In: *Proceedings of CoNLL-2000 and LLL-2000*, pages 73–78, Lisbon, Portugal, 2000.

Using Induced Rules as Complex Features in Memory-Based Language Learning

Antal van den Bosch
ILK / Computational Linguistics
Tilburg University, The Netherlands
Antal.vdnBosch@kub.nl

Abstract

An extension to memory-based learning is described in which automatically induced rules are used as binary features. These features have an "active" value when the left-hand side of the underlying rule applies to the instance. The RIPPER rule induction algorithm is adopted for the selection of the underlying rules. The similarity of a memory instance to a new instance is measured by taking the sum of the weights of the matching rules both instances share. We report on experiments that indicate that (i) the method works equally well or better than RIPPER on various language learning and other benchmark datasets; (ii) the method does not necessarily perform better than default memory-based learning, but (iii) when multi-valued features are combined with the rule-based features, some slight to significant improvements are observed.

1 Rules as features

A common machine-learning solution to classification problems is rule induction (Clark and Niblett, 1989; Quinlan, 1993; Cohen, 1995). The goal of rule induction is generally to induce a set of rules from data, that captures all generalisable knowledge within that data, and that is as small as possible at the same time. Classification in rule-induction classifiers is based on the firing of rules on a new instance, triggered by matching feature values to the left-hand side of the rule. Rules can be of various normal forms, and can furthermore be ordered. The appropriate content and ordering of rules can be hard to find, and at the heart of most rule induction systems are strong search algorithms that attempt to minimise search through the space of possible rule sets and orderings.

Although rules appear quite different from instances as used in memory-based or instance-based learning (Aha et al., 1991; Daelemans and Van den Bosch, 1992; Daelemans et al., 1997b) there is a continuum between them. Rules can be seen as generalised instances; they represent the set of training instances with the same class that match on the conditions on the left-hand side of the rule. Therefore, classification strategies from memory-based learning can naturally be applied to rules. For example, (Domingos, 1996) describes the RISE system, in which rules are (carefully) generalised from instances, and in which the k-NN classification rule searches for nearest neighbours within these rules when classifying new instances.

Often, the sets of instances covered by rules overlap. In other words, seen from the instance perspective, a single instance can match more than one rule. Consider the schematic example displayed in Figure 1. Three instances with three multi-valued features match individually with one or two of the four rules; for example, the first instance matches with rule 1 (if $f1 = A$ then $c = Z$) and with rule 3 (if $f2 = C$ then $c = Z$).

Pursuing this reasoning, it is possible to index instances by the rules that apply to them. For example, in Figure 1, the first instance can be indexed by the "active" rule identification numbers 1 and 3. When the left-hand sides of rules are seen as complex features (in which the presence of some combination of feature values is queried) that are strong predictors of a single class, indexing instances by the rules that apply to them is essentially the same as representing instances by a set of complex features.

Note that when a rule matches an instance, this does not guarantee that the class of the instance is identical to the rule's predicted class – many rules will classify with some amount of

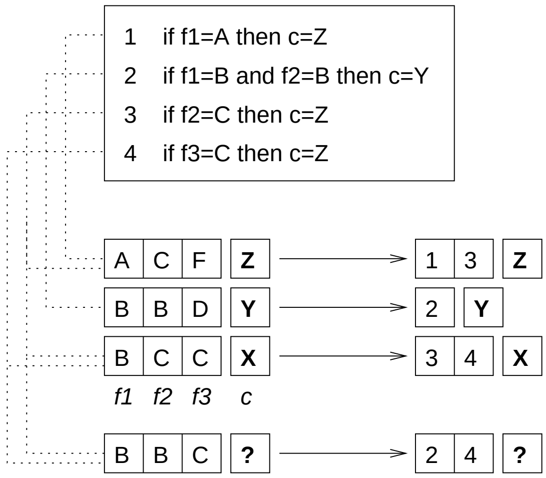

Figure 1: Schematic visualization of the encoding of multi-valued instances via matching rules to rule-indexed instances, characterised by the numbers of the rules that match them. $f1$, $f2$, and $f3$ represent the three features. c represents the class label.

error. In Figure 1, the third memory instance matches rules 3 and 4 which both predict a Z, while the instance itself has class X.

Now when instances are represented this way, they can be used in k-NN classification. Each complex feature then becomes a binary feature, that can also be assigned some weight (e.g., gain-ratio feature weights, chi-square, or equal weights (Daelemans et al., 2000)); when a memory instance and a new test instance share complex features, their similarity becomes the sum of the weights of the matching features. In Figure 1, a new instance (bottom) matches rules 2 and 4, thereby (partially) matching the second and third memory instances. If, for example, rule 4 would have a higher overall weight than rule 2, the third memory instance would become the nearest neighbor. The k-NN rule then says that the class of the nearest neighbour transfers to the new instance, which would mean that class X would be copied – which is a different class than those predicted either by rule 2 or 4. This is a marked difference with classification in RIPPER, where the class is assigned directly to the new instance by the rule that fires first. It can be expected that many classifications in this approach would be identical to those made by RIPPER, but it is possible that the k-NN approach has some consistent advantage in the cases where classification diverges.

In this paper we investigate some effects of recoding instances by complex features induced by an external rule-induction algorithm, and show that the approach is promising for language learning tasks. We find that the method works equally well or better than RIPPER on various language learning and other benchmark datasets. However, the method does not necessarily perform better than default memory-based learning. Only when the rule-indexing features are *added* to the original multi-valued features, improvements are observed.

2 Rule-Based Memory: algorithm

A new memory-based learning variant RBM, which stands for Rule-Based Memory, implements the ideas described in the previous section using the following procedure: given a training set and a test set of a certain classification task, (1) apply RIPPER (Cohen, 1995) to the training set, and collect the set of induced rules; (2) recode the instances in the training and test set according to these rules; (3) apply the basic memory-based learning algorithm IB1-IG to the recoded training set, and k-NN-classify the recoded test set. We describe each of these steps briefly here.

RIPPER (Cohen, 1995) is a fast rule induction algorithm that splits the training set in two. On the basis of one part it induces rules in a straightforward way, with potential overfitting. When the induced rules classify instances in the other part of the split training set below some classification accuracy threshold, they are not stored. Rules are induced per class, in a certain class ordering. By default, the ordering is from low-frequency classes to high frequency classes, leaving the most frequent class as the default rule, which is generally beneficial for the total description length of the rule set. In our experiments, we let RIPPER order the rules from high-frequent to low-frequent, the idea being that this method would yield more complex features.

Then, the rule set was taken as the basis for recoding both the training and test set, as schematically visualised in Figure 1. As with the training material, each test set was recoded in batch, but this could have been done on-line during classification without much computational overhead. For each language task we experimented on, we performed 10-fold cross validation tests, so ten different train-test partitions were produced (Weiss and Kulikowski, 1991) that were recoded, and then tested on. Tests were performed with the TiMBL software package (Daelemans et al., 2000), using the software's dedicated routines for handling binary features. The default IB1-IG algorithm was used (for details, consult (Aha et al., 1991; Daelemans and Van den Bosch, 1992; Daelemans et al., 1997b), with gain ratio selected as feature weighting metric.

3 Results

We performed experiments on the following five language data sets – More details on numbers of features, values per features, number of classes and number of instances are displayed in Table 1:

Diminutive formation (henceforth DIM): 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, nucei and coda of the final three syllables of the word. The data stems from a study described in (Daelemans et al., 1997a).

Grapheme-phoneme conversion (GPSM): the conversion of a window of nine letters to the phonemic transcription of the middle letter. From the original data set described in (Van den Bosch, 1997) a 10% subset was drawn.

Base-NP chunking (NPSM): the segmentation of sentences into non-recursive NPs. (Veenstra, 1998) used the Base-NP tag set as presented in (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., 1999) for a series of experiments on the original data set from which we have used a randomly-extracted 10%.

Part-of-speech tagging (POSSM): 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 et al., 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.

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

Table 2 lists the average (10-fold cross-validation) accuracies, measured in percentages of correctly classified test instances, of IB1-IG, RIPPER, and RBM on these five tasks. The clearest overall pattern in this table is the high accuracy of IB1-IG, surpassed only twice by RBM on the DIM and NPSM tasks (significantly, according to one-tailed t-tests, with $p < 0.05$). On the other three tasks, IB1-IG outperforms RBM. RIPPER performs significantly more accurately than IB1-IG only on the DIM task. Once again, evidence is collected for the global finding that forgetting parts of the training material, as obviously happens in rule induction, tends to be harmful to generalisation accuracy in language learning (Daelemans et al., 1999).

A surprising result apparent in Table 2 is that RBM never performs worse than RIPPER; in fact, it performs significantly more accurately than RIPPER with the GPSM, NPSM, and POSSM tasks. There appears to be an advantage in the k-NN approach to rule matching and voting, over the RIPPER strategy of ordered rule firing, with these tasks.

Another advantage, now of RBM as opposed to IB1-IG, is the reduced memory requirements and resulting speed enhancements. As listed in Table 3, the average number of rules in the rule sets induced by RIPPER range between 29 and 971. Averaged over all tasks, the rules have on

Data set	# Feat.	# Values of feature 1	2	3	4	5	6	7	8	9	10	11	12	# Class	# Data set instances
DIM	11	3	51	19	40	3	61	20	79	2	64	18	43	5	3950
GPSM	9	42	42	42	42	41	42	42	42	42				61	67,575
POS	5	155	157	414	395	384								159	104,617
NP	11	5961	5911	5895	5908	51	50	55	49	3	3	3		3	25,114
PP	4	3474	4612	68	5780									2	23,898

Table 1: Specifications of the five investigated language learning tasks: numbers of features, values per feature, classes, and instances. The rightmost column gives the total number of values times the number of classes.

Task	% Correct test instances IB1-IG	RIPPER	RBM
DIM	96.2 ± 0.6	96.9 ± 0.7 $*$	96.9 ± 0.7 $*$
GPSM	88.9 ± 0.6	80.4 ± 0.5	83.3 ± 0.5 $+$ $\surd$
NPSM	97.2 ± 0.3	96.9 ± 0.4	97.5 ± 0.4 $*$ $\surd$
POSSM	96.6 ± 0.2	94.3 ± 0.2	95.0 ± 0.2 $+$ $\surd$
PP	82.0 ± 0.5	77.0 ± 0.7	77.0 ± 0.6 $+$

Table 2: Average generalisation accuracies of IB1-IG, RIPPER, and RBM on five language learning tasks. '$*$' denotes significantly better accuracy of RBM or RIPPER over IB1-IG with $p < 0.05$. '+' denotes significance in the reverse direction. $\surd$ denotes significantly better accuracy of RBM over RIPPER with $p < 0.05$.

Task	RIPPER / RBM # rules	c/r	f/i	Classif. time (s) IB1-IG	RBM
DIM	61	2.5	1.3	1	1
GPSM	971	3.9	1.5	8	17
NPSM	72	2.8	1.8	19	1
POSSM	628	2.7	1.0	32	13
PP	29	3.0	0.3	19	1

Table 3: Average number of RIPPER rules, conditions per rule (c/r), and coded features per instance (f/i); and one-partition timings (s) of classification of test material in IB1-IG and RBM, for five language tasks.

average about two to four conditions (feature-value tests). More importantly, as the third column of Table 3 shows, the average number of *active* rules in instances is below two for all tasks. This means that in most instances of any of the five tasks, only one complex feature (bit) is active.

Especially with the smaller rule sets (DIM, NPSM, and PP – which all have few classes, cf. Table 1), RBM's classification is very speedy. It reduces, for example, classification of the NPSM test set from 19 seconds to 1 second[1]. Large rule sets (GPSM), however, can have adverse effects – from 8 seconds in IB1-IG to 17 seconds in RBM.

In sum, we observe two cases (DIM and NPSM) in which RBM attains a significant generalisation accuracy improvement over IB1-IG as well as some interesting classification speedup, but for the other tasks, for now unpredictably, generalisation accuracy losses and even a slowdown are observed. The latter occurs with GPSM, which has been analysed earlier as being extremely disjunct in class space, and therefore highly sensitive to the "forgetting exceptions is harmful" syndrome (Daelemans et al., 1999; Van den Bosch, 1999a).

The complex features used in RBM are taken as the only information available; the original information (the feature values) are discarded. This need not be the case; it is possible that the recoded instances are merged with their original feature-value vectors. We performed experiments in which we made this fusion; the results are listed in Table 4. Comparing the column labeled "IB1-IG+RBM, denoting the fusion variant, with the IB1-IG column, it can be seen that it reaches some modest error reduction percentages (rightmost column in Table 4). In fact, with NPSM and POSSM, it performs significantly better (again, according to one-tailed t-tests, with $p < 0.05$) than IB1-IG. On the other hand, adding the (average) 971 complex features to the nine multi-valued features in the

[1]Timings are measured on one partition, using a dual-Pentium II 200 Mhz machine running Linux 2.2.

Task	% Correct test instances IB1-IG	IB1-IG+RBM	% Error reduct.
DIM	96.2 ± 0.6	96.2 ± 0.7	0.0
GPSM	88.9 ± 0.6	88.6 ± 0.4	-2.3
NPSM	97.2 ± 0.3	97.6 ± 0.4 *	6.0
POSSM	96.6 ± 0.2	96.8 ± 0.2 *	4.6
PP	82.0 ± 0.5	82.1 ± 0.5	1.0

Table 4: Average generalisation accuracies of IB1-IG and IB1-IG + RBM, and the percentage of error reduction, on five language learning tasks. '*' denotes significantly better accuracy of IB1-IG+RBM over IB1-IG with $p < 0.05$.

Task	% Correct test instances IB1-IG	RBM	IB1-IG+RBM
CAR	93.9 ± 2.1	98.9 ± 0.8	97.2 ± 1.3
NURSERY	94.6 ± 0.6	98.6 ± 0.5	98.7 ± 0.2
SPLICE	91.7 ± 1.1	89.0 ± 2.1	92.7 ± 1.7

Table 5: Average generalisation accuracies of IB1-IG, RIPPER, and RBM on three machine-learning benchmark tasks.

GPSM causes a slight drop in performance – and a slowdown.

4 Discussion

Representing instances by complex features that have been induced by a rule induction algorithm appears, in view of the measured accuracies, a viable alternative approach to using rules, as compared to standard rule induction. This result is in line with results reported by Domingos on the RISE algorithm (Domingos, 1995; Domingos, 1996). A marked difference is that in RISE, the rules are the *instances* in k-NN classification (and due to the careful generalisation strategy of RISE, they can be very instance-specific), while in RBM, the rules are the *features* by which the original instances are indexed. When a nearest neighbor is found to a query instance in RBM, it is because the two instances share *one or more* matching rules. The actual classification that is transferred from the memory instance to the new instance is just the classification that this memory item is stored with – it may well be another class than any of its matching rules predict.

Second, the method is a *potentially* helpful extension to memory-based learning of language processing tasks. When nothing is known about the characteristics of a language processing data set, it is advisable to *add* the induced complex features to the original features, and do k-NN classification on the combination; it is not advisable to base classification only on the induced complex features. On its own, the method basically inherits a part of the detrimental "forgetting exceptions is harmful" effect from its rule-induction source (this effect is stronger when a data set is more disjunct (Daelemans et al., 1999)). Although RBM performs equal to or better than RIPPER, it often does not regain the level of IB1-IG.

High disjunctivity appears to be a typical feature of language tasks (Van den Bosch, 1999b); other non-language tasks generally display less disjunctivity, which opens the possibility that the RBM approach may work well for some of these tasks. We performed pilot tests on three machine learning benchmark classification tasks (taken from the UCI benchmark repository (Blake and Merz, 1998)) with symbolic, multi-valued features. Table 5 displays the results of these experiments. Although the data set selection is small, the results of RBM and especially of IB1-IG+RBM are promising; the latter algorithm is consistently better than IB1-IG. More research and comparisons are needed to arrive at a broader picture.

An immediate point of further research lies in the external rule induction algorithm. First, RIPPER has options that have not been used here, but that may be relevant for the current issue, e.g. RIPPER's ability to represent sets of values at left-hand side conditions, and its flexibility in producing larger or smaller numbers of rules. Second, other rule induction algorithms exist that may play RIPPER's role, such as C4.5RULES (Quinlan, 1993).

More generally, further research should focus on the scaling properties of the approach (including the scaling of the external rule-induction algorithm), should investigate more and larger language data sets, and should seek comparisons with other existing methods that claim to handle complex features efficiently (Brill, 1993; Ratnaparkhi, 1997; Roth, 1998; Brants, 2000).

Acknowledgements

The author thanks the members of the Tilburg ILK group and the Antwerp CNTS group for fruitful discussions. This research has been made possible by a fellowship of the Royal Netherlands Academy of Arts and Sciences (KNAW).

References

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

C.L. Blake and C.J. Merz. 1998. UCI repository of machine learning databases.

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

E. Brill. 1993. A Corpus-Based Approach to Language Learning. Dissertation, Department of Computer and Information Science, University of Pennsylvania.

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

W. W. Cohen. 1995. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, Lake Tahoe, California.

M.J Collins and J. Brooks. 1995. Prepositional phrase attachment through a backed-off model. In *Proc. of Third Workshop on Very Large Corpora*, Cambridge.

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

W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. 1996. MBT: A memory-based part of speech tagger generator. In E. Ejerhed and I.Dagan, editors, *Proc. of Fourth Workshop on Very Large Corpora*, pages 14–27. ACL SIGDAT.

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

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

W. Daelemans, A. Van den Bosch, and J. Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning*, 34(1–3):11–43.

W. Daelemans, J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2000. TiMBL: Tilburg Memory Based Learner, version 3.0, reference manual. Technical Report ILK-0001, ILK, Tilburg University.

P. Domingos. 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.

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

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

L.A. Ramshaw and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *Proc. of Third Workshop on Very Large Corpora*, pages 82–94, June.

A. Ratnaparkhi, J. Reynar, and S. Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Workshop on Human Language Technology*, Plainsboro, NJ, March. ARPA.

A. Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. Technical Report cmp-lg/9706014, Computation and Language, http://xxx.lanl.gov/list/cmp-lg/, June.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the National Conference on Artificial Intelligence*, pages 898–904.

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

A. Van den Bosch. 1999a. Careful abstraction from instance families in memory-based language learning. *Journal for Experimental and Theoretical Artificial Intelligence*, 11(3):339–368.

A. Van den Bosch. 1999b. Instance-family abstraction in memory-based language learning. In I. Bratko and S. Dzeroski, editors, *Machine Learning: Proceedings of the Sixteenth International Conference*, pages 39–48, Bled, Slovenia.

J. Veenstra. 1998. Fast NP chunking using memory-based learning techniques. In *Proceedings of BENELEARN'98*, Wageningen, The Netherlands.

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

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