results on the SA task are clearly in favor of the Memory-Based model. The variety of strong cue-phrases in this domain is more easily modeled using 'word' features and a large context size. Similar results have been shown by [17] in their optimization of HMM structures toward this task.

	speaker	location	stime	etime
TK_SemTagger	**0.71**	**0.87**	**0.95**	**0.96**
TnT Context	0.66	0.70	0.85	0.90

Table 10. Best parameter setting for TK_SemTagger, tested on the test set, and compared with the TnT Context model.

The results on the German CV data set are more favorable for the HMM approach. The small size of this data set, its larger set of fillers and the large

lexical variety in these fillers seem, so far, to favor the more simple and robust approach.

	TnT Context	TK_ST Best	*Textractor full*
address	0.08	**0.10**	*0.25*
birthday	0.55	0.55	*0.67*
compskill	0.43	0.43	*0.83*
degree	**0.34**	0.29	*0.31*
degreedate	**0.50**	0.35	*0.55*
email	0.53	**0.59**	*0.88*
experience	**0.31**	0.17	*0.38*
experiencedate	**0.42**	0.34	*0.48*
langprofi	**0.60**	0.52	*0.72*
langskill	**0.79**	0.65	*0.87*
marstat	**0.95**	0.87	*0.88*
military	0.58	**0.70**	*0.59*
name	**0.28**	0.21	*0.80*
phone	**0.34**	0.20	*0.84*

Table 11. Comparison between TnT and the best settings of TK_SemTagger in a ten fold cross-validation experiment on the German CV data set. For reference, we have included the results of a somewhat more elaborate system trained by Textkernel on this dataset. This uses a special CV tuned feature set, a combination of HMMs, MBL, and NLP preprocessing (results here are without rule-based post-processing).

5 Discussion and Future Work

We have shown the application of a Memory-Based classification-oriented framework for Information Extraction as an extension of our earlier work in Memory-Based tagging, chunking and shallow parsing. We have compared our approach to Hidden Markov Models and showed competitive or better results. Without denying the robustness and good accuracy of HMMs, we have argued that the feature-rich classification-oriented approach which factors diverse information sources into the feature space, rather than into the state space, offers the potential for future advances in the state of the art, and leads to a more natural way of modeling complex Information Extraction tasks. Similar results can also be achieved with suitable modifications of HMMs. In fact, a number of recently proposed modifications [17,24] which have shown good results have a similar feature-oriented flavor to them. We think it will be possible to borrow the best of the HMM approach, and apply it to the classification-based approach, and *vice versa*. For example, so far,

we have lacked the advantage of optimizing the whole sequence of tags probabilistically. Work on this issue is under way. On the other hand, the work on similarity metrics and variations in feature representation that has received a lot of attention in the classification-oriented line of work could very well benefit the further refinement of HMM style models.

In the present paper, we have shown how our approach can easily be implemented in terms of Memory-Based classifiers. This learning framework has a number of important advantages at the moment, most notably its easy implementation, its flexibility with respect to various modeling scenarios, and its low computational complexity during training and classification. However, the Memory-Based approach also suffers from a number of problems, such as the effective weighting of sparse and interacting features, and the current lack of a clear probabilistic interpretation that allows the principled combination of sequences of decisions into larger representations for sequences and hierarchical structures. Several other classification frameworks, such as Maximum Entropy models [2,24], and Conditional Random Fields [22] are explicitly probabilistic, but this advantage is not always translated into higher generalization accuracy. Moreover, other non-probabilistic classifiers such as Support Vector Machines [21], and Winnow perceptrons [25] have also successfully been adopted as semi-probabilistic components in sequence models, so it seems worthwhile at the moment to further investigate this class of learning models both experimentally and theoretically.

References

1. Abney, S. 1996. Part-of-Speech Tagging and Partial Parsing. In K.W. Church, S. Young and G. Bloothooft (eds.), Corpus-Based Methods in Language and Speech. Kluwer Academic Publishers, Dordrecht, 1996.
2. Berger, A., S. Della Pietra, and V. Della Pietra. 1996. Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1).
3. Bikel, D.M., S. Miller, R. Schwartz, and R. Weischedel. 1997. Nymble: a high-performance learning name-finder. In Proceedings of ANLP-97, pp. 194–201.
4. Brants, T. 2000. TnT – a statistical part-of-speech tagger. In Proc. of the 6th Applied NLP Conference, ANLP-2000, April 29 – May 3, 2000, Seattle, WA.
5. Brill, E. 1994. Some advances in transformation-based part-of-speech tagging. In *Proceedings AAAI'94*.
6. Buchholz, S., J.B. Veenstra, and W. Daelemans. 1999. Cascaded Grammatical Relation Assignment. In Proceedings of EMNLP/VLC-99, University of Maryland, USA, June 21-22, 1999, pp. 239-246.
7. Califf, M.E. and R.J. Mooney 1999. Relational Learning of Pattern-Match Rules for Information Extraction. In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), Orlando, FL, pp. 328-334, July 1999.
8. Church, K. W. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. of Second Applied NLP (ACL)*.
9. Cohen, W.W. 1995. Fast effective rule induction. In Proceedings of the Twelfth International Conference on Machine Learning, Lake Tahoe, CA, 1995.

10. Daelemans, W., J. Zavrel, P. Berck, and S. Gillis. 1996. MBT: A memory-based part of speech tagger generator. In E. Ejerhed and I. Dagan, editors, *Proc. of Fourth Workshop on Very Large Corpora*, pages 14–27. ACL SIGDAT.
11. Daelemans, W., S. Buchholz, and J. Veenstra. 1999. Memory-Based Shallow Parsing In: Proc, of CoNLL-99, Bergen, Norway, June 12, 1999.
12. Daelemans, W., A. Van den Bosch, and J. Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning.*
13. Daelemans, W., J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2001. TiMBL: Tilburg Memory Based Learner, version 4.0, reference manual. Technical Report ILK-0001, ILK, Tilburg University.
14. DeRose, S. 1988. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14:31–39.
15. Freitag, D. Machine Learning for Information Extraction in Informal Domains. PhD. thesis, November, 1998.
16. Freitag D. and A. McCallum. 1999. Information extraction using HMMs and shrinkage. Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction.
17. Freitag D. and A. McCallum, "Information extraction with HMM structures learned by stochastic optimization," Proceedings of AAAI-2000.
18. Grefenstette, G. 1996. Light parsing as finite state filtering. In Workshop on Extended finite state models of language, Budapest, Hungary, Aug 11–12 1996. ECAI'96.
19. van Halteren, H., J. Zavrel, and W. Daelemans. 2001. Improving Accuracy in Word Class tagging through the Combination of Machine Learning Systems. *Computational Linguistics*, Vol.27(2).
20. Huffman, S.B. 1996. Learning Information Extraction Patterns from Examples. In Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing, pp. 246–260. Springer Verlag.
21. Kudoh, T. and Y. Matsumoto. 2001. Chunking with Support Vector Machines. In Proceedings of NAACL 2001, Pittsburgh, PA, USA, 2001.
22. Lafferty, J., A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of ICML-01, pages 282-289, 2001.
23. Magerman, D.M. 1994. Natural Language Parsing as Statistical Pattern Recognition. PhD Thesis, February 1994.
24. McCallum, A., D. Freitag, and F. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. Proceedings of ICML-2000.
25. Punyakanok, V. and D. Roth. 2000. The Use of Classifiers in Sequential Inference. NIPS-13, Dec, 2000.
26. Quinlan, J.R. 1993. C4.5*: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann.
27. Rabiner, L.R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2), pp. 257–286.
28. Ramshaw, L.A. and Marcus, M.P. 1995. Text Chunking using Transformation-Based Learning. In Third Workshop on Very Large Corpora, ACL, pp. 82-94, 1995.

29. Ratnaparkhi, A. 1996. A maximum entropy part-of-speech tagger. In *Proc. of the Conference on Empirical Methods in Natural Language Processing, May 17-18, 1996, University of Pennsylvania*.
30. Ratnaparkhi, A. 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. Aug. 1-2, 1997. Brown University, Providence, RI.
31. Riloff, E. 1993 Automatically Constructing a Dictionary for Information Extraction Tasks. In Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93) , AAAI Press/The MIT Press, pp. 811–816.
32. Soderland, S., D. Fisher, J. Aseltine, and W. Lehnert. 1995. Crystal: Inducing a conceptual dictionary. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, pp. 1314–1319.
33. Zavrel, J. and W. Daelemans. 1997. Memory-based learning: Using similarity for smoothing. In *Proc. of 35th annual meeting of the ACL*, Madrid.
34. Zavrel, J. and W. Daelemans. Bootstrapping a tagged corpus through combination of existing heterogeneous taggers. In Proceedings of the second international conference on language resources and evaluation (LREC-2000), Athens, Greece, 17-20, 2000.