

Expanding k -NN analogy with instance families

Draft version

Antal van den Bosch

ILK Research Group, Computational Linguistics
Tilburg University, The Netherlands
Antal.vdnBosch@kub.nl

Abstract

The classical k -NN algorithm assumes a fixed k , i.e. a fixed number of nearest neighbors in the classification of a new instance. This is in contrast with the intuition, of which analogical modeling is an implementation, that the neighborhood from which to extrapolate a class, can be variable in k and in numbers of neighbors. We present FAMBL2, a variant of k -NN that transforms instances to generalized instances. A generalized instance represents a “family” of same-class nearest neighbors, and can be seen as a precompiled, conservative variant of an analogical set. We show that classification in FAMBL2 can indeed benefit in terms of memory usage from the analogical effect of a variable k , but the net effect in generalization accuracy is generally not significant when compared to standard information-gain-ratio-weighted k -NN in IB1-GR.

1 Instance families: an implementation of the analogical set

A marked difference between the k -nearest-neighbor (k -NN) rule (Cover and Hart 1967) on the one hand, and analogical modeling (AM) (Skousen 1989) on the other hand, is their rigid (k -NN) versus dynamic and global (AM) bias in collecting evidence from memorized instances for the classification of a new instance. Especially with $k = 1$ (a common setting in k -NN) only the set of minimally- and equally-differing nearest neighbors is used for determining the class of a new instance. Fixing k ignores the fact that an instance is often surrounded in instance space by a number of instances 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 instance *family*.

Instance families, when generalized before the actual classification of new instances, can be seen as generalized instances that can be used in the same way as normal instances in regular k -NN classification. The key difference between k -NN classification based on instances versus families is that families can match new instances that contain value combinations *that have not been observed in in-*

dividual memorized instances. The idea of precompiling an instance base into instance families and then use these families for further k -NN classification has been implemented in the FAMBL algorithm (Van den Bosch 1999).

The precompilation step within FAMBL2 is the main difference with analogical modeling (Skousen 1989); the latter algorithm compiles analogical sets only during classification. Consequently, in AM as many different analogical sets can occur as there are test instances, and they are not stored in memory.

Precompiling generalized instances has a potential advantage of memory compression; less memory is needed when many instances can be summarised by a small number of families. Second, there is a potential classification speed advantage, since less memory items need to be traversed in k -NN classification. Nevertheless, our major interest lies in the effects that generalizing instances may have on generalization accuracy. The central question addressed in this contribution is whether FAMBL2 benefits from its strategy in that respect, when applied to natural language processing tasks. From the results obtained in a range of experiments, we conclude that generalizing instances has the prospected effects of memory compression and bringing, implicitly, more than a rigid number of instances in the nearest-neighbor set. However, the results show that the net effects in generalization accuracy are generally small and not significantly different when compared to standard k -NN classification with information-gain-ratio feature weighting as implemented in IB1-IG (Daelemans and Van den Bosch 1992, Daelemans, Van den Bosch and Weijters 1997b).

In this contribution, we start with a description of the FAMBL2 algorithm in Section 2. We then report on experiments on a range of natural language processing tasks with FAMBL2 and standard weighted k -NN, in Section 3. We summarize our findings and discuss the relation between FAMBL2 and analogical modeling in Section 4.

2 FAMBL2: Description of algorithm

Memory-based learning, also known as instance-based, example-based, lazy, case-based, exemplar-based, locally weighted, and analogical learning (Stanfill and Waltz 1986, Aha, Kibler and Albert 1991, Salzberg 1991, Kolodner 1993, Aha 1997, Atkeson, Moore and Schaal 1997), is a class of supervised inductive learning algorithms for learning classification tasks (Shavlik and Dietterich 1990). Memory-based learning treats a set of labeled (pre-classified) training instances as points in a multi-dimensional feature space, and stores them as such in an *instance base* in memory (rather than performing some abstraction over them).

An instance consists of a fixed-length vector of n feature-value pairs, and an information field containing the classification of that particular feature-value vector.

After the instance base is built, new (test) instances are classified by matching them to all instances in the instance base, and by calculating with each match the *distance*, given by a distance function $\Delta(X, Y)$ between the new instance X and the memory instance Y . The memory instances with the smallest distances are collected, and the classifications associated with these nearest neighbors are merged and extrapolated to assign a classification to the test instance.

The most basic distance function for patterns with symbolic features is the *overlap metric* $\Delta(X, Y) = \sum_{i=1}^n \delta(x_i, y_i)$, where $\Delta(X, Y)$ is the distance between patterns X and Y , represented by n features, and δ is the distance between feature values, $\delta(x_i, y_i) = 0$ if $x_i = y_i$, else 1. Classification in memory-based learning systems is basically performed by the k -nearest neighbor (k -NN) classifier (Cover and Hart 1967, Devijver and Kittler 1982), with k usually set to 1.

Early work on the k -NN classifier pointed at advantageous properties of the classifier in terms of generalization accuracies, under certain assumptions, because of its reliance on full memory (Fix and Hodges 1951, Cover and Hart 1967). However, the trade-off downside of full memory is computational inefficiency of the classification process, as compared to parametric classifiers that do abstract from the learning material. Therefore, several early investigations were performed into *editing* methods: finding criteria for the removal of instances from memory (Hart 1968, Gates 1972) without harming classification accuracy. Other studies on editing also explored the possibilities of detecting and removing noise from the learned data, so that classification accuracy might even improve (Wilson 1972, Devijver and Kittler 1980). The renewed interest in the k -NN classifier from the late 1980s onwards in the AI-subfield of machine learning (Stanfill and Waltz 1986, Stanfill 1987, Aha et al. 1991, Salzberg 1991) caused several new implementations of ideas on criteria for editing, but also other approaches to abstraction in memory-based learning emerged. We discern three types:

1. **Editing** (Hart 1968, Wilson 1972, Aha et al. 1991): removing instances according to a classification-related utility threshold they do not reach. Editing is not careful in principle, but the approaches that are discussed here and that are included in the empirical comparison (i.e. IB2 and IB3, (Aha et al. 1991)) collect statistical support for an editing operation to be harmless.
2. **Oblivious (partial) decision-tree abstraction** (Daelemans et al. 1997b): compressing (parts of) instances in the instance base into (parts of) decision-trees. Part of the motivation to perform top-down induction of decision trees (TDIDT) is the presence of clear differences in the relative importance of instance features, allowing features to be strictly ordered in matching (Quinlan 1986). The approach is dependent on the use of a feature-weighting metric.
3. **Carefully merging instances** (Salzberg 1991, Wettschereck and Dietterich 1995, Domingos 1996): merging multiple instances in single generalized

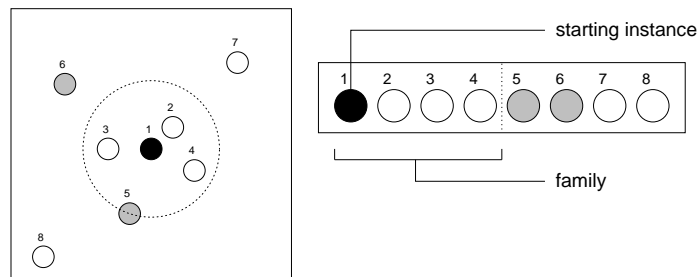


Figure 1: An example of a family in a two-dimensional instance space (left). The family, at the inside of the dotted circle, spans the focus instance (black) and the three nearest neighbors labeled with the same class (white). When ranked in the order of distance (right), the family boundary is put immediately before the first instance of a different class (grey).

instances. Generalized instances can be represented by conjunctions of *disjunctions* of feature values, or rules with wild-cards.

Here, we describe FAMBL2, belonging to the third group of carefully-abstracting memory-based learning algorithms. FAMBL2 merges groups of very similar instances (families) into family expressions. The core idea of FAMBL2 is to transform an instance base into a set of *instance family expressions*. First, we outline the ideas and assumptions underlying FAMBL2. We then give a procedural description of the learning algorithm.

2.1 Instance families: definition

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

Thus, families are class clusters, 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

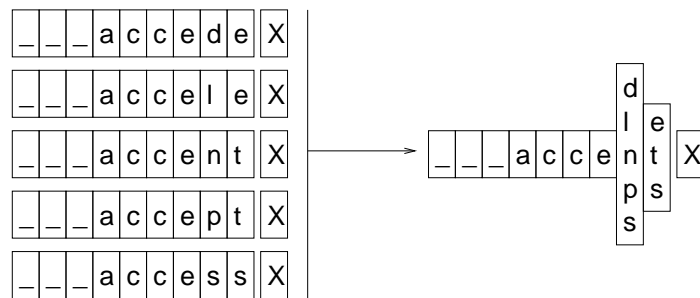


Figure 2: An example of family creation in FAMBL2. Four grapheme-phoneme instances, along with their token occurrence counts (left) are merged into a family expression (right).

the extremes of total disjunctivity (one instance per cluster) and no disjunctivity (one cluster per class). Many types of language data appear to be quite disjunct (Daelemans, Van den Bosch and Zavrel 1999a). In highly disjunct data, classes are scattered among many small clusters, which means that instances have few nearest neighbors of the same class on average.

Figure 1 illustrates how FAMBL2 determines the family of an instance in a simple two-dimensional example instance space. All nearest neighbors of a starting instance (marked by the black dot) are searched and ranked in the order of their distance to the starting instance. Although there are five instances of the same class in the example space, the family of the starting instance contains only three instances, since its fourth-nearest instance is of a different class.

Families are converted in FAMBL2 to *family expressions*, which are hyperrectangles, by merging all instances belonging to that family simultaneously. Figure 2 illustrates the creation of a family expression from an instance family. The general modus of operation of FAMBL2 is that it picks instances from an instance base one by one from the set of instances that are not already part of a family. For each newly-picked instance, FAMBL2 determines its family, generates a family expression from this set of instances, and then marks all involved instances as belonging to a family (so that they will not be picked as starting point or member of another family). FAMBL2 continues determining families until all instances are marked as belonging to a family.

The ordering of instances to be picked as centres of new families is important. Whenever instances are encapsulated in a family, they cannot be the starting point of another. However, one of these instances could have been a better starting point (e.g., because it is the central same-class nearest neighbor of a larger group of instances). Intuitively, it would be best to start building families with those in-

stances that are the middle instances of the largest families. Although this may appear circular, it is possible to estimate the suitedness of an instance to be a starting point for family generation, by computing its *class-prediction strength* (CPS), that expresses the success of that instance in predicting the class of its surrounding nearest-neighbor instances. Instances in the middle of large families will have high class-prediction strengths; when used for classification, they will serve as correct nearest neighbors to a large number of instances surrounding them.

Class-prediction strength of an instance i is typically defined (Salzberg 1990, Domingos 1995) as the number of times the instance is a nearest neighbor of a training instance regardless of its class (N), minus the number of these nearest neighbors that are of a different class (Ndf_i), divided by N to express a portion between 0.0 and 1.0: $e_i = \frac{N - Ndf_i}{N}$. An instance with class-prediction strength $e = 1.0$ is a perfect predictor of its own class; an e near or at 0.0 indicates that the family is a bad predictor. As argued in (Domingos 1995), this “raw” class prediction strength has a bias towards low-frequent instances that is sometimes unwanted: it assigns a maximal score of 1.0 to an instance when it is used correctly as a nearest neighbor only once. The Laplace correction is a common operation to favor high-frequent over low-frequent instances with the same raw score. Laplace correction introduces the number of classes, c , into the equation: $e_i = \frac{(N - Ndf_i) + 1}{N + c}$.

To compute CPS, we perform an auto-classification test with standard k -NN with (by default) information-gain-ratio feature weighting as implemented in the IB1-IG algorithm (Daelemans and Van den Bosch 1992, Daelemans et al. 1997b)¹. In this experiment, k is set to 3. This means that not only all nearest (equidistant) neighbors of an instance that differ in one or two features are taken into account, but also the instance itself. This ensures that all instances receive some baseline non-null score, reflecting the intuition that in language data, also low-frequent events may reoccur and thus be a nearest neighbor to a new occurrence of themselves (unless they are true noise) (Daelemans et al. 1999a). The same $k = 3$ limit is used when searching for nearest neighbors in family creation. This means that family members are allowed to differ in two features maximally.

In the FAMBL2 algorithm, instances are ordered by their CPS, and are picked as starting points for new instances beginning with the instance with the highest CPS. This is the key difference with the original FAMBL algorithm, in which starting points were selected randomly (Van den Bosch 1999). To summarize, a pseudo-code description of the learning phase is given in Figure 3.

After learning, the original instance base is discarded, and further classification is based only on the set of family expressions yielded by the family-extraction phase. Classification in FAMBL2 works analogously to classification in pure memory-

¹Auto-classification with IB1-IG is performed using the TiMBL software package, version 3.0.2 (Daelemans, Zavrel, Van der Sloot and Van den Bosch 1999b).

Procedure FAMBL LEARNING PHASE:

Input: A training set TS of instances $I_{1\dots n}$, each instance being labeled with a family-membership flag set to $FALSE$

Output: A family set FS of family expressions $F_{1\dots m}$, $m \leq n$

$i = f = 0$

1. Determine the class-prediction strength of all instances $I_{1\dots n}$ in TS and order them (largest CPS first)
 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 instances with family-membership flag $FALSE$. Nearest-neighbor instances of a different class with family-membership flag $TRUE$ are still used for marking the boundaries of the family.
 - Select all members in NS that fall within the 3 closest k buckets ($k = 3$) and remove all other instances from NS
 - Set the membership flags of I_i and all remaining instances in NS to $TRUE$
 - Merge I_i and all instances in NS into the family expression F_f and store this expression along with a count of the number of instance merged in it
 - $f = f + 1$
-

Figure 3: Schematized overview of the learning (family-extraction) phase in FAMBL.

based learning: a match is made between a new test instance and all stored family expressions. When a family expression records a disjunction of values for a certain feature, matching is perfect when one of the disjunctive values matches the value at that feature in the new instance. When two or more family expressions of different classes match equally well with the new instance, the class is selected with the highest occurrence summed over the matching expressions. When the tie remains, the class is selected that occurs the most frequently in the complete family expression set.

We conclude our description of the FAMBL2 algorithm by noting that FAMBL2 allows for the inclusion of informational abstraction in the form of feature-weighting, instance-weighting and value-difference metrics. For comparison with IB1-IG, as described in the next section, we have included information-gain-ratio feature weighting in FAMBL2. Weighting metrics are likely to have a profound effect on family extraction. For example, a study by (Van den Bosch 1997) suggests that using information-gain feature weighting (Quinlan 1986) in pure memory-based learning (viz. IB1-IG, (Daelemans and Van den Bosch 1992)), can yield considerably bigger families.

3 Effects of family generalization

In our comparative experiments between FAMBL2 and standard IB1-IG, we are interested in the differences caused by FAMBL2’s family generalization stage. Given a test instance, FAMBL2 and IB1-IG may assign the same correct or false classification based on different classifications (e.g. one family in FAMBL2, and three instances in IB1-IG), but they may also disagree – in which case one of them may be right, or they may be both wrong.

An illustration of a difference that actually occurs between FAMBL2 and IB1-IG trained on a dataset of English word pronunciation, is the following. Consider a small family generated by FAMBL2, from two instances representing the /l/ pronunciation of the “l” in “singularize” and “angularity”, supposing that instances represent one focus letter and four left and right neighbor letters to represent context. This family is generalized in FAMBL2 as “[i or a][n][g][u][l][a][r][i][z or t]”. Upon presentation of the instance representing the unseen word “singularity”, the generalized family expression offers a complete match with the new instance, due to the disjunction in the [i or a] and [z or t] parts of the expression, producing the correct /l/ pronunciation. Given the same test word, IB1-IG would yield two best-matching nearest neighbors that both would mismatch on one feature, while producing the same correct classification. In general, using family expressions for classification strengthens the class votes of the instances generalized in families: it can move their class votes up in the k -ranking (and never down).

To investigate the occurring differences in detail, we have collected results on

datasets representing English grapheme-phoneme conversion, Dutch diminutive noun formation, German plural noun formation, English part-of-speech tagging, English base-noun-phrase chunking, and English prepositional-phrase attachment. We briefly describe these six datasets here.

English grapheme-phoneme conversion (henceforth referred to as GP) is the mapping of English words to their phonemic counterparts, where the classification occurs at the letter level: mappings are made between letters in context and their appropriate phonemes. The grapheme-phoneme conversion data used in the experiments described here is derived from the CELEX lexical data base (Baayen, Piepenbrock and van Rijn 1993). We have used the first of the ten partitionings from the 10% 10-fold cross-validation experiment in (Van den Bosch 1999).

Dutch diminutive formation (henceforth DIM) is choosing the correct diminutive inflection to Dutch nouns out of five possible: *je*, *tje*, *pje*, *kje*, and *etje*, on the basis of phonemic word transcriptions, segmented at the level of syllable onset, nuclei and coda of the final three syllables of the word. The data stems from a study described in (Daelemans, Berck and Gillis 1997a).

German plural formation (henceforth PLU) is predicting the correct plural inflection (with possible umlaut) out of 8 possibilities, on the basis of singular nouns, represented by their phonemic representation segmented at the level of syllable onset, nuclei and coda of the final three syllables of the word. The data is also described and tested on in (Daelemans 2000, this volume).

English part-of-speech tagging (henceforth POS) is the disambiguation of syntactic classes of words in particular contexts. We assume a tagger architecture that processes a sentence from a disambiguated left to an ambiguous right context, as described in (Daelemans, Zavrel, Berck and Gillis 1996). The original data set for the part-of-speech tagging task, extracted from the LOB corpus, contains 1,046,151 instances; we have used a randomly-extracted 10% of this data.

English base-NP chunking (henceforth NP) is the segmentation of sentences into non-recursive NPs. (Veenstra 1998) used the Base-NP tag set as presented by (Ramshaw and Marcus 1995): *I* for inside a Base-NP, *O* for outside a Base-NP, and *B* for the first word in a Base-NP following another Base-NP. See (Veenstra 1998) for more details, and (Daelemans et al. 1999a) for a series of experiments on the original data set from which we have used a randomly-extracted 10%.

English PP attachment (henceforth PP) is the attachment of a PP in the sequence VP NP PP (VP = verb phrase, NP = noun phrase, PP = prepositional phrase). The data consists of four-tuples of words, extracted from the Wall Street Journal Treebank. From the original data set, used by (Ratnaparkhi,

task	generalization accuracy		number of families	fam. vs inst. compression
	IB1-IG	FAMBL2		
GP	88.1% (5975/6781)	87.9% (5962/6781)	31862	41.2%
DIM	95.4% (377/395)	96.2% (380/385)	1893	46.3%
PLU	94.8% (2385/2517)	94.6% (2377/2517)	4742	61.5%
POS	96.6% (10105/10462)	96.6% (10102/10462)	21802	71.0%
NP	97.5% (2448/2512)	97.5% (2448/2512)	17386	22.2%
PP	80.8% (1932/2390)	79.9% (1909/2390)	6980	65.9%

Table 1: Generalization accuracies in percentages and absolute numbers of correctly classified test instances, of IB1-IG and FAMBL2 on single partitionings of the six tasks; numbers of families; and compression rates of FAMBL2 vs IB1-IG in terms of numbers of families vs numbers of instance types.

Reynar and Roukos 1994), (Collins and Brooks 1995), and (Zavrel, Daelemans and Veenstra 1997), (Daelemans et al. 1999a) took the train and test set together to form the particular data also used here.

In all experiments, both FAMBL2 and IB1-IG use information-gain-ratio feature weighting (Quinlan 1986), which appears crucial in producing adequate k rankings; an important difference with standard AM (Daelemans et al. 1997b). For each dataset we generated a single random partitioning into a 90% training set and a 10% test set. Classification was done by both algorithms using $k = 1$: IB1-IG finds the set of closest nearest neighbors that all differ in the same zero or more features, and FAMBL2 does the same for families.

Table 1 displays the overall generalization accuracies yielded by the two algorithms on the six test sets. Accuracy differences are small. Yet, reasonable compression is obtained in the number of families produced by FAMBL2 when compared to the number of instance types (i.e. the number of unique instances, without duplicates) maintained in IB1-IG’s memory. These results are in line with the findings reported in (Van den Bosch 1999): FAMBL(2) compresses, but does not improve generalization accuracy as compared to IB1-IG.

There are more differences in the classifications made by the two algorithms, when focusing on the instance level, and looking for differences in the sense of the “singularize” – “angularity” – “singularity” example given above. First, Table 2 lists the average distance between a test instance and its nearest neighbors in IB1-IG and FAMBL2. These results indicate that classifications by FAMBL2 are indeed based on nearest neighbors at closer distances: apparently, as with the “singularity” example, at least some nearest neighbors are merged and appear as closer families in FAMBL2 classification. On average, FAMBL2 finds nearest neighbors at about 5%

task	average distance to nearest neighbor		% closer
	IB1-IG	FAMBL2	
GP	0.119	0.115	3.3%
DIM	0.075	0.071	5.3%
PLU	0.028	0.026	5.8%
POS	0.153	0.141	7.8%
NP	0.151	0.151	0.1%
PP	0.054	0.052	3.3%

Table 2: Average distance between test instances and their nearest neighbors for IB1-IG and FAMBL2, and the percentage of distance decrease obtained by FAMBL2, measured on the six tasks.

closer distance.

Second, Table 3 displays detailed counts of cases in which the two algorithms disagree on the classification of a test instance. We discern five possible situations with disagreements:

1. both algorithms are wrong;
2. IB1-IG is right, although FAMBL2 found one or more nearest families at closer distance;
3. IB1-IG is right, finding nearest neighbors at the same distance as FAMBL2, but having a class distribution in which the correct class is the most frequent, while FAMBL2 has an incorrect class as most frequent (because one or more families entered the $k = 1$ nearest-neighbor set, carrying incorrect classes with them);
4. FAMBL2 is right, finding one or more nearest families at closer distance than the nearest neighbors found by IB1-IG;
5. FAMBL2 is right, finding nearest neighbors at the same distance as IB1-IG, but having a class distribution in which the correct class is the most frequent, while IB1-IG has an incorrect class as most frequent.

The five columns in Table 3 display the absolute numbers within these five outcome types yielded by the two algorithms on the six tasks. FAMBL2 and IB1-IG agree completely on the NP task. In the majority of the disagreements, one of the two algorithms has the better class distribution, while both have nearest neighbors at the same close distance (situations 3 and 5). Overall, the results confirm that FAMBL2 does classify differently from IB1-IG, but the net effect as compared to the accuracies yielded by IB1-IG is slightly negative, and close to zero.

task	total #	both	IB1-IG right		FAMBL2 right	
	disagree	wrong	further	distr.	closer	distr.
GP	102	15	24	26	17	20
DIM	3	0	0	0	0	3
PLU	48	8	9	15	1	15
POS	76	9	10	25	5	27
NP	0	0	0	0	0	0
PP	131	0	21	56	11	43
total	360	32	64	122	34	108

Table 3: Test classification disagreement statistics. See text for explanation.

4 Discussion

First, we summarize the findings reported in the previous section, and draw conclusions from them. We then discuss the relation between memory-based learning, careful abstraction in FAMBL2, and analogical modeling.

4.1 Summary of findings and conclusions

Merging instances to form families, and use these precompiled families to base classifications on, is a close alternative to standard memory-based learning as implemented in the IB1-IG algorithm. FAMBL2 is able to reach comparable levels of generalization accuracy, while obtaining memory compression rates ranging from 20 to 80 %. On the other hand, its learning phase is a computationally costly (though not exponential) procedure, as it involves a complete memory-based classification of the training set.

The effect of merging instances to families, thereby opening the possibility that they jointly end up in a less distant k -level of nearest neighbors, has a negligible net effect on generalization accuracy as compared to standard memory-based learning. When a family moves its class up a k -level, results show that this improves and deteriorates class distributions roughly equally often.

In sum, pure memory-based learning remains a recommendable choice due to its simplicity. Unlike FAMBL2 and AM, it bases its classifications on very local information. With natural language processing tasks, this seems to be the best overall strategy available. No reported results with FAMBL (Van den Bosch 1999) or FAMBL2, or results obtained in comparisons between standard memory-based learning and AM (Daelemans 2000, this volume) show a trend towards advantage of using more global information (i.e. beyond a low, fixed k of nearest neighbors)

in classification.

4.2 Relation with analogical modeling

In terms of the original AM algorithm (Skousen 1989), standard k -NN classification takes only the instances in the k most specific supracontexts that contain observations, whether these are homogeneous or not, as the basis for extrapolating the output class. The lack of a check on homogeneity may be a cause for the general finding that increasing k is often detrimental to generalization accuracy with standard k -NN on language processing tasks (Daelemans et al. 1999a). In FAMBL2, with the same k , generating a disjunction of values in the generalized family expression entails an explicit *union* of adjacent homogeneous supracontexts that have the same class label. In classification, these joined supracontexts henceforth act as one, representing a potentially higher number of instance pointers and hence a higher analogical effect in further classification. A major difference with AM is that FAMBL precompiles its families, usually down to a set of family expressions that is (considerably) smaller than the original instance set. In AM, generating the analogical set is done for each test instance. There can be many more analogical sets than training instances. It would, however, be interesting to explore the possibilities of precompiling within AM a limited set of analogical sets on the basis of the training set, before classification.

To conclude, AM strongly suggests that more global data should be encapsulated in analogical sets than just the k closest matches. Implementing this general idea in a strictly limited manner as family generalization appears to be a valid step in bridging the gap between the k -NN and AM approaches and finding the general class of algorithms that combines the best of both worlds.

Acknowledgements

The author wishes to thank Walter Daelemans, Jakub Zavrel and the other members of the ILK (Tilburg) and CNTS (Antwerp) research groups for fruitful discussions and criticisms. This research has been made possible by a fellowship of the Royal Netherlands Academy of Arts and Sciences.

References

- Aha, D. W.(1997), Lazy learning: Special issue editorial, *Artificial Intelligence Review* **11**, 7–10.
- Aha, D. W., Kibler, D. and Albert, M.(1991), Instance-based learning algorithms, *Machine Learning* **6**, 37–66.

- Atkeson, C., Moore, A. and Schaal, S.(1997), Locally weighted learning, *Artificial Intelligence Review* **11**(1–5), 11–73.
- Baayen, R. H., Piepenbrock, R. and van Rijn, H.(1993), *The CELEX lexical data base on CD-ROM*, Linguistic Data Consortium, Philadelphia, PA.
- Collins, M. and Brooks, J.(1995), Prepositional phrase attachment through a backed-off model, *Proc. of Third Workshop on Very Large Corpora*, Cambridge.
- Cover, T. M. and Hart, P. E.(1967), Nearest neighbor pattern classification, *Institute of Electrical and Electronics Engineers Transactions on Information Theory* **13**, 21–27.
- Daelemans, W.(2000, this volume), A comparison of analogical modeling of language to memory-based language processing, in xxx (ed.), xxx, xxx, pp. xx–xx.
- Daelemans, W. and Van den Bosch, A.(1992), Generalisation performance of backpropagation learning on a syllabification task, in M. F. J. Drossaers and A. Nijholt (eds), *Proc. of TWLT3: Connectionism and Natural Language Processing*, Twente University, Enschede, pp. 27–37.
- Daelemans, W., Berck, P. and Gillis, S.(1997a), Data mining as a method for linguistic analysis: Dutch diminutives, *Folia Linguistica*.
- Daelemans, W., Van den Bosch, A. and Weijters, A.(1997b), IGTtree: using trees for compression and classification in lazy learning algorithms, *Artificial Intelligence Review* **11**, 407–423.
- Daelemans, W., Van den Bosch, A. and Zavrel, J.(1999a), Forgetting exceptions is harmful in language learning, *Machine Learning* **34**(1–3), 11–43.
- Daelemans, W., Zavrel, J., Berck, P. and Gillis, S.(1996), MBT: A memory-based part of speech tagger generator, in E. Ejerhed and I. Dagan (eds), *Proc. of Fourth Workshop on Very Large Corpora*, ACL SIGDAT, pp. 14–27.
- Daelemans, W., Zavrel, J., Van der Sloot, K. and Van den Bosch, A.(1999b), TiMBL: Tilburg Memory Based Learner, version 3.0, reference manual, *Technical Report ILK-0001*, ILK, Tilburg University.
- Devijver, P. A. and Kittler, J.(1980), On the edited nearest neighbor rule, *Proceedings of the Fifth International Conference on Pattern Recognition*, The Institute of Electrical and Electronics Engineers.
- Devijver, P. A. and Kittler, J.(1982), *Pattern recognition. A statistical approach*, Prentice-Hall, London, UK.
- 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.
- Fix, E. and Hodges, J. L.(1951), Discriminatory analysis—nonparametric discrimination; consistency properties, *Technical Report Project 21-49-004, Report No. 4*, USAF School of Aviation Medicine.
- Gates, G. W.(1972), The reduced nearest neighbor rule, *IEEE Transactions on Information Theory* **18**, 431–433.

- Hart, P. E.(1968), The condensed nearest neighbor rule, *IEEE Transactions on Information Theory* **14**, 515–516.
- Kolodner, J.(1993), *Case-based reasoning*, Morgan Kaufmann, San Mateo, CA.
- Quinlan, J.(1986), Induction of Decision Trees, *Machine Learning* **1**, 81–206.
- Ramshaw, L. and Marcus, M.(1995), Text chunking using transformation-based learning, *Proc. of Third Workshop on Very Large Corpora*, pp. 82–94.
- Ratnaparkhi, A., Reynar, J. and Roukos, S.(1994), A maximum entropy model for prepositional phrase attachment, *Workshop on Human Language Technology*, ARPA, Plainsboro, NJ.
- Salzberg, S.(1990), *Learning with nested generalised exemplars*, Kluwer Academic Publishers, Norwell, MA.
- Salzberg, S.(1991), A nearest hyperrectangle learning method, *Machine Learning* **6**, 277–309.
- Shavlik, J. W. and Dietterich, T. G. (eds)(1990), *Readings in Machine Learning*, Morgan Kaufmann, San Mateo, CA.
- Skousen, R.(1989), *Analogical modeling of language*, Kluwer Academic Publishers, Dordrecht.
- Stanfill, C.(1987), Memory-based reasoning applied to English pronunciation, *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, Morgan Kaufmann, Los Altos, CA, pp. 577–581.
- Stanfill, C. and Waltz, D.(1986), Toward memory-based reasoning, *Communications of the ACM* **29**(12), 1213–1228.
- Van den Bosch, A.(1997), *Learning to pronounce written words: A study in inductive language learning*, PhD thesis, Universiteit Maastricht.
- Van den Bosch, A.(1999), Careful abstraction from instance families in memory-based language learning, *Journal for Experimental and Theoretical Artificial Intelligence* **11**(3), 339–368.
- Veenstra, J. B.(1998), Fast NP chunking using memory-based learning techniques, *Proceedings of BENELEARN'98*, Wageningen, The Netherlands.
- Wettschereck, D. and Dietterich, T. G.(1995), An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms, *Machine Learning* **19**, 1–25.
- Wilson, D.(1972), Asymptotic properties of nearest neighbor rules using edited data, *Institute of Electrical and Electronic Engineers Transactions on Systems, Man and Cybernetics* **2**, 408–421.
- Zavrel, J., Daelemans, W. and Veenstra, J.(1997), Resolving PP attachment ambiguities with memory-based learning, in M. Ellison (ed.), *Proc. of the Workshop on Computational Language Learning (CoNLL'97)*, ACL, Madrid.