Proceedings ELSNET in Wonderland, 105-113, 1998

RAPID DEVELOPMENT OF NLP MODULES WITH MEMORY-BASED LEARNING

Walter Daelemans, Antal van den Bosch, Jakub Zavrel, Jorn Veenstra, Sabine Buchholz, Bertjan Busser

ILK / Computational Linguistics
Tilburg University
P.O. Box 90153, NL-5000 LE, Tilburg, The Netherlands
URL: http://ilk.kub.nl, E-mail: walter@kub.nl

ABSTRACT

The need for software modules performing natural language processing (NLP) tasks is growing. These modules should perform efficiently and accurately, while at the same time rapid development is often mandatory. Recent work has indicated that machine learning techniques in general, and memory-based learning (MBL) in particular, offer the tools to meet both ends. We present examples of modules trained with MBL on three NLP tasks: (i) text-to-speech conversion, (ii) part-of-speech tagging, and (iii) phrase chunking. We demonstrate that the three modules display high generalization accuracy, and argue why MBL is applicable similarly well to a large class of other NLP tasks.

1. INTRODUCTION

The task of language technology is to develop efficient, high-accuracy software modules that perform NLP tasks or subtasks. The increasing market pull for NLP applications in human-computer interaction intensifies the need for solving the *knowledge acquisition bottleneck* problem identified with many NLP tasks. As is usually argued and shown in present-day NLP applications, some targets are hard to accomplish as they involve many linguistic levels such as syntax and semantics simultaneously (e.g., domain-free multilingual translation, or dialogue modeling); it is time-consuming, if not plainly impossible to gather and compile knowledge covering the many-to-many mappings between these levels. On the other hand, we argue [9, 10] that all NLP tasks can be seen as either

- *light* NLP tasks, involving disambiguation or segmentation [9] locally at one language level or between two closely-related language levels; or as
- *compositions* of light NLP tasks, when the task surpasses the complexity of single light NLP tasks.

This research was performed in the context of the "Induction of Linguistic Knowledge" research programme, partially supported by the Foundation for Language Speech and Logic (TSL), funded by the Netherlands Organization for Scientific Research (NWO).

Rapid development of a system that has to perform a very complex (or *heavy*) NLP task efficiently and accurately, may not be a problem when a good decomposition can be found, and high-accuracy modules performing the light NLP subtasks are available.

In this paper, we consider light NLP tasks which can easily be rephrased as classification tasks. Such tasks can be learned relatively accurately by generic inductive learning techniques. We show that within this class of techniques, MBL is particularly suited for learning light NLP tasks rapidly. MBL is in essence a simple learning method in which examples are massively retained in memory and similarity between memory examples and new examples is used to predict the outcomes of new examples. In Section 2 we briefly describe the functioning of MBL in the context of the TIMBL software package developed at Tilburg University. Section 3 illustrates some of our findings with applying MBL to three light NLP tasks: text-to-speech conversion, part-of-speech tagging, and phrase chunking. In Section 4 we provide explanations for the success of applying MBL to NLP tasks. In Section 5 we mention our ongoing and planned work with combinations of light modules, and we conclude by summarizing our findings, and draw some conclusions on the feasibility of applying TIMBL to light NLP tasks.

2. TIMBL: TILBURG MEMORY-BASED LEARNING

Memory-based learning is founded on the hypothesis that performance in cognitive tasks (in our case NLP) is based on reasoning on the basis of similarity of new situations to *stored representations of earlier experiences*, rather than on the application of *mental rules* abstracted from earlier experiences (as in rule induction and rule-based processing). The approach has surfaced in different contexts using a variety of alternative names such as similarity-based, example-based, exemplar-based, analogical, case-based, instance-based, and lazy learning[23, 7, 17, 2, 3]. Historically, memory-based learning algorithms are descendants of the

k-nearest neighbor (henceforth k-NN) algorithm [8, 16, 2].

A MBL system, visualized schematically in Figure 1, contains two components: a *learning component* which is memory-based (from which MBL borrows its name), and a *performance component* which is similarity-based. The learning component of MBL is memory-based as it involves adding training examples to memory; it is sometimes referred to as 'lazy' as memory storage is done without abstraction or restructuring. In the performance of an MBL system, the product of the learning component is used as a basis for mapping input to output; in the context of performing NLP tasks, this usually takes the form of performing classification. During classification, a previously unseen test example is presented to the system. Its similarity to all examples in memory is computed using a *similarity metric*, and the category of the most similar example(s) is used as a basis for extrapolating the category of the test example.

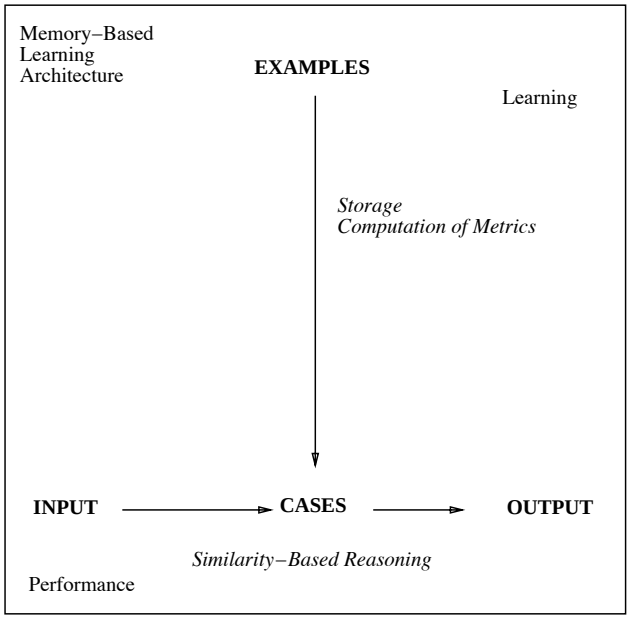

Figure 1: General architecture of an MBL system.

TIMBL[1] is the name of a software package developed by the ILK group at Tilburg University, containing variations of the memory-based learning algorithms IB1,IB1-IG, and MVDM [2, 13, 12, 7], and IGTREE [12], a decision-tree optimization of memory-based learning. Below, we outline the functioning of IB1-IG and IGTREE in Subsections 2.1 and 2.2, respectively.

[1]The TIMBL software package is freely available for research purposes from the ILK web pages; consult URL http://ilk.kub.nl.

2.1. Weighted MBL in TIMBL: IB1-IG

IB1-IG [13, 12] is a memory-based learning algorithm that builds a data base of instances (the *instance base* or case base) during learning. An instance consists of a fixed-length vector of n feature-value pairs, and an information field containing the classification of that particular feature-value vector. After the instance base is built, new (test) instances are classified by matching them to all instances in the instance base, and by calculating with each match the *distance* between the new instance X and the memory instance Y.

The most basic metric for patterns with symbolic features is the **Overlap metric** given in equations 1 and 2; where $\Delta(X, Y)$ is the distance between patterns X and Y, represented by n features, w_i is a weight for feature i, and δ is the distance per feature. The k-NN algorithm with this metric, and equal weighting for all features is called IB1 [2]. Usually k is set to 1.

$$\Delta(X, Y) = \sum_{i=1}^{n} w_i \, \delta(x_i, y_i) \tag{1}$$

where:

$$\delta(x_i, y_i) = 0 \; if \; x_i = y_i, \; else \; 1 \tag{2}$$

We have made two additions to the original algorithm [2] in our version of IB1. First, in the case of nearest neighbor sets larger than one instance ($k > 1$ or ties), our version of IB1 selects the classification with the highest frequency in the class distribution of the nearest neighbor set. Second, if a tie cannot be resolved in this way because of equal frequency of classes among the nearest neighbors, the classification is selected with the highest overall occurrence in the training set.

The distance metric in equation 2 simply counts the number of (mis)matching feature values in both patterns. In the absence of information about feature relevance, this is a reasonable choice. Otherwise, we can add linguistic bias to weight or select different features [6] or look at the behavior of features in the set of examples used for training. We can compute statistics about the relevance of features by looking at which features are good predictors of the class labels. Information Theory gives us a useful tool for measuring feature relevance in this way [19, 20].

Information Gain (IG) weighting looks at each feature in isolation, and measures how much information it contributes to our knowledge of the correct class label. The Information Gain of feature f is measured by computing the difference in uncertainty (i.e. entropy) between the situations without and with knowledge of the value of that feature (Equation 3).

$$w_f = \frac{H(C) - \sum_{v \in V_f} P(v) \times H(C|v)}{si(f)} \tag{3}$$

$$si(f) = -\sum_{v \in V_f} P(v) \log_2 P(v) \quad (4)$$

Where C is the set of class labels, V_f is the set of values for feature f, and $H(C) = -\sum_{c \in C} P(c) \log_2 P(c)$ is the entropy of the class labels. The probabilities are estimated from relative frequencies in the training set. The normalizing factor $si(f)$ (split info) is included to avoid a bias in favor of features with more values. It represents the amount of information needed to represent all values of the feature (Equation 4). The resulting IG values can then be used as weights in equation 1.

The possibility of automatically determining the relevance of features implies that many different and possibly irrelevant features can be added to the feature set. This is a very convenient methodology if theory does not constrain the choice enough beforehand, or if we wish to measure the importance of various information sources experimentally.

2.2. Optimized MBL in TIMBL: IGTREE

Using information gain rather than Overlap distance to define similarity in IB1 improves its performance on several NLP tasks [13, 26, 24]. The positive effect of information gain on performance prompted us to develop an alternative approach in which the instance memory is restructured in such a way that it contains the same information as before, but in a compressed decision tree structure. In this structure, instances are stored as paths of connected nodes and leaves contain classification information. Nodes are connected via arcs denoting feature values. Information gain is used to determine the order in which instance feature values are added as arcs to the tree. The reasoning behind this compression is that when the computation of information gain points to one feature clearly being the most important in classification, search can be restricted to matching a test instance to those memory instances that have the same feature value as the test instance at that feature. Instead of indexing all memory instances only once on this feature, the instance memory can then be optimized further by examining the second most important feature, followed by the third most important feature, etc. A considerable compression is obtained as similar instances share partial paths.

The tree structure is compressed even more by restricting the paths to those input feature values that disambiguate the classification from all other instances in the training material. The idea is that it is not necessary to fully store an instance as a path when only a few feature values of the instance make the instance classification unique. This implies that feature values that do not contribute to the disambiguation of the instance (i.e., the values of the features with lower information gain values than the lowest information gain value of the disambiguating features) are not stored in the tree.

Apart from compressing all training instances in the tree structure, the IGTREE algorithm also stores with each non-terminal node information concerning the *most probable* or *default* classification given the path thus far, according to the bookkeeping information maintained by the tree construction algorithm. This extra information is essential when processing unknown test instances. Processing an unknown input involves traversing the tree (i.e., matching all feature-values of the test instance with arcs in the order of the overall feature information gain), and either retrieving a classification when a leaf is reached (i.e., an exact match was found), or using the default classification on the last matching non-terminal node if an exact match fails.

In sum, it can be said that in the trade-off between computation during learning and computation during classification, the IGTREE approach chooses to invest more time in organizing the instance base using information gain and compression, to obtain considerably simplified and faster processing during classification, as compared to IB1 and IB1-IG.

The generalization accuracy of IGTREE is comparable to that of IB1-IG; most of the time not significantly differing, and occasionally slightly (but statistically significantly) worse or even better. The two reasons for this surprisingly good accuracy are that (i) most 'unseen' instances contain parts that in fact fully match stored parts of training instances, and (ii) the probabilistic information stored at non-terminal nodes (i.e., the default classifications) still produce strong 'best guesses' when exact matching fails.

3. MEMORY-BASED LANGUAGE ENGINEERING: TIMBL APPLICATIONS

Applying MBL algorithms to NLP tasks (which we will refer to henceforth as memory-based language engineering, MBLE) deviates considerably from traditional, mainstream approaches to language engineering, most particularly in the following three aspects:

1. MBLE models can fit themselves to data automatically using domain-independent heuristics (e.g., from information theory). They are thus essentially task-independent, language-independent and in principle independent of expert knowledge, assuming that the task can be described as a classification problem.

2. Regularities, sub-regularities, and exceptions underlying the NLP task at hand are not modeled explicitly, as is done in traditional rule-based systems, but automatically, using a uniform representation.

3. Analogously, generalization to unseen instances is not modeled by rules, but by best-guess estimation using

similarity-based reasoning with full memory of all instances of the task that the system has seen during learning.

MBLE shares these properties with statistical approaches to NLP. However, because of its computational efficiency and its non-parametric nature, the approach is better equipped than current statistical techniques to handle tasks described with relatively large numbers of features without estimation problems [29].

As described in the Introduction, MBLE treats NLP tasks either as light classification tasks, or as decomposed into multiple light classification subtasks. In this section we illustrate some examples of recent work on MBLE on three 'light' NLP tasks: (i) text-to-speech conversion in TREETALK, (ii) part-of-speech tagging in MBT, and (iii) phrase chunking in MBC.

3.1. TREETALK: Text-to-speech conversion

The TREETALK system [26, 25, 11, 24] has originally been designed for isolated word pronunciation, i.e., converting a written word to its phonemic representation as found in a pronunciation dictionary, and efforts are underway to extend it to modeling speech phenomena in texts, such as sentence accents and prosody. In this subsection we concentrate on word pronunciation; in the research mentioned we have applied MBL to word pronunciation in English, Dutch, Flemish, and French.

We define the word-pronunciation task as the conversion of fixed-sized instances representing a letter and its context to a class representing the phoneme and, if desired (i.e., as input to a speech synthesizer), the stress marker associated with the focus letter. We henceforth refer to the task as GS, an acronym of Grapheme-phoneme conversion and Stress assignment. To generate the instances, windowing is used [22]. Table 1 displays four example instances (of English word pronunciation) and their classifications. Classifications, i.e., phonemes with stress markers, are denoted by composite labels. For example, the first instance in Table 1, _hearts, maps to class label 0A:, denoting an elongated short 'a'-sound which is not the first phoneme of a syllable receiving primary stress. In this study, we chose a fixed window width of seven letters, which offers sufficient context information for adequate performance.

Table 2 lists the generalization accuracy results obtained with the different languages tested, along with the size of the dictionary (number of words) that the respective MBL algorithms were trained and tested on. The accuracy results indicate the percentages of correctly produced phonemes of test words. The error levels produced for all three languages are close to, or fall within, generally accepted boundaries of error tolerance as input for speech synthesis (viz. 2.5% to 5% errors on phonemes [28]). Thus, MBL of word pronunciation using the simple classification task definition as displayed in Table 1 leads to systems performing at a high level of accuracy.

Features							
Left context			Focus	Right context			Class
_	h	e	a	r	t	s	0A:
b	o	o	k	i	n	g	0k
t	i	e	s	_	_	_	0z
_	_	a	f	a	r	_	1f

Table 1: Example of instances of the GS learning task. Instances represent fixed-sized snapshots of a focus (a letter), surrounded by a left and right context (of neighboring letters).

		# Words × 1000		% Correct test
Language	MBL	train	test	phonemes
Dutch	IGTREE	279	31	98.5 ± 0.04
English	IB1-IG	69	7	96.9 ± 0.09
French	IGTREE	18	2	98.3

Table 2: Results for TREETALK on the GS task for different languages. For each language the algorithm used and the size of training and test sets is given, as well as generalization accuracy.

Development of these systems is extremely rapid in the sense that the time required by the learning component is in the order of a few seconds or minutes. Data preprocessing takes some time, depending of the completeness of annotation of the source data. For many languages, well-annotated electronic pronunciation dictionaries are available that can be used for this purpose directly; for example, our English and Dutch data was extracted from the CELEX lexical data base [4]. Preprocessing of this type of data typically does not take more than a few days.

3.2. MBT: Part-of-speech tagging

The MBT tagger-generator [15, 14] takes an annotated corpus as input, and produces a lexicon and memory-based part-of-speech (POS) tagger as output. The problem of POS tagging (morphosyntactic disambiguation) is the following: given a text, provide for each word in the text its contextually disambiguated part of speech (morphosyntactic category). I.e. transform a string of words into a string of tags. E.g., in the sentence *"They can can a can"*, the word *can* is tagged as a modal verb, main verb, and noun respectively. The target category inventory (tag set) may range from extremely simple (order 10) to extremely complex (order 1000). Tagging is a difficult task because of the massive

word	case representation				
	d	d	f	a	t
Pierre	=	=	np	np	np
Vinken	=	np	np	,	np
,	np	np	,	cd	,
61	np	,	cd	nns	cd
years	,	cd	nns	jj-np	nns
old	cd	nns	jj-np	,	jj

Table 3: Example of instances of the POS learning task (known words case base). Instances represent fixed-sized snapshots of a focus (an ambiguous tag), surrounded by a left and right context (of disambiguated tags on the left, and ambiguous tags on the right).

word	case representation						
	p	d	a	s	s	s	t
Pierre	P	=	np	r	r	e	np
Vinken	V	np	,	k	e	n	np
61	6	,	nns	=	6	1	cd
years	y	cd	jj-np	a	r	s	nns
old	o	nns	,	o	l	d	jj

Table 4: Example of instances of the POS learning task (unknown words case base). Instances represent 'morphological' information about the focus word (first letter and the three last letters), surrounded by a left and right context (of one disambiguated tags on the left, and one ambiguous tag on the right).

ambiguity in natural language text. An accurate tagger is instrumental in a large number of language engineering applications (ranging from text-to-speech over parsing to information retrieval). We will refer to this task as the POS task.

The construction of a POS tagger for a specific corpus is achieved in the following way. Given an annotated corpus, three data structures are automatically extracted: a *lexicon* (associating words to possible tags as evidenced in the training corpus), a case base for *known words* (words occurring in the lexicon), and a case base for *unknown words*. Case Bases are compressed using IGTREE for efficiency. During tagging, each word in the text to be tagged is looked up in the lexicon. If it is found, its lexical representation is retrieved and its context is determined, and the resulting pattern is disambiguated using extrapolation from nearest neighbors in the known words case base. When a word is not found in the lexicon, its lexical representation is computed on the basis of its form, its context is determined, and the resulting pattern is disambiguated using extrapolation from nearest neighbors in the unknown words case base. In each case, output is a best guess of the category for the word in its current context.

For known words, cases consist of information about a focus word to be tagged, its left and right context, and an associated category (tag) valid for the focus word in that context. For unknown words, a tag can be guessed only on the basis of the *form* or the *context* of the word. In our lazy learning approach, we provide word form information (especially about suffixes) indirectly to the tagger by encoding the three last letters of the word as separate features in the case representation. The first letter is encoded as well because it contains information about prefix and capitalization of the word. Context information is added to the case representation in a similar way as with known words.

Table 3 and 4 display example instances from the known words and the unknown words case bases respectively.

For English, the complete tagger generation process was performed on a 2 million words training set (lexicon construction and known and unknown words case-base construction), and tested on 200,000 test words, both from the Penn Treebank tagged [18] Wall Street Journal corpus. Generalization performance on known words (96.7%), unknown words (90.6%), and total (96.4%) is competitive with alternative hand-crafted and statistical approaches, and both training and testing speed are excellent (text tagging is possible with a speed of 1200 words per second). In this case, the use of IGTrees as a heuristic approximation to IB-IG did not result in a loss of generalization accuracy while at the same time accounting for a spectacular decrease in memory and time consumption: IGTree retrieval is 100 to 200 times faster than IB-IG retrieval, and uses over 95% less memory.

Language	Tag-set size	# Words × 1000 train	test	% Correct test words
Dutch	13	611	100	95.7
English	44	2000	200	96.4
Spanish	484	711	89	97.8
Czech	42	495	100	93.6

Table 5: Results for the POS task for different languages. The size of the tag-set used, the size of train and test set and the generalization accuracy (combines known and unknown) are given. All taggers use the IGTREE algorithm.

As with the text-to-speech application, tagger development time with these methods is rapid once a suitable annotated training corpus is available. Currently, the MBT approach has been applied to Dutch [14], English [15], Spanish (CRATER Multi-Lingual Aligned Corpus), and Czech (An annotated corpus of newspaper texts from the Institute for the Czech Language). The results are shown in Table 5.

3.3. MBC: Chunking and Bracket Prediction

Phrase chunking is defined as the detection of boundaries between phrases (e.g., noun phrases, verb phrases) in sentences. Chunking can be seen as light parsing. In this sec-

tion we present two applications of MBL to light parsing: NP chunking, and the more complicated task of bracket prediction.

In **NP chunking** sentences are segmented into non-recursive NP's, so called baseNP's [1]. NP chunking can e.g. be used to reduce the complexity of sub-sequential parsing, or to identify named entities for information retrieval. To perform this task, we used the baseNP tag set as presented in [21]: I for inside a baseNP, O for outside a baseNP, and B for the first word in a baseNP following another baseNP. As an example, the IOB tagged sentence: "The/I postman/I gave/O the/I man/I a/B letter/I ./O" will result in the following baseNP bracketed sentence: "[The postman] gave [the man] [a letter]." We applied IB1-IG and IGTREE to a corpus of Wall Street Journal text from the parsed Penn Treebank [18] corpus, using the same train and test set as [21]. A case is constructed for each focus word. The features are words and POS tags of the focus and adjacent words. In one test we used Ramshaw & Marcus' tags as assigned by a Brill tagger [5], and in another test we used the tags assigned by MBT as discussed in section 3.2. As can be seen in the upper half of table 6, the results are comparable, although Ramshaw & Marcus' error-driven transformation-based approach slightly outperforms the memory-based methods. In a second experiment the IOB tags that were predicted in the first experiment were added as features of the context words for a second stage learner. Using IGTREE this improved the correct score up to 98.0%, see the lower part of table 6. Such a high accuracy combined with the speed of IGTREE offers an attractive preprocessing stage for parsing.

In **bracket prediction** a full parse tree is approximated by combination of local predictions. The task is to predict the sequence of (closing and opening) brackets preceding a word. Closing all pending open brackets at the end of the sentence suffices to construct an unlabeled parse tree of the sentence out of the sequence of predictions. The input of the bracket predictor is a tagged sentence (e.g. The/DT men/NN liked/VBD him/PP), the output an unlabeled, bracketed sentence (e.g. [[[The][men]][[liked][[him]]]].). The work on this task is still in progress, although the preliminary results show that the 'light module' approach can be taken to interesting length. As a training set we used four sections of parsed WSJ text, and we tested on another section. Again, a case is constructed for each focus word with locally adjacent words and POS tags as features. The experiment shows that the bracket prediction task, while intuitively different from the classification paradigm (i.e. the number of different classes is unbounded, in theory) and rather complex, produces over 80% correctly predicted bracket sequences and around 25% sentences correct (exact unlabeled match).

Algorithm	POS tagger	second phase	% Correct test words
R&M	Brill	–	97.4
IB1-IG	Brill	–	97.2
IB1-IG	MBT	–	97.3
IGTREE	MBT	–	96.8
IGTREE	MBT	IOB	98.0

Table 6: Results of NP chunking, on the same dataset as Ramshaw and Marcus (1996), their results are given in row 1. The measured score is percentage correct predicted IOB-tags. IOB in the column "second phase" indicates that IOB predictions of surrounding words are taken into account as features for a second phase (see text).

4. WHY MBL WORKS FOR NLP TASKS

We presented MBLE applications that show sufficient to excellent generalization performance on NLP tasks that can be considered light, but are relatively complex in their own right. Furthermore, we showed that they could be developed rapidly as soon as data is available. In this section we provide arguments in favor of using MBL algorithms over other generic learning algorithms, specifically decision-tree learning. The arguments relate to inherent characteristics of NLP data in general.

One of the most salient characteristics of natural language processing mappings is that from the point of view of a *rule-based* approach to the task, these mappings are noisy and complex with, apart from some regularities, also many sub-regularities and (pockets of) exceptions. In other words, apart from a core of generalizations, there is a relatively large periphery of irregularities [10]. In rule-based NLP, this problem has to be solved using mechanisms such as rule ordering, subsumption, inheritance, or default reasoning (in linguistics this type of "priority to the most specific" mechanism is called the *elsewhere condition*). In the feature-vector-based classification approximations of these complex language processing mappings, as employed by MBLE, this property is reflected in the high degree of disjunctivity of the instance space: classes exhibit a high degree of polymorphism. One way to visualize this disjunctivity is by looking at the average number of *friendly neighbors* for each instance in a leave-one-out experiment [27].

For each instance in the GS, POS, and CHUNK data sets a distance ranking of the 50 nearest neighbors to an instance was produced. In case of ties in distance, nearest neighbors with an identical class as the left-out instance are placed higher in rank than instances with a different class. Within this ranked list we count the ranking of the nearest neighbor of a different class. This rank number minus one is then taken as the cluster size surrounding the left-out in-

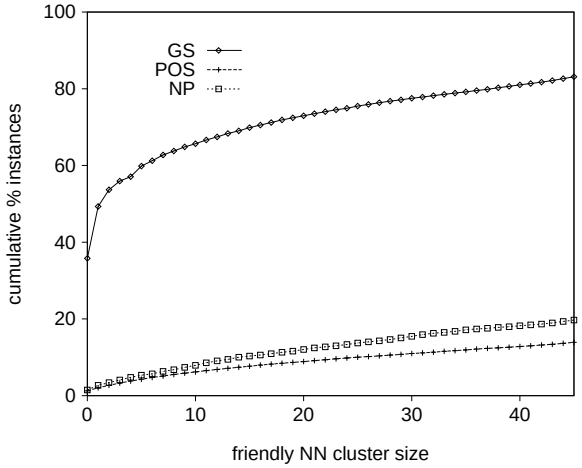

Figure 2: Cumulative percentages of occurrences of friendly-neighbor clusters of sizes 0 to 45, as found in the GS, POS, and NP data sets.

stance. If, for example, a left-out instance is surrounded by three instances of the same class at distance 0.0 (i.e., no mismatching feature values), followed by a fourth nearest-neighbor instance of a different class at distance 0.3, the left-out instance is said to be in a cluster of size three. The results of the three leave-one-out experiments are displayed graphically in Figure 2. The x-axis of Figure 2 denotes the numbers of friendly neighbors found surrounding instances; the y-axis denotes the cumulative percentage of occurrences of friendly-neighbor clusters of particular sizes.

The cumulative percentage graphs in Figure 2 display that for the case of the GS task, many instances have only a handful of friendly neighbors; 59.9 % of the GS instances have five friendly neighbors or less, while 35.8 % has no friendly neighbors at all. Instances of the POS and the NP task tend to have more friendly neighbors surrounding them. In sum, the GS task appears to display very high disjunctivity (i.e., a high degree of polymorphism) of its 159 classes; for the other two tasks, disjunctivity appears to be slightly lower, but still the classes are scattered across many unconnected clusters in the instance space.

Given these properties of NLP datasets, a learning method should be able to represent and recognize the 'different faces' of the categories to be learned, in order to be useful. MBL is particularly well-suited for this task, as it is based on the premise that all information in the training data is potentially relevant. The information-gain weighted similarity metric used in IB1 and approximated by IGTREE has the following two effects: (i) the more overlap there is with a new pattern to be classified, the more chance a training item has in contributing to the decision about the category of the new pattern (this implements a specificity ordering), and (ii) the information-gain weighting modifies the specificity ordering according to the relevance of the parts matching. The combined effect is that the extrapolation is always based on the most specific relevant training instances in the training set (including low-frequency items), and that back-off to more general relevant instances is automatic [29].

Statistical approaches and *eager* machine learning methods such as rule and decision tree induction methods on the other hand, often need to abstract from the infrequent patterns (e.g. by using pruning methods), in order to avoid overfitting, thereby losing some of the training data as a basis for extrapolation.

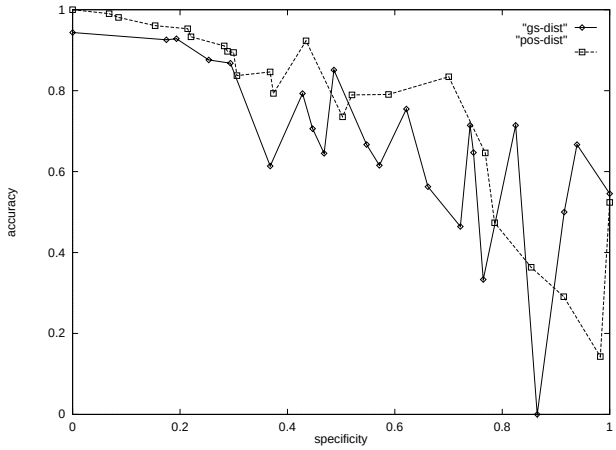

Figure 3: Percentage correct for two of our datasets plotted as a function of specificity. Specificity is the distance between a test item and its nearest neighbor. The distances are normalized to be between zero and one to make a comparison across data sets possible.

Figure 3 shows why this elimination of specific data can be harmful. In this figure the percentage correct for two of our datasets is plotted as a function of specificity. The decrease of the accuracy seen in the graph clearly confirms the intuition that an extrapolation from a more specific support set is more likely to be correct. Reasoning in the other direction, it suggests that any forgetting of specific information from the training set will push at least some test items in the direction of a less specific support set, and thus of lower accuracy.

5. CONCLUSIONS

We have shown that MBL makes the rapid development of accurate NLP modules possible. The approach is based on the definition of light language processing tasks as classification problems, on the storage of example classifications, and the use of similarity-based reasoning and an information-theoretic relevance weighting technique for solving new problems. Efficient approximations of the approach such as IGTREE were shown to operate with competitive efficiency and accuracy, at extremely rapid development times.

We have constructed several of these modules, as described in this paper, and are currently extending the approach to more languages and more NLP tasks (word sense disambiguation, subcategorization frame induction, light parsing, sentence accent computation, intonation phrase chunking, structural disambiguation, etc.)[2].

A next step in our research strategy is to combine and integrate several of these *light* modules into more complex language processing systems. E.g. we are at present working on the integration of word pronunciation, POS tagging, and chunking modules into a sentence-level text-to-speech system which also computes intonation and sentence accent, and on a text analysis system which combines tagging, chunking, subcategorization, and structural disambiguation modules.

Finally, from the results in this paper and those reported earlier [10, 24], it appears that the properties of NLP tasks are such that non-abstracting approaches such as MBL are at an advantage in learning from the highly disjunctive datasets typical in NLP in contrast to eager methods such as decision tree induction and statistical inference.

Acknowledgments

We thank Peter Berck and the members of the CNTS group in Antwerp for their input and discussions, and Prof. Jiri Kraus of the Czech Academy of Sciences for permission to use the Czech POS-annotated corpus.

6. REFERENCES

[1] Steven Abney. Parsing by chunks. In *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht, 1991.

[2] D. W. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[3] Lazy learning: Special issue editorial. *Artificial Intelligence Review*, 11:7–10, 1997.

[4] R. H. Baayen, R. Piepenbrock, and H. van Rijn. *The CELEX lexical data base on CD-ROM*. Linguistic Data Consortium, Philadelphia, PA, 1993.

[5] Eric Brill. Some advances in transformation-based part-of-speech tagging. In *AAAI'94*, 1994.

[6] Claire Cardie. Automatic feature set selection for case-based learning of linguistic knowledge. In *Proc. of Conference on Empirical Methods in NLP*. University of Pennsylvania, 1996.

[2]For more information, publications, and web demos, see the ILK homepage: `http://ilk.kub.nl/`

[7] S. Cost and S. Salzberg. A weighted nearest neighbour algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.

[8] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27, 1967.

[9] W. Daelemans. Memory-based lexical acquisition and processing. In P. Steffens, editor, *Machine Translation and the Lexicon*, volume 898 of *Lecture Notes in Artificial Intelligence*, pages 85–98. Springer-Verlag, Berlin, 1995.

[10] W. Daelemans. Experience-driven language acquisition and processing. In M. Van der Avoird and C. Corsius, editors, *Proceedings of the CLS Opening Academic Year 1996-1997*, pages 83–95. CLS, Tilburg, 1996.

[11] W. Daelemans and A. Van den Bosch. Language-independent data-oriented grapheme-to-phoneme conversion. In J. P. H. Van Santen, R. W. Sproat, J. P. Olive, and J. Hirschberg, editors, *Progress in Speech Processing*, pages 77–89. Springer-Verlag, Berlin, 1996.

[12] W. Daelemans, A. Van den Bosch, and A. Weijters. IGTree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423, 1997.

[13] Walter Daelemans and Antal van den Bosch. Generalisation performance of backpropagation learning on a syllabification task. In M. F. J. Drossaers and A. Nijholt, editors, *Proc. of TWLT3: Connectionism and Natural Language Processing*, pages 27–37, Enschede, 1992. Twente University.

[14] Walter Daelemans, Jakub Zavrel, and Peter Berck. Part-of-speech tagging for dutch with mbt, a memory-based tagger generator. In K. van der Meer, editor, *Informatiewetenschap 1996, Wetenschappelijke bijdrage aan de Vierde Interdisciplinaire Onderzoeksconferentie Informatiewetenchap*, pages 33–40, The Netherlands, 1996. TU Delft.

[15] Walter Daelemans, Jakub Zavrel, Peter Berck, and Steven Gillis. 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, 1996.

[16] P. .A. Devijver and J. Kittler. *Pattern recognition. A statistical approach*. Prentice-Hall, London, UK, 1982.

[17] J. Kolodner. *Case-based reasoning*. Morgan Kaufmann, San Mateo, CA, 1993.

[18] M. Marcus, B. Santorini, and M.A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[19] J.R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–206, 1986.

[20] J.R. Quinlan. C4.5*: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[21] L.A. Ramshaw and M.P. Marcus. Text chunking using transformation-based learning. In *Proc. of third workshop on very large corpora*, pages 82–94, June 1995.

[22] T. J. Sejnowski and C. S. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168, 1987.

[23] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the* ACM, 29(12):1213–1228, December 1986.

[24] A. Van den Bosch. *Learning to pronounce written words: A study in inductive language learning*. PhD thesis, Universiteit Maastricht, 1997.

[25] A. Van den Bosch, A. Content, W. Daelemans, and B. De Gelder. Measuring the complexity of writing systems. *Journal of Quantitative Linguistics*, 1(3), 1995.

[26] A. Van den Bosch and W. Daelemans. Data-oriented methods for grapheme-to-phoneme conversion. In *Proceedings of the 6th Conference of the EACL*, pages 45–53, 1993.

[27] S. Weiss and C. Kulikowski. *Computer systems that learn*. San Mateo, CA: Morgan Kaufmann, 1991.

[28] F. Yvon. *Prononcer par analogie: motivation, formalisation et évaluation*. PhD thesis, Ecole Nationale Supérieure des Télécommunication, Paris, 1996.

[29] J. Zavrel and W. Daelemans. Memory-based learning: Using similarity for smoothing. In *Proc. of 35th annual meeting of the ACL*, Madrid, 1997.