Memory-Based Learning

Walter Daelemans[1] and Antal van den Bosch[2]

[1] CLiPS Research Group, University of Antwerp, Prinsstraat 13, 2000 Antwerpen, Belgium. walter.daelemans@ua.ac.be

[2] ILK / Tilburg centre for Creative Computing, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands. antal.vandenbosch@uvt.nl

A draft chapter for the Blackwell *Computational Linguistics and Natural Language Processing Handbook*, edited by Alex Clark, Chris Fox and Shalom Lappin. This draft formatted on 12th June 2009.

1 Introduction

Most Natural Language Processing (NLP) tasks require the translation of one level of representation to another. For example, in text to speech systems, it is necessary to have a component that translates the spelling representation of words to a corresponding phonetic representation; in part of speech (POS) tagging, the words of a sentence are translated into their contextually appropriate POS tags. Some tasks in NLP involve segmentation: identifying the syllable boundaries in a word or the syntactic phrases in a sentence are examples of such *chunking* tasks. Other tasks, such as document categorization and word sense disambiguation require a choice between a limited number of possibilities.

What all these types of NLP tasks have in common, is that they can be formulated as a *classification task*, and are therefore appropriate problems for discriminative supervised Machine Learning methods. With some effort, even tasks like coreference resolution and machine translation can be cast as a classification problem. In this Chapter, we will see an assortment of examples of NLP problems formulated as classification-based learning.

Classification-based learning starts from a set of instances (examples) consisting each of a set of input features (a feature vector) and an output class. For example, for the NLP task of predicting the pronunciation of a word, given a number of words with their phonetic transcription as training material, we could create an instance for each letter, as in Table 1. One of the input features is the letter to be transcribed (here indicated as the focus feature) and other features would be the spelling symbols before and after the focus; in this case a context of three such symbols to the left and to the right are used to make a total of seven predictive features. The output class is the phoneme corresponding with the focus letter in that context. Data like this can be used as training material to construct a *classifier* that is subsequently used to classify feature vectors belonging to new words, not part of the training data. In this way, the classifier generalizes from the original training data, which is the purpose of machine learning.

Instance number	Left context	Focus letter	Right context	Classification
1	_ _ _	b	o o k	b
2	_ _ b	o	o k i	–
3	_ b o	o	k i n	u
4	b o o	k	i n g	k
5	o o k	i	n g _	I
6	o k i	n	g _ _	–
7	k i n	g	_ _ _	N

Table 1. Examples generated for the letter-phoneme conversion task, from the word-phonemization pair *booking* – [bukIN], aligned as [b-ukI-N].

Memory-Based Learning (MBL) is one of the techniques that has been proposed to learn these NLP classification problems. Many other techniques for supervised classification-based learning exist. See CHAPTER 6 MAXIMUM ENTROPY, CHAPTER 8 DECISION TREES, CHAPTER 9 INDUCTIVE LOGIC PROGRAMMING, CHAPTER 11 ARTIFICIAL NEURAL NETWORKS. In this Chapter, we will show how MBL differs from these approaches.

MBL has as its defining characteristic that it stores in memory all available instances of a task, and that it extrapolates from the most similar instances in memory to solve problems for which no solution is present in memory. What the most similar instances (the *nearest neighbours*) are, is defined by an adaptive *similarity metric*. The general principle is well-known in Artificial Intelligence and cognitive psychology, and can be found under different labels (case-based reasoning, exemplar-based models, k-NN, instance-based learning, memory-based reasoning, etc.). The approach has been used in application areas ranging from vision and speech via expert systems to robotics and models of human categorization.

In the remainder of this Chapter, we introduce an operationalization of MBL, implemented in the open source software package TiMBL in Section 2. Applications in computational linguistics and computational psycholinguistics are discussed in Sections 3 and 4, respectively. We then move to a discussion of the strengths and limitations of the approach in Section 5, and show how FAMBL, a variant of MBL based on careful abstraction, discussed in Section 6, can strike a balance between abstraction and memory.

2 Memory-Based Language Processing

MBL, and its application to NLP, which we will call Memory-Based Language Processing (MBLP) here, is based on the idea that learning and processing are two sides of the same coin. Learning is the storage of examples in memory, and processing is similarity-based reasoning with these stored examples. The approach is inspired by work in pre-Chomskyan linguistics, categorization psychology, and statistical pattern recognition. The main claim is that, contrary to majority belief since Chomsky, generalization (going beyond the data) can also be achieved without formulating abstract representations such as rules. Abstract representations such as rules, decision trees, statistical models, and trained artificial neural networks forget about the data itself, and only keep the abstraction. Such *eager learning* approaches are usually contrasted with table lookup, a method that obviously cannot generalize. However, by adding similarity-based reasoning to table lookup, *lazy learning* approaches such as MBL are capable of going beyond the training data as well, and on top of that keep all the data available. This is arguably a useful property for NLP tasks: in such tasks, low-frequency or atypical examples are often not noise to be abstracted from in models, but on the contrary an essential part of the model. In the remainder of this Section, we will describe a particular instantiation of memory-based approaches, MBLP, that we have found to work well for language processing problems and for which we make available open source software (TiMBL). The approach is a combination and extension of ideas from Instance Based Learning (Aha *et al.*, 1991)) and Memory-Based Reasoning (Stanfill & Waltz, 1986)) and a direct descendent of the k-NN algorithm (Fix & Hodges, 1951; Cover & Hart, 1967).

2.1 MBLP: An operationalization of MBL

An MBLP system has two components: a *learning component* which is memory-based, and a *performance component* which is similarity-based. The learning component is memory-based as it involves storing examples in memory without abstraction, selection, or restructuring. In the performance component of an MBLP system the stored examples are used as a basis for mapping input to output; input instances are classified by assigning them an output label. During classification, a previously unseen test instance is presented to the system. The class of this instance is determined on the basis of an extrapolation from the most similar example(s) in memory. There are different ways in which this approach can be operationalized. The goal of this section is to provide a clear definition of the operationalizations we have found to work well for NLP tasks. TiMBL is an open source software package implementing all algorithms and metrics discussed here[1].

First, a visual example serves to illustrate the basic concepts of memory-based or k-nearest neighbor classification. The left part of Figure 1 displays part of a two-dimensional Euclidean space with three examples labeled black (i.e. they are examples of the class "black"), and three examples labeled white. Each example's two coordinates are its two numeric feature values. An example occupies a piece of the space, a Voronoi tile, in which it is the closest example. The so-called Voronoi tesselation depicted in the left part of Figure 1 is essentially a map of the decision boundaries of the 1-nearest neighbor classification rule: the tile on which a new instance is positioned determines the single nearest neighbor, and the subsequent classification step simply copies

[1] The software, reference guide, and instructions on how to install it can be downloaded from `http://ilk.uvt.nl/timbl`

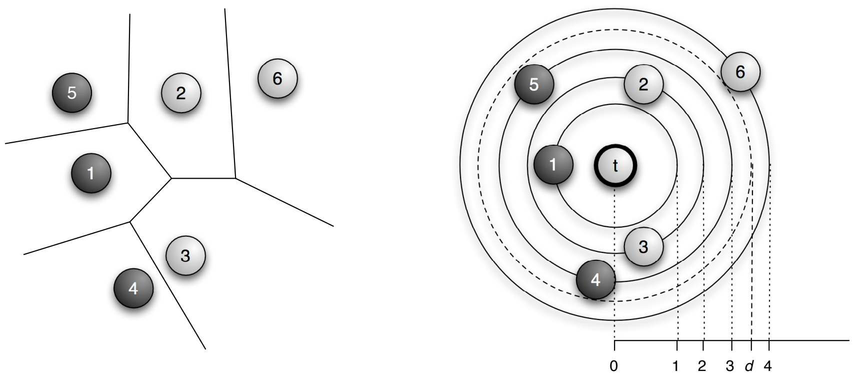

Figure 1. An example 2D space with six examples labeled white or black. Left: the Voronoi tesselation of the space. Right: around a new test item t, nearest neighbors are found at four different distances; some examples are equidistant. The parameter k can either regulate the number of nearest neighbors or distances. Alternatively, a distance d can specify the circle (Parzen window) within which nearest neighbors are sought.

the class label of that nearest neighbor (here, black or white) to the new instance.

Rather than pre-computing the Voronoi tesselation, which is restricted to be used for single nearest-neighbor classification, the common mode of operation of the more generic k-nearest neighbor classifier is to perform a search for the nearest examples around each new instance t to base a classification on. The key parameter k determines the number of examples within an expanding circle (or hyperball) around the new instance. This can either be the actual number of examples found while extending outwards, or the number of distance rings on which equi-distant examples are found. In Figure 1, the six visible examples are found at four different distances. Alternatively, a distance d can be specified as the fixed size of the hyperball or Parzen window (Parzen, 1962) in which nearest neighbors are sought. Using Parzen windows implies ignoring the local example density; a Parzen window may contain no examples or all examples. In contrast, the k-nearest neighbor approach in its most basic form ignores the actual distance at which the k nearest neighbors are found, and adapts the hyperball to the local example density around the new instance. In the remainder of this chapter we adopt the k-nearest neighbor approach, and show how the distance of the target to different neighbors can be factored into the classification.

As a sidenote, the k-nearest neighbor classifier has some strong formal consistency results. With $k = 1$, the classification rule is guaranteed to yield an error rate no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data) as the amount of data approaches infinity (Cover & Hart, 1967). Another useful property of the classifier is its insensitivity to the number of classes; this number is neither a factor in learning (storage) nor in classification.

Abstracting over the particular type of feature spaces (such as Euclidean space in the example of Figure 1), the similarity between a new instance X and all examples Y in memory is computed using a *similarity metric* (that actually measures distance) $\Delta(X, Y)$. Classification works by assigning the most frequent class within the k most similar example(s) as the class of a new test instance.

The most basic metric that works for instances with symbolic features such as many datasets in language and speech processing is the *overlap metric* given in Equations 1 and 2; where $\Delta(X, Y)$ is the distance between instances X and Y, represented by n features, and δ is the distance per feature. The distance between two patterns is simply the sum of the differences between the features. In the case of symbolic feature values, the distance is 0 with an exact match, and 1 with a mismatch. The k-NN algorithm with this metric is called IB1 in Aha *et al.* (1991).

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

where:

$$\delta(x_i, y_i) = \begin{cases} \left|\frac{x_i - y_i}{max_i - min_i}\right| & \text{if numeric, otherwise} \\ 0 & \text{if } x_i = y_i \\ 1 & \text{if } x_i \neq y_i \end{cases} \tag{2}$$

Our definition of this basic algorithm is slightly different from the IB1 algorithm originally proposed by Aha *et al.* (1991). The main difference is that in our version the value of k refers to k-nearest distances rather than k-nearest examples. As illustrated in the right-hand side of Figure 1, Several examples in memory can be equally similar to a new instance. Instead of choosing one at random, all examples at the same distance are added to the nearest-neighbour set.

The distance metric in Equation 2 simply counts the number of (mis)matching feature-values in two instances being compared. In the absence of information about feature relevance, this is a reasonable choice. Otherwise, we can use domain knowledge to weight or select different feature. We can also compute statistics about the relevance of features by looking at which features are good predictors of the class labels, using feature weighting methods such as Information Gain.

Information gain (IG) weighting looks at each feature in isolation, and estimates how much information it contributes to our knowledge of the correct class label. The information gain estimate of feature i is measured by computing the difference in uncertainty (i.e., entropy) between the situations without and with knowledge of the value of that feature (the formula is given in Equation 3), where C is the set of class labels, V_i is the set of values for feature i, and $H(C) = -\sum_{c \in C} P(c) \log_2 P(c)$ is the entropy of the class labels. IG is used in decision tree learning (CHAPTER 8 DECISION TREES) as a splitting criterion.

$$w_i = H(C) - \sum_{v \in V_i} P(v) \times H(C|v) \tag{3}$$

The probabilities are estimated from relative frequencies in the training set. For numeric features, an intermediate step needs to be taken to apply the symbol-based computation of IG. All real values of a numeric feature are temporarily discretized into a number of intervals. Instances are ranked on their real value, and then spread evenly over the intervals; each interval contains the same number of instances (this is necessary to avoid empty intervals in the case of skewed distributions of values). Instances in each of these intervals are then used in the IG computation as all having the same unordered, symbolic value per group. Note that this discretization is only temporary; it is not used in the computation of the distance metric.

The IG weight of a feature is a probability-weighted average of the informativeness of the different values of the feature. This makes the values with low frequency but high informativity invisible. Such values disappear in the average. At the same time, this also makes the IG weight robust to estimation problems in sparse data. Each parameter (weight) is estimated on the whole data set.

A well-known problem with IG is that it tends to overestimate the relevance of features with large numbers of values, MBLP therefore also includes the *gain ratio* normalization and several alternative feature relevance weighting methods (chi-squared, shared variance, special metrics for binary features etc.).

The choice of representation for instances in MBLP is the key factor determining the accuracy of the approach. The feature values and classes in NLP tasks are often represented by symbolic labels. The metrics that have been described so far, i.e., (weighted) overlap, are limited to either a match or a mismatch between feature values. This means that all values of a feature are seen as equally dissimilar to each other. However, we would like to express that some feature *value* pairs are more or less similar than other pairs. E.g., we would like vowels to be more similar to each other than to consonants in problems where features are letters or phonemes, nouns more similar to other nouns than to verbs in problems where features are words, etc. As with feature weights, domain knowledge can be used to create a feature system expressing these similarities, e.g., by splitting or collapsing features. But again, an automatic technique might be better in modeling these statistical relations.

For such a purpose a metric was defined by Stanfill & Waltz (1986) and further refined by Cost & Salzberg (1993). It is called the (modified) (MVDM; equation 4), a method to determine the similarity of the values of a feature by looking at co-occurrence of values with target classes. For the distance between two values $v_1,\ v_2$ of a feature, we compute the difference of the conditional distribution of the classes $C_{1...n}$ for these values.

$$\delta(v_1, v_2) = \sum_{i=1}^{n} |P(C_i|v_1) - P(C_i|v_2)| \tag{4}$$

MVDM differs considerably from overlap-based metrics in its composition of the nearest-neighbor sets. Overlap causes an abundance of ties in nearest-neighbor position. For example, if the nearest neighbor is at a distance of one mismatch from the test instance, then the nearest-neighbor set will contain the entire partition of the training set that contains *any* value for the mismatching feature. With the MVDM metric, however, the nearest-neighbor set will either contain patterns which have the value with the lowest $\delta(v_1, v_2)$ in the mismatching position, or MVDM will select a totally different nearest neighbor which has less exactly matching features, but a smaller distance in the mismatching features (Zavrel & Daelemans, 1997).

MBLP also contains different metrics for extrapolation from nearest neighbors (linear or exponential distance-based decay) and for computing exemplar similarity with weighted examples. Such weights could be based on frequency of instances, or on their goodness or typicality according to some criterion. MBLP is not a new algorithm, rather, it is a set of algorithm parameterizations selected and optimised for use with language processing data. We will not go into further details of MBLP here. However, we will return to the crucial discussion about generalization and abstraction in lazy and eager learning methods in Section 6. First we provide an overview of application areas of MBLP.

3 NLP Applications

As explained in Section 1, MBL shares its generic applicability to classification tasks with any other machine learning classifier. Hence, when an NLP task is framed as a classification task, memory-based learning can be applied to it. In the past decade, memory-based learning has indeed been applied across a wide range of NLP tasks. Before we turn to the limitations of memory-based learning in Section 5, we provide an overview of types of NLP tasks in which memory-based learning has been successful in this and the next Section.

Morpho-phonology

Tasks at the phonological and morphological levels are often framed as sliding-window tasks over sequences of letters or phonemes, where the task is framed as a mapping of one symbol set to another (letters to phonemes), or a mapping from an unsegmented string to a segmented string (words to morphological analyses). In case of segmentation tasks such as syllabification, the output symbol set typically consist of a "null" value that signifies that no boundary occurs at the focus input symbol, and one or more positive values marking that some type of boundary does occur at the focus letter. Example morpho-phonological tasks to which memory-based learning has been applied are hyphenation and syllabification (Daelemans & Van den Bosch, 1992); grapheme-to-phoneme conversion (Van den Bosch & Daelemans, 1993; Daelemans & Van den Bosch, 1996); and morphological analysis (Van den Bosch & Daelemans, 1999; De Pauw *et al.*, 2004). Although these examples are applied mostly to Germanic languages (English, Dutch, and German), applications to other languages with more complicated writing systems or morphologies, or with limited resources, have also been presented: for example, letter-phoneme conversion in Scottish Gaelic (Wolters & Van den Bosch, 1997), morphological analysis of Arabic (Marsi *et al.*, 2006), or diacritic restoration in languages with a diacritic-rich writing system (Mihalcea, 2002; De Pauw *et al.*, 2007).

Most of these studies report the important advantage of the memory-based approach to faithfully reproduce all training data; essentially, the method can be seen as a compressed lexicon that also generalizes to unseen words if needed. As an average training lexicon typically covers unseen text at about 95% (i.e. 5% of the words in a new text are not in the lexicon), the key goal of the memory-based learner is to process the 5% unknown words as accurately as possible. In the reported studies, most attention is indeed paid to evaluating the classifiers' generalization performance on unseen words, often at the word level. Actual percentages are intrinsically linked to the task, the language, and the amount of training data, and can typically only be assessed properly in the context of a higher-level task, such as comparative human judgements of the understandability of a speech synthesizer with and without the module under evaluation.

Syntacto-semantics

In the mid-1990s, memory-based learning was among the early set of machine-learning classifiers to be applied to tasks in shallow parsing and lexical semantics: part-of-speech tagging (Daelemans *et al.*, 1996; Zavrel & Daelemans, 1999; Van Halteren *et al.*, 2001) and PP-attachment (Zavrel *et al.*, 1997), mostly on English benchmark tasks. Also, early developments of shallow parsing modules using memory-based learning contributed to the development of the field of shallow parsing: subcategorization (Buchholz, 1998); phrase chunking (Veenstra, 1998; Tjong Kim Sang & Veenstra, 1999); and the integration of memory-based modules for shallow parsing (Daelemans *et al.*, 1999a; Buchholz *et al.*, 1999; Yeh, 2000). More recently, memory-based learning has been integrated as a classifier engine in more complicated dependency parsing systems (Nivre *et al.*, 2004; Sagae & Lavie, 2005; Canisius *et al.*, 2006).

Memory-based learning has been applied succesfully to lexical semantics, in particular to word sense disambiguation (Veenstra *et al.*, 2000; Stevenson & Wilks, 1999; Kokkinakis, 2000; Mihalcea, 2002; Hoste *et al.*, 2002; Decadt *et al.*, 2004), but also in other lexical semantic tasks such as determining noun countability (Baldwin & Bond, 2003), animacy (Orăsan & Evans, 2001), and semantic relations within noun compounds (Kim & Baldwin, 2006; Nastase *et al.*, 2006).

Text analysis

Extending the simple sliding-window approach that also proved to be useful in phrase chunking, memory-based learning has also been used for named-entity recognition (Buchholz & Van den Bosch, 2000; Hendrickx & Van den Bosch, 2003; De Meulder & Daelemans, 2003; Sporleder *et al.*, 2006; Leveling & Hartrumpf, 2007), and domain-dependent information extraction (Zavrel *et al.*, 2000; Zavrel & Daelemans, 2003; Ahn, 2006).

Many NLP tasks beyond the sentence level tend not to be phrased (or phrasable) in simple sliding-window representations. Some tasks require more complicated structures, such as pairs of phrases in their context bearing some relation to be classified, as in anaphora and coreference resolution (Preiss, 2002; Mitkov *et al.*, 2002; Hoste, 2005), while other tasks appear to be best solved using vector space or bag-of-words representations, to which memory-based learning is also amenable, such as text classification (Spitters, 2000), question classification (Cumbreras *et al.*, 2006; Dridan & Baldwin, 2007), or spam filtering (Androutsopoulos *et al.*, 2000).

Dialogue and discourse

In the field of discourse and dialogue modeling, memory-based learning has been used for shallow semantic analysis of speech-recognised utterances (Gustafson *et al.*, 1999; Van den Bosch *et al.*, 2001; Lendvai *et al.*, 2002, 2003a; Lendvai & Geertzen, 2007), in disfluency detection in transcribed spontaneous speech (Lendvai *et al.*, 2003b), and in classifying ellipsis in dialogue (Fernández *et al.*, 2004). In most of these studies, the task is framed as a classification task into a limited number of labels (usually, some dialogue act labeling scheme), while the input can be a mix of bag-of-word features, dialogue history features (e.g. previous dialogue acts), and acoustic features of recognized speech in the context of spoken dialogue systems. As memory-based learning handles numeric features as easily as symbolic features, it is unproblematic to mix these heterogeneous feature sets in a single classifier.

Generation, language modeling, and translation

While the general scope of natural language generation, language modeling, and translation comprises full sequences, memory-based learning has been applied to word or phrase-level subtasks within these more general problem fields. For instance, in natural language generation, memory-based learning has been applied particularly to morpho-syntactic generation subtasks: inflection generation, such as diminutive formation (Daelemans *et al.*, 1998), article generation (Minnen *et al.*, 2000), or determining the order of multiple prenominal adjectives (Malouf, 2000).

Language modeling has mostly been the domain of stochastic n-gram models, but as (Zavrel & Daelemans, 1997) already showed, there is an equivalence relation between back-off smoothing in n-gram models and memory-based classification. Essentially, language modeling in n-gram models can be phrased as the classification task of predicting the next word given a context of previous words. Indeed, memory-based language models can be developed that perform this task (Van den Bosch, 2006a). As a specialization of these generic language models, memory-based confusable-specific disambiguators can be trained to determine which of a confusible set of words (e.g. *to*, *too*, and *two*) is appropriate in a certain context. An accurate confusable disambiguator can be useful as a spelling correcting module in a proofing environment.

In machine translation, memory-based learning bears a close relation with example-based machine translation (EBMT). A first EBMT-implementation using memory-based learning is described in (Van den Bosch *et al.*, 2007). Analogous to memory-based language modeling, memory-based translation maps a local context of words (a part of a source-language sentence) to target word or n-gram of words (part of the corresponding target sentence), where the target word or centre of the target n-gram is aligned to the source word according to an externally computed word alignment.

We have not tried to be exhaustive in this Section. There are other implementations of k-nearest neighbor classification apart from TiMBL that have been used in NLP, and alternative memory-based algorithms have been proposed for specific tasks. As a good example, Bob Damper and colleagues have developed a psycholinguistic proposal for modeling pronunciation (Pronunciation by Analogy) into a state of the art grapheme-to-phoneme conversion approach (Damper & Eastmond, 1997). Other researchers have argued for richer analogy processes in memory-based approaches than the basic overlap metric and its extensions that is used in the research described in this Section (Pirrelli & Yvon, 1999; Yvon & Stroppa, 2007; Lepage & Denoual, 2005). This work is also relevant when memory-based approaches are intended as models of human language acquisition and processing as in the work we turn to next.

4 Exemplar-Based Computational Psycholinguistics

From the time Chomsky substituted the vague notions of analogy and induction existing in linguistics in his time (in work of e.g., de Saussure, Bloomfield and Harris) by a better formalised notion of rule-based grammars, most mainstream linguistic theories, even the functionally and cognitively inspired ones, have assumed rules to be the only or main means to describe any aspect of language. Also in computational modeling of human language processing and human language acquisition, mental rule application and acquisition has been the standard approach. See CHAPTER COMPUTATIONAL PSYCHOLINGUISTICS. A good example is the dual mechanism model advocated by (Pinker, 1999) and others for inflectional morphology. In such a model, a mental rule governing the regular cases in inflectional morphology is complemented by an associative memory explaining subregularities and exceptions. In contrast, single mechanism models (mostly based on neural network approaches following (Rumelhart & McClelland, 1986)) model regular and exceptional language behaviour in a single model. See CHAPTER ARTIFICIAL NEURAL NETWORKS.

MBLP can be considered an operationalisation of the pre-Chomskyan analogical approach to language, and as a predictive model for human language acquisition and processing that is an alternative to both rule-based and neural network approaches. The main advantage from a theoretical point of view is that no ontological distinction has to be made between regular and exceptional cases, and that the gradedness of language learning and processing is an emergent phenomenon of the way the model works. The approach is also incremental, in that the addition of new experience immediately affects processing without any need of recomputation of knowledge structures. Conceptually, to model language acquisition and processing, memorized experiences of previous language use are searched looking for instances similar to a new item, and a decision is extrapolated for the new item from these nearest neighbours. Language acquisition is simply the incremental storage of experience.

The Analogical Modeling (AM) approach of Skousen (1989, 1992, 2002) is an early alternative example of a computational operationalization of analogy in a memory-based context and its application in modeling language. It is memory-based in that all available training data (experience) is used in extrapolating to the solution for a new input. As it searches combinatorial combinatons of input features, it is exponential in the number of features, which makes the approach impractical for problems with many features. The approach has been applied to different problems in language processing, mainly in the phonology and morphology domains. Although algorithmically very different from and more costly than MBLP (which is linear in the number of features), empirical comparisons have never shown important accuracy or output differences between AM and MBLP (Eddington, 2002a; Daelemans, 2002; Krott *et al.*, 2002).

Inflectional morphology has proven a useful and interesting testing ground for models of language acquisition and processing because of the relative simplicity of the processes (compared to syntax), the availability of lexical databases, and the ample psycholinguistic experimental data in the form of accounts of acquisition, adult processing experiments, production tasks on pseudowords etc. This makes possible controlled comparisons between different computational models. Problems like English past tense formation, German and Dutch plural formation etc. have therefore become important benchmark problems. Memory-based psycholinguistic models of inflectional morphology have been provided for the English past tense by Keuleers (2008), for Dutch plural formation by Keuleers *et al.* (2007); Keuleers & Daelemans (2007), for Spanish diminutive formation by Eddington (2002c), and for Dutch and German linking phenomena in compounds by Krott *et al.* (2001); A. *et al.* (2007). See (Hay & Baayen, 2005) for an overview of the state of the art in modeling morphology and the role of memory-based models in current theory formation. In phonology, memory-based models have been proposed and matched to psycholinguistic empirical data for such tasks as final devoicing in Dutch (Ernestus, 2006), Italian conjugation (Eddington, 2002b), stress assignment in Dutch (Daelemans *et al.*, 1994), Spanish (Eddington, 2004), and in English compounds (Plag *et al.*, 2007), etc.

Much less work has attempted to develop memory-based models of syntactic processing. Data-Oriented Parsing (DOP) (Scha *et al.*, 1999; Bod, 2006) is one influential algorithm where parsing is seen as similarity-based lookup and reconstruction of memorized fragments of previously analyzed sentences, kept in memory. It has led to experiments modeling priming effects in syntactic processing (Snider, 2007). See Hay & Bresnan (2006) for additional empirical work in exemplar-based syntax. In addition to work based on traditional parsing approaches rooted in phrase-based or dependency-based grammar theory, the memory-based shallow parsing research described in the previous Section also makes possible psycholinguistic studies (e.g. on attachment preferences).

As for our overview of memory-based approaches in computational linguistics, we have not tried to be exhaustive here, but rather to point to interesting studies and starting points in the literature illustrating the power of memory-based models as models of language acquisition and use.

5 Generalization and Abstraction

As discussed in Section 3, the memory-based learning approach is functionally similar to other supervised discriminative machine-learning methods capable of learning classification tasks. It is hard, if not fundamentally impossible to say in general that one discriminative machine-learning algorithm is better than the other (Wolpert, 2002). Yet, certain advantages of memory-based learning in learning NLP tasks have been noted in the literature. First, we expand in some detail the tenet that "forgetting exceptions is harmful in language learning" (Daelemans *et al.*, 1999b); then, we review a few algorithmic advantages of memory-based learning.

"Forgetting" training examples is a common trait of many machine learning algorithms; the identity of training examples is lost, while in exchange, each training example influences to a small extent the construction of an abstract model composed of probabilities or rules. In machine learning, learning is often equated with abstraction; in turn, abstraction is often equated with the capacity to generalize to new cases. A key realization is that memory-based learning is able to generalize, yet does not abstract from the data. In two studies, memory-based learning was contrasted against abstracting learners, namely decision-tree learners and rule learners (Daelemans *et al.*, 1999b; Daelemans & Van den Bosch, 2005), resulting in the consistent observation that the abstracting learners do not outperform the memory-based learners on any of a wide selection of NLP tasks. In a second series of experiments, Daelemans and Van den Bosch show that selected removal of training examples from the memory of a memory-based classifier, guided by criteria that supposedly express the utility of an individual example in classification, does not produce better generalization performance, although with some tasks, up to 40% of the examples can be removed from memory without damaging performance significantly (Daelemans & Van den Bosch, 2005). A safe conclusion from these studies is that when high accuracy is more important than optimal memory usage or speed, it is best to never forget training examples.

In practical terms, the k-nearest neighbour classifier has a number of advantages that make memory-based learning the method of choice in certain particular situations, compared to other rival discriminative supervised machine-learning algorithms:

(1) The basic version of the k-NN classifier that uses the overlap metric, is insensitive to the number of class labels, both in terms of efficiency in training and in classification. This makes memory-based learning suited for classification tasks with very large numbers of classes, such as word prediction or machine translation;

(2) Memory-based learning is able to reproduce the classification of training data flawlessly, as long as there are no identical training instances in memory with different class labels. This advantage, an important component of the "forgetting exceptions is harmful" tenet, is especially useful in NLP tasks in which much of the training data can be expected to recur in new data, such as in word pronunciation, where a typical lexicon used for training will already contain the pronunciation of approximately 95% of all words in a new text;

(3) Memory-based learning allows for incremental learning at no cost, or with little cost if the similarity function uses weighting functions; this is practical in situations in which training examples become available over time, and the classifier needs to be retrained preferably with the availability of each new training example, e.g. in active learning (Thompson *et al.*,

1999). Also, the algorithm is equally easily *decremental*, allowing for fast leave-one-out testing, a powerful evaluation scheme (Weiss & Kulikowski, 1991).

(4) As mentioned earlier, it has been shown that the 1-nearest neighbor classifier has an attractive error upper bound: as the amount of data approaches infinity, it is guaranteed to yield an error rate no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data) (Cover & Hart, 1967).

The main disadvantage of memory-based learning, compared to most rival approaches, is its slow classification speed. Its worst-case complexity of classification is $O(nf)$, where n is the number of memorized examples, and f is the number of features; each new example needs to be compared against all of the memorized examples, each time involving a comparison of all f features. Implementing k-nearest neighbor classification in a trie (Knuth, 1973) can under the proper conditions, viz. highly differing feature weights, or by dropping the guarantee of finding the exact nearest neighbors (Daelemans *et al.*, 1997b), reduce classification time to $O(f)$.

Another disadvantage of the memory-based learning approach that it shares with other discriminative classifiers is that its strength is in classification tasks with relatively low dimensionality in the class space. In the larger context of NLP tasks with structured output specifications, such as parsing or machine translation, it is widely recognized that discriminative classification alone is not enough to perform these global tasks, as the class spaces that would cover entire sequences, or large subsequences, would be too high-dimensional, thus too sparse to allow for sufficient amounts of examples per class. Even memory-based learning, with its insensitivity towards the number of classes, suffers directly from such sparseness. Currently, the generally adopted solution is to combine discriminitive classifiers with an inference method that searches for an optimal global solution.

6 Generalizing examples

To alleviate the computational inefficiency of the classifiation process in memory-based learning, part of the early work in k-NN classification focused on *editing* methods, i.e., methods for the removal of certain examples in memory that are estimated to be useless or even harmful to classification. Yet, bad estimates may lead to the removal of useful examples, thus to loss of generalization performance. While keeping full memory may be a safe guideline to avoid any eventual harmful effect of editing, in the interest of speed of classification, it is still interesting and tempting to explore other means to reduce the need for memory, provided that performance is not harmed. In this section we explore methods that attempt to abstract over memorized examples in a different and more careful manner, namely by merging examples into generalized examples, using various types of merging operations.

We start, in subsection 6.1, with an overview of existing methods for generalizing examples in memory-based learning. Subsequently we present Fambl, a memory-based learning algorithm variant that merges similar same-class nearest-neighbor examples into "families". In subsection 6.2 we compare Fambl to pure memory-based learning on a range of NLP tasks.

6.1 Careful abstraction in memory-based learning

Paths in decision trees can be seen as generalized examples. In IGTREE (Daelemans *et al.*, 1997b) and C4.5 (Quinlan, 1993) this generalization is performed up to the point where no actual example is left in memory; all is converted to nodes and arcs. Counter to this decision-tree compression, approaches exist that start with storing individual examples in memory, and carefully merge some of these examples to become a single, more general example, only when there is some evidence that this operation is not harmful to generalization performance. Although overall memory is compressed, the memory still contains individual items on which the same k-nearest neighbor classification can be performed. The abstraction occurring in this approach is that after a merge, the merged examples incorporated in the new generalized example are deleted individually, and cannot be reconstructed. Example approaches to merging examples are NGE (Salzberg, 1991) and its batch variant BNGE (Wettschereck & Dietterich, 1995), and RISE (Domingos, 1996). We provide brief discussions of two of these algorithms: NGE and RISE.

NGE (Salzberg, 1991), an acronym for *Nested Generalized Exemplars*, is an incremental learning theory for merging instances (or exemplars, as Salzberg prefers to refer to examples stored in memory) into *hyperrectangles*, a geometrically motivated term for merged exemplars. NGE adds examples to memory in an incremental fashion (at the onset of learning, the memory is seeded with a small number of randomly picked examples). Every time a new example is presented, it is matched with all exemplars in memory, which can be individual or merged exemplars (hyperrectangles). When it is classified correctly by its nearest neighbor (an individual exemplar or the smallest matching hyperrectangle), the new example is merged with it, yielding a new, more general hyperrectangle.

Figure 2 illustrates two mergings of examples of a morphological task (German plural) with exemplars. On the top of figure 2, the example *-urSrIftF* (from the female-gender word *Urschrift*), labeled with class *en* (representing the plural form *Urschriften*), is merged with the example *t@rSrIftF* (from the female-gender word *Unterschrift*), also of class *en*, to form the generalized exemplar displayed on the right-hand side. On the first two features, a dis-

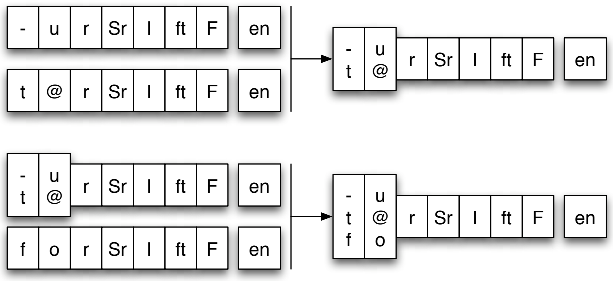

Figure 2. Two examples of the generation of a new hyperrectangle in NGE: from a new example and an individual exemplar (top) and from a new example and the hyperrectangle from the top example (bottom).

junction is formed of, respectively, the values - and *t*, and *u* and @. This means that the generalized example matches on any other example that has value - or value *t* on the first feature, and any other example that has value *u or* value @ on the second feature. The lower part of Figure 2 displays a subsequent merge of the newly generalized example with another same-class example, *forSrIftF* (the female-gender word *Forschrift*), which leads to a further generalization of the first two features.

In nested generalized examples, abstraction occurs because it is not possible to retrieve the individual examples nested in the generalized example; new generalization occurs because the generalized example not only matches fully with its nested examples, but would also match perfectly with potential examples with feature-value combinations that were not present in the nested examples; the generalized example in Figure 2 would also match *torSrIft*, *f@rSrIft*, *furSrIft*, *-orSrIft*. These examples do not necessarily match existing German words, but they might – and arguably they would be labeled with the correct plural inflection class.

RISE (*Rule Induction from a Set of Exemplars*) (Domingos, 1995, 1996) is a multi-strategy learning method that combines memory-based learning with rule-induction (Michalski, 1983; Clark & Niblett, 1989; Clark & Boswell, 1991). As in NGE, the basic method is that of a memory-based learner and classifier, only operating on a more general type of example. RISE learns a memory filled with *rules* which are all derived from individual examples. Some rules are example-specific, and other rules are generalized over sets of examples.

RISE inherits parts of the rule induction method of CN2 (Clark & Niblett, 1989; Clark & Boswell, 1991). CN2 is an incremental rule-induction algorithm that attempts to find the "best" rule governing a certain amount of examples in the example base that are not yet covered by a rule. "Goodness" of a rule is estimated by computing its apparent accuracy, i.e. class prediction strength (Cost & Salzberg, 1993) with Laplace correction (Niblett, 1987; Clark & Boswell, 1991).

RISE induces rules in a careful manner, operating in cycles. At the onset of learning, all examples are converted to example-specific rules. During a cycle, for each rule a search is made for the nearest example not already covered by it that has the same class. If such an example is found, rule and example are merged into a more general rule. Instead of disjunctions of values, RISE generalizes by inserting wild card symbols (that match with any other value) on positions with differing values. At each cycle, the goodness of the rule set on the original training material (the individual examples) is monitored. RISE halts when this accuracy measure does not improve (which may already be the case in the first cycle, yielding a plain memory-based learning algorithm).

Figure 3 illustrates the merging of individual examples into a rule. The rule contains seven normally valued conditions, and two wild cards, '*'. The rule now matches on every female-gender example ending in *SrIft* (*Schrift*).

When processing new examples, RISE classifies them by searching for the best-matching rule.

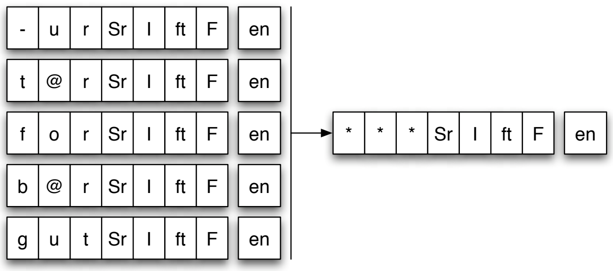

Figure 3. An example of an induced rule in RISE, displayed on the right, with the set of examples that it covers (and from which it was generated) on the left.

Fambl: merging example families

Fambl, for *FAMily-Based Learning*, is a variant of MBL that constitutes an alternative approach to careful abstraction over examples. The core idea of Fambl, in the spirit of NGE and RISE, is to transform an example base into a set of *example family expressions*. An example family expression is a hyper-rectangle, but the procedure for merging examples differs from that in NGE or in RISE. First, we outline the ideas and assumptions underlying Fambl. We then give a procedural description of the learning algorithm.

Classification of an example in memory-based learning involves a search for the nearest neighbors of that example. The value of k in k-NN determines how many of these neighbors are used for extrapolating their (majority) classification to the new example. A fixed k ignores (smoothes) the fact that an example is often surrounded in example space by a number of examples of the same class that is actually larger or smaller than k. We refer to such a variable-sized set of same-class nearest neighbors as an example's *family*. The

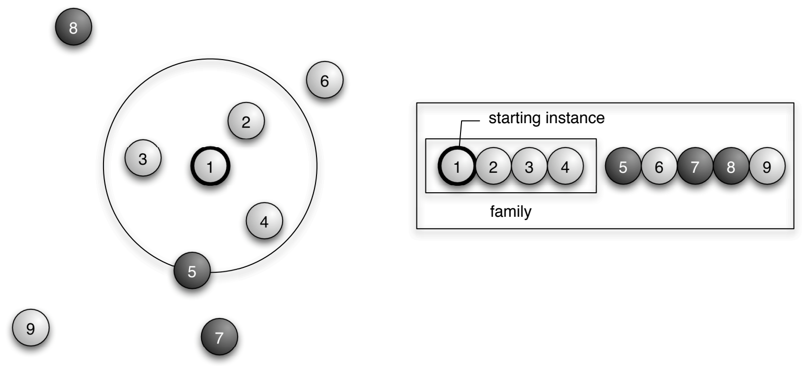

Figure 4. An example of a family in a two-dimensional example space (left). The family, at the inside of the circle, spans the focus example (marked with number 1) and the three nearest neighbors labeled with the same class (indicated by their color). When ranked in the order of distance (right), the family boundary is put immediately before the first example of a different class, the gray example with number 5.

extreme cases are on the one hand examples that have a nearest neighbor of a different class, i.e., they have no family members and are a family on their own, and on the other hand examples that have as nearest neighbors all other examples of the same class.

Thus, families represent same-class clusters in example space, and the number and sizes of families in a data set reflect the *disjunctivity* of the data set: the degree of scatteredness of classes into clusters. In real-world data sets, the situation is generally somewhere between the extremes of total disjunctivity (one example per cluster) and no disjunctivity (one cluster per class). Many types of language data appear to be quite disjunct (Daelemans *et al.*, 1999b). In highly disjunct data, classes are scattered among many small clusters, which means that examples have few nearest neighbors of the same class on average.

Figure 4 illustrates how Fambl determines the family of an example in a simple two-dimensional example space. All nearest neighbors of a randomly picked starting example (marked by the black dot) are searched and ranked in the order of their distance to the starting example. Although there are five examples of the same class in the example space, the family of the starting example contains only three examples, since its fourth-nearest example is of a different class.

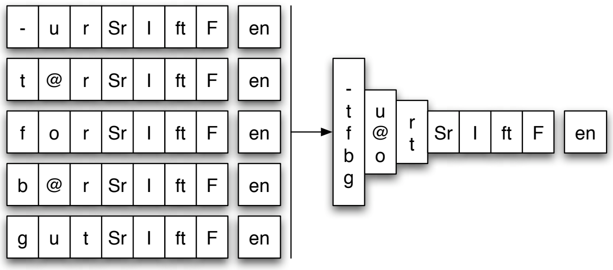

Figure 5. An example of family creation in Fambl. Five German plural examples (left) are merged into a family expression (right).

Families are converted in Fambl to *family expressions*, which are hyper-rectangles, by merging all examples belonging to that family simultaneously. Figure 5 illustrates the creation of a family expression from an example family. In contrast with NGE,

- family expressions are created in one non-incremental operation on the entire example base, rather than by step-wise nesting of each individual family member;
- a family is abstracted only once and is not merged later on with other examples or family expressions;
- families cannot contain “holes”, i.e., examples with different classes, since the definition of family is such that family abstraction halts as soon as the nearest neighbor with a different class is met in the local neighborhood.

The general mode of operation of Fambl is that it randomly picks examples from an example base one by one from the set of examples that are not already part of a family. For each newly picked example, Fambl determines its family, generates a family expression from this set of examples, and then marks all involved examples as belonging to a family (so that they will not be picked as a starting point or member of another family). Fambl continues determining families until all examples are marked as belonging to a family.

Families essentially reflect the locally optimal k surrounding the example around which the family is created. The locally optimal k is a notion that is also used in locally weighted learning methods (Vapnik & Bottou, 1993; Wettschereck & Dietterich, 1994; Wettschereck, 1994; Atkeson *et al.*, 1997); however, these methods do not abstract from the learning material. In this sense, Fambl can be seen as a local abstractor.

```
Procedure FAMBL FAMILY-EXTRACTION:
Input: A training set $TS$ of examples $I_{1...n}$, each example being labeled with a
   family-membership flag set to $FALSE$
Output: A family set $FS$ of family expressions $F_{1...m}$, $m \leq n$
$i = f = 0$
   (1) Randomize the ordering of examples in $TS$
   (2) While not all family-membership flags are $TRUE$, Do
       • While the family-membership flag of $I_i$ is $TRUE$ Do increase $i$
       • Compute $NS$, a ranked set of nearest neighbors to $I_i$ with the same
         class as $I_i$, among all examples with family-membership flag $FALSE$.
         Nearest-neighbor examples of a different class with family-membership
         flag $TRUE$ are still used for marking the boundaries of the family.
       • Set the membership flags of $I_i$ and all remaining examples in $NS$ to
         $TRUE$
       • Merge $I_i$ and all examples in $NS$ into the family expression $F_f$ and store
         this expression along with a count of the number of example merged in
         it
       • $f = f + 1$
```

Figure 6. Pseudo-code of the family extraction procedure in FAMBL.

The Fambl algorithm converts any training set of labeled examples to a set of family expressions, following the procedure given in Figure 6. After learning, the original example base is discarded, and further classification is based only on the set of family expressions yielded by the family-extraction phase. Classification in Fambl works analogously to classification in pure memory-based learning (with the same similarity and weighting metrics as we used so far with MBL): a match is made between a new test example and all stored family expressions. When a family expression contains a disjunction of values for a certain feature, a match is counted when one of the disjunctive values matches the value at that feature in the new example. How the match is counted exactly depends on the similarity metric. With the overlap metric, the feature weight of the matching feature is counted, while with the MVDM metric the smallest MVDM distance among the disjuncted feature values is also incorporated in the count.

6.2 Experiments with Fambl

We performed experiments with Fambl on four language processing tasks. We first introduce these four tasks, ranging from morpho-phonological tasks to semanto-syntactic tasks, varying in scope (word level and sentence level) and basic type of example encoding (non-windowing and windowing). We briefly describe the four tasks here and provide some basic data set specifications in Table 6.2. At the same time, we also provide results for standard MBLP for comparison.

(1) GPLURAL, the formation of the plural form of German nouns. The task is to classify a noun as mapping to one out of eight classes, representing the noun's plural formation. We collected 25,753 German nouns from the

German part of the CELEX-2 lexical database[2]. We removed from this data set cases without plurality marking, cases with Latin plural in `-a`, and a miscellaneous class of foreign plurals. From the remaining 25,168 cases, we extracted or computed for each word the plural suffix, the gender feature, and the syllable structure of the last two syllables of the word in terms of onsets, nuclei, and codas expressed using a phonetic segmental alphabet. We use a 50%–50% split in 12,584 training examples and 12,584 test instances. Generalization performance is measured in accuracy, viz. the percentage of correctly classified test instances.

(2) DIMIN, Dutch diminutive formation, uses a similar scheme to the one used in the GPLURAL task to represent a word as a single example. The task and data were introduced by Daelemans *et al.* (1997a). A noun, or more specifically its phonemic transcription, is represented by its last three syllables, which are each represented by four features: (1) whether the syllable is stressed (binary), (2) the onset, (3) the nucleus, and (4) the coda. The class label represents the identity of the diminutive inflection, which is one out of five (*-je*, *-tje*, *-etje*, *-pje*, or *-kje*). For example, the diminutive form of the Dutch noun *beker* (cup) is *bekertje* (little cup). Its phonemic representation is *['bek@r]*. The resulting example is _ _ _ _ + *b e* _ − *k @ r tje*. The data are extracted from the CELEX-2 lexical database (Baayen *et al.*, 1993). The training set contains 2,999 labeled examples of nouns; the test set contains 950 instances. Again, generalization performance is measured in accuracy, viz. the percentage of correctly classified test instances.

(3) PP, prepositional-phrase attachment, is the classical benchmark data set introduced by Ratnaparkhi *et al.* (1994). The data set is derived from the Wall Street Journal Penn Treebank (Marcus *et al.*, 1993). All sentences

[2] Available from the Linguistic Data Consortium (`http://www.ldc.upenn.edu/`).

containing the pattern "VP NP PP" with a single NP in the PP were converted to four-feature examples, where each feature contains the head word of one of the four constituents, yielding a "V N1 P N2" pattern such as "*each pizza with Eleni*", or "*eat pizza with pineapple*". Each example is labeled by a class denoting whether the PP is attached to the verb or to the N1 noun in the treebank parse. We use the original training set of 20,800 examples, and the test set of 3,097 instances. Noun attachment occurs slightly more frequently than verb attachment; 52% of the training examples and 59% of the test examples are noun attachment cases. Generalization performance is measured in terms of accuracy (the percentage of correctly classified test instances).

(4) CHUNK is the task of splitting sentences into non-overlapping syntactic phrases or constituents, e.g., to analyze the sentence "*He reckons the current account deficit will narrow to only $ 1.8 billion in September.*" as

$[He]_{NP}$ $[reckons]_{VP}$ $[the\ current\ account\ deficit]_{NP}$ $[will\ narrow]_{VP}$
$[to]_{PP}$ $[only\ \$\ 1.8\ billion]_{NP}$ $[in]_{PP}$ $[September]_{NP}$.

The data set, extracted from the WSJ Penn Treebank through a flattened, intermediary representation of the trees (Tjong Kim Sang & Buchholz, 2000), contains 211,727 training examples and 47,377 test instances. The examples represent seven-word windows of words and their respective part-of-speech tags computed by the Brill tagger (Brill, 1992) (which is trained on a disjoint part of the WSJ Penn Treebank), and each example is labeled with a class using the IOB type of segmentation coding as introduced by (Ramshaw & Marcus, 1995). Generalization performance is measured by the F-score on correctly identified and labeled constituents in test data, using the evaluation method originally used in the "shared task" sub-event of the CoNLL-2000 conference (Tjong Kim Sang & Buchholz, 2000) in which this particular training and test set were used.

As a first experiment, we varied both the normal k parameter (which sets the number of equidistant neighbors in the nearest neighbor set used in k-NN classification), and the Fambl-specific parameter that sets the maximum k distances in the family extraction stage, which we will refer to as K (the `-K` parameter in the command-line version of Fambl). The two parameters are obviously related - the K can be seen as a preprocessing step that "pre-compiles" the k for the k-NN classifier. The k-nearest neighbor classifier that operates on the set of family expressions can be set to 1, hypothetically, since the complete example space is pre-partitioned in many small regions of various sizes (with maximally K different distances) that each represent a locally appropriate k.

If the empirical results would indeed show that k can be set to 1 safely when K is set at an appropriately large value, then FAMBL could be seen as a means to factor the important k parameter away from MBL. We performed comparative experiments with normal MBL and Fambl on the four benchmark tasks, in which we varied both the k parameter in MBL, and the K parameter in Fambl while keeping $k = 1$. Both k and K were varied in the pseudo-exponential series $[0, 1, \ldots, 9, 10, 15, \ldots, 45, 50, 60, \ldots, 90, 100]$. The results of the experiments are visualized in Figure 7.

A very large value of K means that Fambl incorporates virtually any same-class nearest neighbor at any furthest distance in creating a family, as long as there are no different-class nearest neighbors in between. It would be preferable to be able to fix K at a very high value without generalization performance loss, since this would effectively factor out not only the k

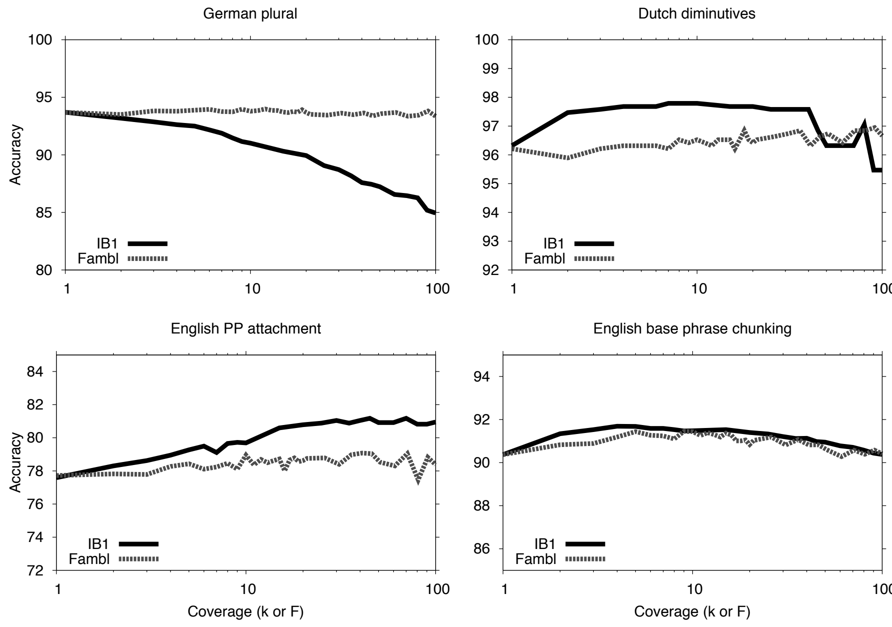

Figure 7. Generalization accuracies (in terms of % of correctly classified test instances) and F-scores, where appropriate, of MBL with increasing k parameter, and Fambl with $k = 1$ and increasing K parameter.

parameter, but also the K parameter. This situation is represented in the graph displaying the results of GPLURAL (top left corner of Figure 7). While a larger k in IB1 leads to a steady decline in generalization accuracy on test data of the GPLURAL task, Fambl's accuracy remains very much at the same level regardless of the value of K. The results with the other three tasks also show a remarkably steady generalization accuracy (or F-score, with CHUNK) of Fambl, with increasing K, but in all three cases Fambl's score is not higher than IB1's. Especially with the DIMIN and PP tasks, matching on families rather than on examples leads to less accurate classifications at wide ranges of K.

While it retains a similar performance to MBL, Fambl also attains a certain level of compression. This can be measured in at least two ways. First, in

Figure 8 the amount of compression (in terms of percentages) is displayed of the number of families versus the original number of examples, with increasing values of K, for four of our tasks. As Figure 8 shows, the compression rates converge for all four tasks at similar and very high levels; between 77% for GPLURAL to 92% for DIMIN. Apparently, setting K at a large enough value ensures that at that point even the largest families are identified; typically there will be 100 or less different distances in any found family.

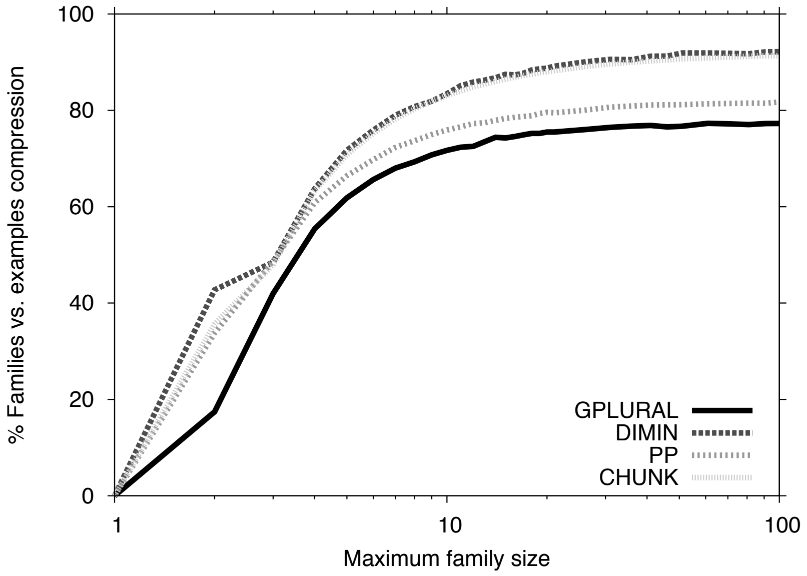

Figure 8. Compression rates (percentages) of families as opposed to the original number of examples, produced by Fambl at different maximal family sizes (represented by the x-axis, displayed at a log scale).

Some more detailed statistics on family extraction are listed in Table 2, measured for four tasks at the $K = 100$ mark. The actual number of families varies widely among the tasks, but this correlates with the number of training examples (cf. Table 6.2). The average number of members lies at about the same order of magnitude for the four tasks – between six and thirteen.

The table also shows the raw memory compression when compared with a straightforward storage of the flat example base. In the straightforward implementation of Fambl, storing a family with one example uses more memory than storing one example because of the bookkeeping information associated with storing possible disjunctions at each feature. The net gains of the high compression rates displayed in Figure 8 are still positive: from 23% to 73% compression. This is, however, dependent on the particular implementation.

Task	Number of families	Av. number of members	Memory compression (%)
GPLURAL	1,749	7.2	62.0
DIMIN	233	12.9	73.4
PP	3,613	5.8	23.4
CHUNK	17,984	11.8	51.9

Table 2. Number of extracted families at a maximum family size of 100, the average number of family members, and the raw memory compression, for four tasks.

Two example families, one for the PP and the other for the CHUNK task, are displayed in Table 3. The first example family, labeled with the Verb attachment class, represents the *attributed* ... *to* ... pattern, but also includes the example *bring focus to opportunities*, which is apparently the closest neighbor to the other four examples having the same class. The second family represents cases of the beginning of a noun phrase starting with *most of*. The context left of *most of* deviates totally between the four examples making up the family, while the right context represents a noun phrase beginning with *the* or *his*. This family would also perfectly match sentence fragments inside the family hyperrectangle, such as *because computers do most of the top*, or *he still makes most of the 50*, and many more recombinations. Analogously, the

PP family example displayed in Table 3 would also perfectly match *attributed decline to increases*, *bring focus to demand*, etcetera.

Task	Example family	Class
PP	*attributed gains to demand*	Verb attachment
	attributed improvement to demand	
	attributed performance to increases	
	attributed decline to demand	
	bring focus to opportunities	
NP	*because computers do* **most** *of the work*	B-NP
	demand rights to **most** *of the 50*	
	he still makes **most** *of his furs*	
	screens , said **most** *of the top*	

Table 3. Two example families (represented by their members) extracted from the PP and CHUNK data sets, respectively. The part-of-speech tags in the CHUNK example family are left out for legibility. The bold words in the CHUNK example are the focus words in the windows.

Overall, the comparison between Fambl and MBL shows that Fambl does not profit from the relatively large generalizing capacity of family expressions, that in principle would allow some unseen examples attain a higher score in the similarity function. Apart from the question whether this relative re-ranking of examples would have any effect on classification, it is obvious that many examples covered by family expressions are unlikely to occur — consider, for example, *because computers do most of his furs*.

We conclude that Fambl has two main merits. First, Fambl can compress an example base down to a smaller set of family expressions (or, a generalizing hyperrectangle), attaining various compression rates in the same ballpark as attained by editing methods, but with a very steady generalization accuracy that is very close to IB1's. Second, Fambl almost factors out the k parameter. Fairly constant performance was observed while keeping $k = 1$ and varying

K, the maximal number of family members, across a wide range of values. To sum up, Fambl is a successful local k pre-compiler.

In this section, we discussed the fundamental eager-lazy dimension in Machine Learning from the point of view of lazy learning approaches such as MBL. We argued that it makes sense to keep all training data available (including "exceptional" cases) in learning language tasks because they may be good models to extrapolate from. At the same time, while being the cheapest possible learning approach, it is also an inherently expensive strategy during classification. There are several ways in which this problem can be alleviated: by using fast approximations of MBL such as IGTree (Daelemans *et al.*, 1997b; Daelemans & Van den Bosch, 2005), special optimized algorithms (Liu *et al.*, 2003) or even use of special hardware (Yeh *et al.*, 2007). In this Section, we showed that a alternative way to approach this problem is to develop algorithms for weak, bottom up generalization from the original instance space, making possible an efficiency increase while keeping generalization accuracy at the same levels as with normal MBL.

7 Further Reading

General introductions to memory-based learning (lazy learning, k-nearest neighbor classification, instance-based learning) and its relation to other strands of machine learning can be found in (Mitchell, 1997). Key historic publications on k-nearest neighbor classification are (Fix & Hodges, 1951; Cover & Hart, 1967; Dudani, 1976; Dasarathy, 1991). The field of machine learning adopted and adapted the k-nearest neighbor algorithm under different names, such as memory-based reasoning (Stanfill & Waltz, 1986), instance-based learning (Aha *et al.*, 1991), and locally weighted learning (Atkeson *et al.*, 1997). An important development in these latter publications has been the introduction of similarity functions for non-numeric features (Aha *et al.*, 1991; Cost & Salzberg, 1993), which enabled the application to symbolic language tasks. (Stanfill, 1987; Weijters, 1991) both showed that the neural-network approach to grapheme-phoneme conversion of (Sejnowski & Rosenberg, 1987) could be emulated and improved by using a k-nearest neighbor classifier. From the beginning of the 1990s onwards, memory-based learning has been applied to virtually all areas of natural language processing. (Daelemans & Van den Bosch, 2005) is a book-length treatise on memory-based language processing.

Sections 3 and 4 already pointed to studies using MBL and alternative memory-based approaches in various areas of computational linguistics and computational psycholinguistics. More references can be found in the regularly updated reference guide to the TiMBL software (Daelemans *et al.*, 2007).

Relations to statistical language processing, in particular the interesting equivalence relations with back-off smoothing in probabilistic classifiers, are discussed in (Zavrel & Daelemans, 1997). Relations between classification-based word prediction and statistical language modeling are identified in (Van den Bosch, 2005, 2006b).

In machine translation, k-nearest neighbor classification bears a close relation with example-based machine translation (EBMT). A first EBMT-implementation using TiMBL is described in (Van den Bosch *et al.*, 2007).

The first dissertation-length study devoted to the approach is (Van den Bosch, 1997), in which the approach is compared to alternative learning methods for NLP tasks related to English word pronunciation (stress assignment, syllabification, morphological analysis, alignment, grapheme-to-phoneme conversion). TiMBL is also central in the Ph.D. theses of Buchholz (2002), Lendvai (2004), Hendrickx (2005), and Hoste (2005). In 1999 a special issue of the *Journal for Experimental and Theoretical Artificial Intelligence* (Vol. 11(3)), was devoted to Memory-Based Language Processing. In this special issue, the approach was related also to exemplar-based work in the Data Oriented Parsing (DOP) framework (Scha *et al.*, 1999) and analogy-based reasoning in NLP research (Pirrelli & Yvon, 1999).

References

A., Krott, Schreuder R., Baayen R.H., & Dressler W.U. (2007), Analogical effects on linking elements in German compound words, *Language and Cognitive Processes* 22(1):25–57.

Aha, D. W., D. Kibler, & M. Albert (1991), Instance-based learning algorithms, *Machine Learning* 6:37–66.

Ahn, D. (2006), The stages of event extraction, in *Proceedings of the COLING-ACL 2006 Workshop on Annotating and Reasoning about Time and Events*, Sydney, Australia, (1–8).

Androutsopoulos, I., G. Paliouras, V. Karkaletsis, G. Sakkis, C. D. Spyropoulos, & P. Stamatopoulos (2000), Learning to filter spam e-mail: A comparison of a Naive Bayesian and a memory-based approach, in *Proceedings of the "Machine Learning and Textual Information Access" Workshop of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases*.

Atkeson, C., A. Moore, & S. Schaal (1997), Locally weighted learning, *Artificial Intelligence Review* 11(1–5):11–73.

Baayen, R. H., R. Piepenbrock, & H. van Rijn (1993), *The CELEX lexical data base on CD-ROM*, Linguistic Data Consortium, Philadelphia, PA.

Baldwin, T. & F. Bond (2003), A plethora of methods for learning English countability, in M. Collins & M. Steedman (eds.), *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, ACL, New Brunswick, NJ, (73–80).

Bod, R. (2006), Exemplar-based syntax: How to get productivity from examples, *The Linguistic Review* 23(3):291–320.

Brill, Eric (1992), A simple rule-based part of speech tagger, in *Proceedings of the DARPA Workshop on Speech and Natural Language*.

Buchholz, S. (1998), Distinguishing complements from adjuncts using memory-based learning, in *Proceedings of the ESSLLI-98 Workshop on Automated Acquisition* *of Syntax and Parsing, Saarbrücken, Germany.*

Buchholz, S. (2002), *Memory-Based Grammatical Relation Finding*, PhD thesis, University of Tilburg.

Buchholz, S. & A. Van den Bosch (2000), Integrating seed names and n-grams for a named entity list and classifier, in *Proceedings of the Second International Conference on Language Resources and Evaluation*, Athens, Greece, (1215–1221).

Buchholz, S., J. Veenstra, & W. Daelemans (1999), Cascaded grammatical relation assignment, in *EMNLP-VLC'99, the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Canisius, S., T. Bogers, A. Van den Bosch, J. Geertzen, & E. Tjong Kim Sang (2006), Dependency parsing by inference over high-recall dependency predictions, in *Proceedings of the Tenth Conference on Computational Natural Language Learning, CoNLL-X*, New York, NY.

Clark, P. & R. Boswell (1991), Rule induction with CN2: Some recent improvements, in *Proceedings of the Sixth European Working Session on Learning*, Berlin: Springer Verlag, (151–163).

Clark, P. & T. Niblett (1989), The CN2 rule induction algorithm, *Machine Learning* 3:261–284.

Cost, S. & S. Salzberg (1993), A weighted nearest neighbour algorithm for learning with symbolic features, *Machine Learning* 10:57–78.

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

Cumbreras, M.A. García, L.A. Ureña López, & F. Martínez Santiago (2006), Bruja: Question classification for spanish, in *Proceedings of the EACL 2006 Workshop on Multilingual Question Answering*, Trento, Italy, (39–44).

Daelemans, W. (2002), A comparison of analogical modeling to memory-based language processing, in R. Skousen, D. Lonsdale, & D.B. Parkinson (eds.), *Analogical Modeling*, John Benjamins, Amsterdam, The Netherlands.

Daelemans, W., P. Berck, & S. Gillis (1997a), Data mining as a method for linguistic analysis: Dutch diminutives, *Folia Linguistica* XXXI(1–2):57–75.

Daelemans, W., S. Buchholz, & J. Veenstra (1999a), Memory-based shallow parsing, in *Proceedings of CoNLL*, Bergen, Norway.

Daelemans, W., S. Gillis, & G. Durieux (1994), The acquisition of stress: a data-oriented approach, *Computational Linguistics* 20(3):421–451.

Daelemans, W. & A. Van den Bosch (1992), Generalisation performance of backpropagation learning on a syllabification task, in M. F. J. Drossaers & A. Nijholt (eds.), *Proceedings of TWLT3: Connectionism and Natural Language Processing*, Twente University, Enschede, (27–37).

Daelemans, W. & A. Van den Bosch (1996), Language-independent data-oriented grapheme-to-phoneme conversion, in J. P. H. Van Santen, R. W. Sproat, J. P. Olive, & J. Hirschberg (eds.), *Progress in Speech Processing*, Springer-Verlag, Berlin, (77–89).

Daelemans, W. & A. Van den Bosch (2005), *Memory-based language processing*, Cambridge University Press, Cambridge, UK.

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

Daelemans, W., A. Van den Bosch, & J. Zavrel (1999b), Forgetting exceptions is harmful in language learning, *Machine Learning, Special issue on Natural Language Learning* 34:11–41.

Daelemans, W., J. Zavrel, P. Berck, & S. Gillis (1996), MBT: A memory-based part of speech tagger generator, in E. Ejerhed & I. Dagan (eds.), *Proceedings of the Fourth Workshop on Very Large Corpora*, ACL SIGDAT, (14–27).

Daelemans, W., J. Zavrel, K. Van der Sloot, & A. Van den Bosch (1998), TiMBL: Tilburg Memory Based Learner, version 1.0, reference manual, Technical Report ILK 98-03, ILK Research Group, Tilburg University.

Daelemans, W., J. Zavrel, K. Van der Sloot, & A. Van den Bosch (2007), TiMBL: Tilburg Memory Based Learner, version 6.1, reference guide, Technical Report ILK 07-07, ILK Research Group, Tilburg University.

Damper, R. & J. Eastmond (1997), Pronunciation by analogy: impact of implementational choices on performance, *Language and Speech* 40:1–23.

Dasarathy, B. V. (1991), *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, IEEE Computer Society Press, Los Alamitos, CA.

De Meulder, F. & W. Daelemans (2003), Memory-based named entity recognition using unannotated data, in W. Daelemans & M. Osborne (eds.), *Proceedings of CoNLL-2003*, Edmonton, Canada, (208–211).

De Pauw, G., T. Laureys, W. Daelemans, & H. Van hamme (2004), A comparison of two different approaches to morphological analysis of Dutch, in *Proceedings of the ACL 2004 Workshop on Current Themes in Computational Phonology and Morphology*, (62–69).

De Pauw, G., P. Waiganjo, & G.-M. De Schryver (2007), Automatic diacritic restoration for resource-scarce languages, in *Proceedings of Text, Speech and Dialogue, Tenth International Conference*, Springer Verlag, Berlin, Germany, volume 4629 of *Lecture Notes in Computer Science*, (170–179).

Decadt, B., V. Hoste, W. Daelemans, & A. Van den Bosch (2004), GAMBL, genetic algorithm optimization of memory-based WSD, in R. Mihalcea & P. Edmonds (eds.), *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*, ACL, New Brunswick, NJ, (108–112).

Domingos, P. (1995), The RISE 2.0 system: A case study in multistrategy learning, Technical Report 95-2, University of California at Irvine, Department of Information and Computer Science, Irvine, CA.

Domingos, P. (1996), Unifying instance-based and rule-based induction, *Machine Learning* 24:141–168.

Dridan, R. & T. Baldwin (2007), What to classify and how: Experiments in question classification for japanese, in *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, Melbourne, Australia, (333–341).

Dudani, S.A. (1976), The distance-weighted k-nearest neighbor rule, in *IEEE Transactions on Systems, Man, and Cybernetics*, volume SMC-6, (325–327).

Eddington, D. (2002a), A comparison of two analogical models: Tilburg memory-based learner versus analogical modeling, in R. Skousen, D. Lonsdale, & D.B. Parkinson (eds.), *Analogical Modeling*, John Benjamins, Amsterdam, The Netherlands.

Eddington, D. (2002b), Dissociation in Italian Conjugations: A Single-Route Account, *Brain and Language* 81(1-3):291–302.

Eddington, D. (2002c), Spanish diminutive formation without rules or constraints, *Linguistics* 40(2):395–419.

Eddington, D. (2004), Issues in modeling language processing analogically, *Lingua* 114(7):849–871.

Ernestus, M. (2006), Statistically gradient generalizations for contrastive phonological features, *The Linguistic Review* 23(3):217–233.

Fernández, R., J. Ginzburg, & S. Lappin (2004), Classifying ellipsis in dialogue: A machine learning approach, in *Proceedings of the 20th International Conference on Computational Linguistics, COLING 2004*, Geneva, Switzerland, (240–246).

Fix, E. & J. L. Hodges (1951), Disciminatory analysis—nonparametric discrimination; consistency properties, Technical Report Project 21-49-004, Report No. 4, USAF School of Aviation Medicine.

Gustafson, J., N. Lindberg, & M. Lundeberg (1999), The august spoken dialogue system, in *Proceedings of Eurospeech'99*.

Hay, J. & J. Bresnan (2006), Spoken syntax: The phonetics of giving a hand in New Zealand English, *The Linguistic Review* 23(3):321–349.

Hay, J.B. & R.H. Baayen (2005), Shifting paradigms: gradient structure in morphology, *Trends in Cognitive Sciences* 9(7):342–348.

Hendrickx, I. (2005), *Local classification and global estimation: Explorations of the k-nearest neighbor algorithm*, Ph.D. thesis, Tilburg University.

Hendrickx, I. & A. Van den Bosch (2003), Memory-based one-step named-entity recognition: Effects of seed list features, classifier stacking, and unannotated data, in *Proceedings of CoNLL-2003*, (176–179).

Hoste, V. (2005), *Optimization in Machine Learning of Coreference Resolution*, Ph.D. thesis, University of Antwerp.

Hoste, V., I. Hendrickx, W. Daelemans, & A. Van den Bosch (2002), Parameter optimization for machine learning of word sense disambiguation, *Natural Language Engineering* 8(4):311–325.

Keuleers, E., D. Sandra, W. Daelemans, S. Gillis, G. Durieux, & E. Martens (2007), Dutch plural inflection: The exception that proves the analogy, *Cognitive Psychology* 54(4):283–318.

Keuleers, Emmanuel (2008), *Memory-Based Learning of Inflectional Morphology*, Ph.D. thesis, University of Antwerp.

Keuleers, Emmanuel & Walter Daelemans (2007), Memory-based learning models of inflectional morphology: A methodological case study, *Lingue e linguaggio* (2):151–174.

Kim, S.N. & T. Baldwin (2006), Interpreting semantic relations in noun compounds via verb semantics, in *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, Sydney, Australia, (491–498).

Knuth, D. E. (1973), *The art of computer programming*, volume 3: Sorting and searching, Addison-Wesley, Reading, MA.

Kokkinakis, D. (2000), PP-attachment disambiguation for swedish: Combining unsupervised and supervised training data, *Nordic Journal of Linguistics* 23(2):191–213.

Krott, A., R. H. Baayen, & R. Schreuder (2001), Analogy in morphology: modeling the choice of linking morphemes in Dutch, *Linguistics* 39(1):51–93.

Krott, Andrea, Robert Schreuder, & R. Harald Baayen (2002), Analogical hierarchy: exemplar-based modeling of linkers in dutch noun-noun compounds, in R. Skousen, D. Lonsdale, & D.B. Parkinson (eds.), *Analogical Modeling*, John Benjamins, Amsterdam, The Netherlands.

Lendvai, P. (2004), *Extracting information from spoken user input: A machine learning approach*, Ph.D. thesis, Tilburg University.

Lendvai, P. & J. Geertzen (2007), Token-based chunking of turn-internal dialogue act sequences, in *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*, Antwerp, Belgium, (174–181).

Lendvai, P., A. Van den Bosch, & E. Krahmer (2003a), Machine learning for shallow interpretation of user utterances in spoken dialogue systems, in *Proceedings of the EACL Workshop on Dialogue Systems: Interaction, adaptation and styles of management*, (69–78).

Lendvai, P., A. Van den Bosch, & E. Krahmer (2003b), Memory-based disfluency chunking, in *Proceedings of Disfluency in Spontaneous Speech Workshop (DISS'03)*, (63–66).

Lendvai, P., A. Van den Bosch, E. Krahmer, & M. Swerts (2002), Improving machine-learned detection of miscommunications in human-machine dialogues through informed data splitting, in *Proceedings of the ESSLLI Workshop on Machine Learning Approaches in Computational Linguistics*.

Lepage, Yves & Etienne Denoual (2005), Purest ever example-based machine translation: Detailed presentation and assessment, *Machine Translation* 19(3-4):251–282.

Leveling, J. & S. Hartrumpf (2007), On metonymy recogntion for geographic IR, in *Proceedings of GIR-2006, the 3rd Workshop on Geographical Information Retrieval*, Seattle, WA.

Liu, Ting, Andrew W. Moore, & Alexander Gray (2003), Efficient exact k-nn and nonparametric classification in high dimensions, in *In Proceedings of Neural Information Processing Systems*, MIT Press, (15).

Malouf, R. (2000), The order of prenominal adjectives in natural language generation, in *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, ACL, New Brunswick, NJ, (85–92).

Marcus, M., S. Santorini, & M. Marcinkiewicz (1993), Building a Large Annotated Corpus of English: the Penn Treebank, *Computational Linguistics* 19(2):313–330.

Marsi, E., A. Van den Bosch, & A. Soudi (2006), Memory-based morphological analysis generation and part-of-speech tagging of arabic, in *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, Ann Arbor, MI.

Michalski, R. S. (1983), A theory and methodology of inductive learning, *Artificial Intelligence* 11:111–161.

Mihalcea, R. (2002), Instance-based learning with automatic feature selection applied to word sense disambiguation, in *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, Taiwan.

Minnen, G., F. Bond, & A. Copestake (2000), Memory-based learning for article generation, in *Proceedings of the 4th Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*, ACL, New Brunswick, NJ, (43–48).

Mitchell, T. (1997), *Machine Learning*, McGraw-Hill, New York, NY.

Mitkov, R., R. Evans, & C. Orasan (2002), A new, fully automatic version of Mitkov's knowledge-poor pronoun resolution method, in *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, Springer-Verlag, (168–186).

Nastase, V., J. Sayyad-Shiarabad, M. Sokolova, & S. Szpakowicz (2006), Learning noun-modifier semantic relations with corpus-based and wordnet-based features, in *Proceedings of the Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference*, AAAI Press.

Niblett, T. (1987), Constructing decision trees in noisy domains, in *Proceedings of the Second European Working Session on Learning*, Sigma, Bled, Yugoslavia, (67–78).

Nivre, J., J. Hall, & J. Nilsson (2004), Memory-based dependency parsing, in H. T. Ng & E. Riloff (eds.), *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL 2004)*, Boston, Massachusetts, (49–57).

Orăsan, C. & R. Evans (2001), Learning to identify animate references, in W. Daelemans & R. Zajac (eds.), *Proceedings of the Fifth Workshop on Computational Language Learning, CoNLL-2001*, Toulouse, France, (129–136).

Parzen, E. (1962), On the estimation of a probability density function and the mode, *Annals of Mathematical Statistics* 33:1065–1076.

Pinker, S. (1999), *Words and Rules: The Ingredients of Language*, Basic Books.

Pirrelli, Vito & François Yvon (1999), The hidden dimension: a paradigmatic view of data-driven nlp, *J. Exp. Theor. Artif. Intell.* 11(3):391–408.

Plag, I., G. Kunter, & S. Lappe (2007), Testing hypotheses about compound stress assignment in English: a corpus-based investigation, *Corpus Linguistics and Lingustic Theory* 3(2):199–232.

Preiss, J. (2002), Anaphora resolution with memory-based learning, in *Proceedings of the Fifth Annual CLUK Research Colloquium*, (1–9).

Quinlan, J.R. (1993), C4.5*: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA.

Ramshaw, L.A. & M.P. Marcus (1995), Text chunking using transformation-based learning, in *Proceedings of the 3rd ACL/SIGDAT Workshop on Very Large Corpora, Cambridge, Massachusetts, USA*, (82–94).

Ratnaparkhi, A., J. Reynar, & S. Roukos (1994), A maximum entropy model for prepositional phrase attachment, in *Workshop on Human Language Technology*, ARPA, Plainsboro, NJ.

Rumelhart, D. E. & J. L. McClelland (1986), On learning the past tenses of English verbs, in J. L. McClelland & D. E. Rumelhart (eds.), *Parallel Distributed*

Processing: Explorations in the Microstructure of Cognition, The MIT Press, Cambridge, MA, volume 2: Psychological and Biological Models, (216–271).

Sagae, K. & A. Lavie (2005), A classifier-based parser with linear run-time complexity, in *Proceedings of the Ninth International Workshop on Parsing Technologies*, Vancouver, Canada, (125–132).

Salzberg, S. (1991), A nearest hyperrectangle learning method, *Machine Learning* 6:277–309.

Scha, Remko, Rens Bod, & Khalil Sima'an (1999), A memory-based model of syntactic analysis: data-oriented parsing, *Journal of Experimental and Theoretical Artificial Intelligence* 11:409–440.

Sejnowski, T.J. & C.S. Rosenberg (1987), Parallel networks that learn to pronounce English text, *Complex Systems* 1:145–168.

Skousen, R. (1989), *Analogical modeling of language*, Kluwer Academic Publishers, Dordrecht.

Skousen, R. (1992), *Analogy and Structure*, Kluwer Academic Publishers, Dordrecht.

Skousen, R. (2002), An overview of analogical modeling, in R. Skousen, D. Lonsdale, & D. B. Parkinson (eds.), *Analogical modeling: An exemplar-based approach to language*, John Benjamins, Amsterdam, The Netherlands, (11–26).

Snider, Neal (2007), Exemplars in syntax: Evidence from priming in corpora .

Spitters, M. (2000), Comparing feature sets for learning text categorization, in *Proceedings of the Sixth Conference on Content-Based Multimedia Access (RIAO 2002)*, Paris, France, (1124–1135).

Sporleder, C., M. Van Erp, T. Porcelijn, & A. Van den Bosch (2006), Identifying named entities in text databases from the natural history domain, in *Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC-2006*, Trento, Italy.

Stanfill, C. (1987), Memory-based reasoning applied to English pronunciation, in *Proceedings of the Sixth National Conference on Artificial Intelligence*, Morgan Kaufmann, Los Altos, CA, (577–581).

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

Stevenson, M. & Y. Wilks (1999), Combining weak knowledge sources for sense disambiguation, in *Proceedings of the International Joint Conference on Artificial Intelligence*.

Thompson, C. A., M. E. Califf, & R. J. Mooney (1999), Active learning for natural language parsing and information extraction, in *Proceedings of the Sixteenth International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, (406–414).

Tjong Kim Sang, E. & S. Buchholz (2000), Introduction to the CoNLL-2000 shared task: Chunking, in *Proceedings of CoNLL-2000 and LLL-2000*, (127–132).

Tjong Kim Sang, E. & J. Veenstra (1999), Representing text chunks, in *Proceedings of EACL'99*, Bergen, Norway, (173–179).

Van den Bosch, A. (1997), *Learning to pronounce written words: A study in inductive language learning*, Ph.D. thesis, Universiteit Maastricht.

Van den Bosch, A. (2005), Memory-based understanding of user utterances in a spoken dialogue system: Effects of feature selection and co-learning, in *Workshop Proceedings of the 6th International Conference on Case-Based Reasoning*, Chicago, IL, (85–94).

Van den Bosch, A. (2006a), Scalable classification-based word prediction and confusible correction, *Traitement Automatique des Langues* 46(2):39–63.

Van den Bosch, A. (2006b), Spelling space: A computational test bed for phonological and morphological changes in dutch spelling, *Written Language and Literacy* 9(1):25–44.

Van den Bosch, A. & W. Daelemans (1993), Data-oriented methods for grapheme-to-phoneme conversion, in *Proceedings of the 6th Conference of the EACL*, (45–53).

Van den Bosch, A. & W. Daelemans (1999), Memory-based morphological analysis, in *Proceedings of the 37th Annual Meeting of the ACL*, Morgan Kaufmann, San Francisco, CA, (285–292).

Van den Bosch, A., E. Krahmer, & M. Swerts (2001), Detecting problematic turns in human-machine interactions: Rule-induction versus memory-based learning approaches, in *Proceedings of the 39th Meeting of the Association for Computational Linguistics*, ACL, New Brunswick, NJ, (499–506).

Van den Bosch, A., N. Stroppa, & A. Way (2007), A memory-based classification approach to marker-based EBMT, in F. Van Eynde, V. Vandeghinste, & I. Schuurman (eds.), *Proceedings of the METIS-II Workshop on New Approaches to Machine Translation*, Leuven, Belgium, (63–72).

Van Halteren, H., J. Zavrel, & W. Daelemans (2001), Improving accuracy in word class tagging through combination of machine learning systems, *Computational Linguistics* 27(2):199–230.

Vapnik, V. & L. Bottou (1993), Local algorithms for pattern recognition and dependencies estimation, *Neural Computation* 5(6):893–909.

Veenstra, J. (1998), Fast NP chunking using memory-based learning techniques, in *Proceedings of BENELEARN'98*, Wageningen, The Netherlands, (71–78).

Veenstra, J., A. Van den Bosch, S. Buchholz, W. Daelemans, & J. Zavrel (2000), Memory-based word sense disambiguation, *Computers and the Humanities* 34(1/2):171–177.

Weijters, A. (1991), A simple look-up procedure superior to NETtalk?, in *Proceedings of the International Conference on Artificial Neural Networks - ICANN-91, Espoo, Finland*.

Weiss, S. & C. Kulikowski (1991), *Computer systems that learn*, San Mateo, CA: Morgan Kaufmann.

Wettschereck, D. (1994), *A study of distance-based machine learning algorithms*, Ph.D. thesis, Oregon State University.

Wettschereck, D. & T. G. Dietterich (1994), Locally adaptive nearest neighbor algorithms, in J. D. Cowan *et al.* (ed.), *Advances in Neural Information Processing Systems*, Morgan Kaufmann, Palo Alto, CA, volume 6, (184–191).

Wettschereck, D. & T. G. Dietterich (1995), An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms, *Machine Learning* 19:1–25.

Wolpert, D. H. (2002), The supervised learning no-free-lunch theorems, in *Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications*.

Wolters, M. & A. Van den Bosch (1997), Automatic phonetic transcription of words based on sparse data, in W. Daelemans, A. Van den Bosch, & A. Weijters (eds.), *Workshop notes of the ECML/MLnet Familiarization Workshop on Empirical Learning of Natural Language Processing Tasks*, University of Economics, Prague, Czech Republic, (61–70).

Yeh, A. (2000), Comparing two trainable grammatical relations finders, in *Proceedings of the 18th International Conference on Computational Linguistics*, Saarbrücken, Germany, (1146–1150).

Yeh, Y.J., H.Y. Li, W.J. Hwang, & C.Y. Fang (2007), FPGA Implementation of kNN Classifier Based on Wavelet Transform and Partial Distance Search, *LECTURE NOTES IN COMPUTER SCIENCE* 4522:512.

Yvon, F. & N. Stroppa (2007), Proportions in the Lexicon:(Re) Discovering Paradigms, *Lingue e linguaggio* (2):201–226.

Zavrel, J., P. Berck, & W. Lavrijssen (2000), Information extraction by text classification: Corpus mining for features, in *Proceedings of the workshop Information Extraction meets Corpus Linguistics*, Athens, Greece.

Zavrel, J. & W. Daelemans (1997), Memory-based learning: Using similarity for smoothing, in *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, (436–443).

Zavrel, J. & W. Daelemans (1999), Recent advances in memory-based part-of-speech tagging, in *VI Simposio Internacional de Comunicacion Social*, (590–597).

Zavrel, J. & W. Daelemans (2003), Feature-rich memory-based classification for shallow NLP and information extraction, in G. Nakhaeizadeh J. Franke & I. Renz (eds.), *Text Mining, Theoretical Aspects and Applications*, Springer Physica-Verlag, Heidelberg, Germany, (33–54).

Zavrel, J., W. Daelemans, & J. Veenstra (1997), Resolving PP attachment ambiguities with memory-based learning, in M. Ellison (ed.), *Proceedings of the Workshop on Computational Language Learning (CoNLL'97)*, ACL, Madrid.