Rule Meta-learning for Trigram-Based Sequence Processing

Sander Canisius S.V.M.Canisius@uvt.nl
ILK / Computational Linguistics and AI, Tilburg University, The Netherlands

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

Walter Daelemans walter.daelemans@ua.ac.be
CNTS, Department of Linguistics, University of Antwerp, Belgium

Abstract

Predicting overlapping trigrams of class labels is a recently-proposed method to improve performance on sequence labelling tasks. In this method, sequence elements are effectively classified three times, therefore some procedure is needed to post-process those overlapping classifications into one output sequence. In this paper, we present a rule-based procedure learned automatically from training data. In combination with a memory-based leaner predicting class trigrams, the performance of this meta-learned overlapping trigram post-processor matches that of a handcrafted post-processing rule used in the original study on class trigrams. Moreover, on two domain-specific entity chunking tasks, the class trigram method with automatically learned post-processing rules compares favourably with recent probabilistic sequence labelling techniques, such as maximum-entropy markov models and conditional random fields.

1. Introduction

Many tasks in natural language processing involve the complex mapping of sequences to other sequences. One typical machine-learning approach to such sequence labelling tasks is to rephrase the sequence-to-sequence mapping task (where a sequence can have a variable length) as a decomposition into a sequence of local classification steps. In each step, one fixed-length feature vector is mapped to an isolated token in the output sequence. After all local classifications have been made, a simple concatenation of the predicted output tokens yields the complete output sequence.

The standard representational approach to decompose sequence processes into local-classification cases, is *windowing*. Within a window, fixed-length subsequences of adjacent input symbols, representing a certain contextual scope, are mapped to one output symbol, typically associated with one of the input symbols, for example the middle one. The fact that the classifier is only trained to associate subsequences of input symbols to single output symbols as accurately as possible is a problematic restriction: it may easily cause the classifier to produce invalid or impossible output sequences, since it is incapable of taking into account any decisions it has made earlier, or even decisions it might have to make further on in the input sequence.

Techniques attempting to circumvent this restriction can be categorised mainly into two general classes. One of those improves upon naive methods by optimising towards the most likely sequence of class labels, rather than the sequence of individually most likely labels. In order to find this most likely sequence in tasks where the class labels of sequence elements are strongly interrelated, it may be necessary to match sequence elements with class labels that are deemed sub-optimal by the underlying classifier, which only bases its decisions on local information. This class of techniques includes recently proposed methods such as maximum-entropy markov models (McCallum et al., 2000) and conditional random fields (Lafferty et al., 2001).

The other class is formed by techniques that rely more on the quality of the predictions made by the underlying classifier; they do not consider label sequences other than those derived from the class labels sug-

Appearing in *Proceedings of the 4^{th} Learning Language in Logic Workshop (LLL05)*, Bonn, Germany, 2005.

gested by the classifier to be the most-likely ones. Representatives of this class that have been used for sequence labelling tasks in the past are a feedback-loop method as used for example by Daelemans et al. (1996) with memory-based learning, and by Ratnaparkhi (1996) with maximum-entropy modelling, in which previous decisions are encoded as features in the current input of the classifier, and stacking (Wolpert, 1992), a term referring to meta-learning systems that learn to correct errors made by lower-level classifiers.

Predicting overlapping trigrams of class labels, a new method for performing sequence labelling proposed by Van den Bosch and Daelemans (2005), can be seen as a member of the latter class. That is, the technique can be applied to any classifier able to predict a most-likely class label for a given test instance; there is no requirement that the classifier also models the likelihood of other, sub-optimal classes as is required for the methods in the first class. However, with the trigram class method, the final label sequence is not simply obtained by concatenating the classifications for the individual sequence elements. With the trigram class method each sequence element is effectively classified three times; for this reason, some post-processing is required to determine the final label to be assigned to each sequence element.

Van den Bosch and Daelemans (2005) use a simple voting technique with a tie-breaking rule based on classifier-dependent confidence values for this purpose, but emphasise that this is just one possible, and rather simple technique to combine overlapping trigram classes. Other post-proecssing methods may prove to be more appropriate, for example by resulting in better performance scores, or by allowing the full potential of the n-gram class method (with $n > 3$) to be reached. This paper presents one such alternative, based on rule induction where features correspond to various logical assertions about the identity of the overlapping trigrams, or equalities between the three overlapping predictions.

The learned post-processing rules are tested in a pair of benchmark experiments, where a memory-based learner, which was the better performing classifier out of three different classifiers tested by Van den Bosch and Daelemans (2005), is combined with the class trigram method to perform two domain-specific entity chunking task. On these tasks, we evaluate the effect of using rule induction for combining overlapping trigrams and show that the class trigram method with a post-processing method based on rule induction is able to improve upon the baseline performance, with margins comparable to those obtained with a handcrafted combination procedure.

To measure the relative performance of the memory-based learning/class trigram combination, we also perform a series of experiments where three state-of-the-art probabilistic sequence labelling techniques – conditional markov models, maximum-entropy markov models, and conditional random fields – are applied to the same benchmark tasks. On one of the two tasks, the memory-based learner with trigram classes, outperforms all three probabilistic learners; on the other task, conditional random fields prove to be superior, with the memory-based learner ending second.

The structure of the paper is as follows. First, we introduce the two chunking sequence segmentation tasks studied in this paper, in Section 2. Section 3 introduces two approaches for automatically learning class trigram combination rules, and reports on experiments that evaluate the performance of the resulting combination procedures. Next, the class trigram method is empirically compared with three recent probabilistic sequence labelling methods in Section 4. Finally, Section 5 sums up and discusses the main empirical results.

2. Data and Methodology

The two data sets that have been used for this study are examples of sentence-level entity chunking tasks: concept extraction from general medical encyclopedic texts (henceforth MED), and labelling of DNA, RNA, protein, cellular, and chemical terms in MEDLINE abstracts (GENIA). MED is a data set extracted from a semantic annotation of parts of two Dutch-language medical encyclopedias. On the chunk-level of this annotation, there are labels for various medical concepts, such as disease names, body parts, and treatments, forming a set of twelve concept types in total. Chunk sizes range from one to a few tokens. Using a 90%–10% split for producing training and test sets, there are 428,502 training examples and 47,430 test examples.

> *Bij* [*infantiel botulisme*]$_{disease}$ *kunnen in extreme gevallen* [*ademhalingsproblemen*]$_{symptom}$ *en* [*algehele lusteloosheid*]$_{symptom}$ *optreden.*

The GENIA corpus (Tateisi et al., 2002) is a collection of annotated abstracts taken from the National Library of Medicine's MEDLINE database. Apart from part-of-speech tagging information, the corpus annotates a subset of the substances and the biological locations involved in reactions of proteins. Using a 90%–10% split for producing training and test sets, there are

458,593 training examples and 50,916 test examples.

> *Most hybrids express both* $[KBF1]_{protein}$ *and* $[NF\text{-}kappa\ B]_{protein}$ *in their nuclei, but one hybrid expresses only* $[KBF1]_{protein}$.

Apart from having a similar size, both data sets are alike in the sense that most words are outside chunks; many sentences may even contain no chunks at all. Thus, the class distributions of both tasks are highly skewed. In this respect the tasks differ from, for example, syntactic tasks such as part-of-speech tagging or base-phrase chunking, where almost all tokens are assigned a relevant class. However, for all tasks mentioned, whenever chunks are present in a sentence, there is likely to be interaction between them, where the presence of one chunk of a certain type may be a strong indication of the presence of another chunk of the same or a different type in the same sentence.

2.1. Experimental Setup

Van den Bosch and Daelemans (2005) tested their class trigram method with three different classifiers as base classifier; of those three classifiers, memory-based learning performed best; hence, the experiments in Section 3, where the class trigram method is evaluated, are performed using the memory-based learning or k-nearest neighbour algorithm (Cover & Hart, 1967) as implemented in the TiMBL software package (version 5.1) (Daelemans et al., 2004). The combination rules for merging the overlapping class trigrams produced by the base classifier are induced using RIPPER (Cohen, 1995).

The memory-based learning algorithm has algorithmic parameters that bias its performance; for example, the number of nearest neighbours, the distance metric, etc. The optimal values for these parameters may differ depending on the task to be learned. To obtain maximum performance, we optimised the parameter settings on each task using wrapped progressive sampling (WPS) (Van den Bosch, 2004), a heuristic automatic procedure that, on the basis of validation experiments internal to the training material, searches among algorithmic parameter combinations for a combination likely to yield optimal generalisation performance on unseen data. We used wrapped progressive sampling in all experiments.

The experiments described in Section 4 are performed using three different probabilistic sequence learning techniques. Conditional markov models and maximum-entropy markov models have been implemented on top of the maximum-entropy toolkit (version 20041229) by Zhang Le[1]. For conditional random fields, we used the implementation of MALLET (McCallum, 2002).

Instances for all experiments are generated for each token of a sentence, with features for seven-word windows of words and their (predicted) part-of-speech tags. The class labels assigned to the instances form an IOB encoding of the chunks in the sentence, as proposed by Ramshaw and Marcus (1995). In this encoding the class label for a token specifies whether the token is inside (I), outside (O), or at the beginning of a chunk (B). An additional type label appended to this symbol denotes the type of the chunk. The instances are used in exactly this form in all experiments for all algorithms; no feature selection or construction is performed to optimise the instances for a specific task or classifier. Keeping the feature vectors unchanged over all experiments and classifiers is arguably the most objective setup for comparing the results.

Generalisation performance is measured by the F-score ($\beta = 1$) on correctly identified and labelled entity chunks in test data. Experimental results are presented in terms of a mean score, and an approximate 90%-confidence interval; both of those are estimated with bootstrap resampling (Noreen, 1989). Confidence intervals are assumed to be centred around the mean, where the width of the halves at both sides of the mean is taken to be the maximum of the true widths obtained in the resampling process.

3. Predicting Class Trigrams

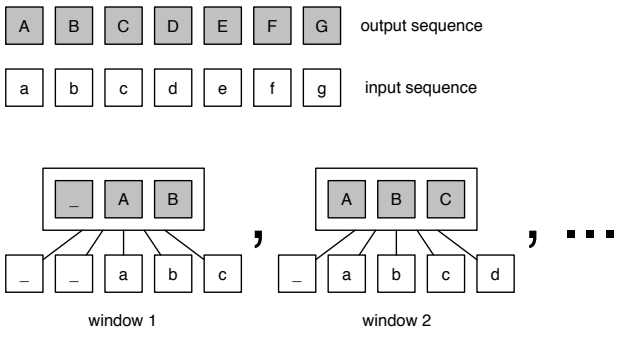

Figure 1. Windowing process with trigrams of class symbols. Sequences of input symbols and output symbols are converted into windows of fixed-width input symbols each associated with, in this example, trigrams of output symbols.

[1] `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`

As Van den Bosch and Daelemans (2005) argue, there is no intrinsic bound to what is packed into a class label associated to a windowed example. For example, complex class labels can span over trigrams of singular class labels. Figure 1 illustrates the procedure by which windows are created with class trigrams. Each windowed instance maps to a class label that incorporates three atomic class labels, namely the focus class label that was the original unigram label, plus its immediate left and right neighbouring class labels.

While creating instances this way is trivial, it is not entirely trivial how the output of overlapping class trigrams recombines into a normal string of class sequences. When the example illustrated in Figure 1 is followed, each single class label in the output sequence is effectively predicted three times; first, as the right label of a trigram, next as the middle label, and finally as the left label. The redundancy caused by predicting each class label three times may be exploited to do better than the classifier that only predicts each class label once. What is needed then, is a combination procedure that intelligently determines the final class label for each token, given a number of overlapping predictions.

Van den Bosch and Daelemans (2005) propose a simple procedure based on the observation that, in the case of overlapping class label trigrams, it is possible to vote over them. The voting scheme proposed by Van den Bosch and Daelemans (2005) returns the class label which receives the majority of votes (in this case, either two or three), or when all tree votes disagree (i.e. when majority voting ties), returns the class label of which the classifier is most confident. Classifier confidence, needed for tie-breaking, is a classifier-specific metric expressing the classifier's confidence in the correctness of the predicted class; for the memory-based learner it may be heuristically estimated by, for example, taking the distance of the nearest neighbour, which is the approach adopted by (Van den Bosch & Daelemans, 2005).

Clearly this scheme is one out of many possible schemes: other post-processing rules may be used, as well as other values of n (and having multiple classifiers with different n, so that some back-off procedure could be followed). Another interesting possibility, and the approach taken in the current study, is to apply meta-learning to the overlapping outputs of the first-stage classifier, so that a data-driven post-processing procedure is learned automatically from examples extracted from a first-stage classifier producing overlapping class trigrams.

In the current study, we developed two meta-learning combination procedures based on rule induction, in which the instances for the meta-learner describe the overlapping class trigrams in some form, and the classes to be predicted correspond to the combined unigram class label. The first method is a rather straightforward approach in which instances consist of features for the three overlapping class trigrams; the other is a more sophisticated procedure, where features correspond to logical assertions about matches between components of the overlapping trigrams. The exact details of both approaches are described in the remainder of this section, where in addition, their performance is evaluated on the two benchmark tasks MED, and GENIA.

Meta-learning based on class trigram features

A straightforward design for a meta-learner for combining multiple overlapping classifications into a single class label is a classifier trained on instances that simply encode the overlapping classifications as features, and the class labels for these meta-learning instances correspond to the combined classes to be predicted.

To generate training data for the rule inducer, an internal cross-validation experiment has been performed on the training data, resulting in a realistic set of examples of first-stage class trigram outputs. On the training set thus obtained, a rule inducer has been trained, leading to a rule set that predicts a single unigram class label given the three overlapping class trigrams predicted by the first-stage MBL classifier. The performance of this rule set on both the MED, and GENIA entity chunking tasks is presented in Table 1 under the "Learned 1" column, where it is compared with the performance of a unigram-class producing MBL classifier, and with the performance of the handcrafted post-processing procedure of Van den Bosch and Daelemans (2005).

Compared with the baseline performance, the class trigram method with this learned combination procedure attains a substantial performance increase, thereby both confirming the advantage of predicting class trigrams for sequence labelling tasks, and showing the possibility of using a rule inducer to automatically produce combination rules that successfully exploit the redundancy present in the overlapping class trigrams. However, the learned combination rules do not perform better than the handcrafted procedure, which outperforms the learned rules by approximately one point on both tasks.

Table 1. Comparison of generalisation performances of the baseline MBL classifier, and three trigram class approaches using different combination procedures. The best performance per task is printed in bold.

		Trigram post-processing		
Task	Baseline	Handcrafted	Learned 1	Learned 2
MED	64.7 ±0.95	67.5 ±1.09	66.7 ±0.88	**67.7** ±1.07
GENIA	55.7 ±1.14	60.1 ±1.01	58.9 ±1.11	**60.4** ±1.12

Meta-learning based on class overlap features

A huge disadvantage of the previous approach is the fact that the class trigrams encoded as features in the instances can only be treated as atomic symbols by the rule inducer; there is no way for a rule to refer to the left component of a trigram, let alone to draw parallels between a component of one trigram and that of another. To circumvent this limitation, we designed a more fine-grained description of the overlapping trigrams based on logical assertions about the components of the overlapping trigrams, and about matches between two trigrams regarding the class label of the token in focus. For example, the following overlapping trigrams

$t_{-1} = (O, O, A)$
$t_0 = (A, B, A)$
$t_{+1} = (A, O, C)$

would be encoded as follows.

$\neg match\_on\_focuspos(t_{-1}, t_0)$,
$match\_on\_focuspos(t_{-1}, t_{+1})$,
$\neg match\_on\_focuspos(t_0, t_{+1})$,
$\neg all\_agree\_on\_focuspos$,

$focussym(t_{-1}) = A$,
$focussym(t_0) = B$,
$focussym(t_{+1} = A)$

Here, the $focussym$ function returns the component label of the argument trigram describing the token currently in focus; that is, for t_{-1}, it returns the right symbol, for t_0, the middle, and for t_{+1}, the left. The $match\_on\_focuspos$ relation can then be defined as

$match\_on\_focuspos(x, y) \iff$
$focussym(x) = focussym(y)$

By using this representation language for the overlapping class trigrams, the rule inducer is guided to focus on the information about the focus class present in the trigrams. Again, training material for this experiment has been generated by performing an internal cross-validation on the training set.

As can be seen in Table 1 in the column marked "Learned 2", this instance description turns out to be a better choice than our previous attempt, allowing the rule inducer to produce combination rules that outperform those produced by the method described previously. It outperforms both the baseline approach, and the previous meta-learned post-processing method. Compared with the handcrafted method, the meta-learned post-processing procedure appears to perform slightly better, although this difference is nowhere near significance.

Analysis

The experiments described in this section show that rule induction can be used to produce a set of post-processing rules with which the class trigram method can outperform the baseline classifier. However, the fact that the best-performing learned combination procedure does not significantly outperform the handcrafted post-processing rule might seem surprising at first. It should be noted though that the majority voting rule is rather high-level rule, not easily expressed in the description language used in our experiments. Suppose we would replace the confidence-based tie-breaking rule in the handcrafted post-processing procedure by a rule that always selects the focus symbol of the middle trigram. In that case, a logical formulation of the majority voting procedure would probably look like the following decision list.

$\forall X($
$focussym(t_{-1}) = focussym(t_{+1}) = X \Rightarrow X$,
$focussym(t_0) = X \Rightarrow X)$

However, such use of variables is not possible in our description language. The best rule induction can do to match majority voting behaviour is to invent it separately for each class.

$focussym(t_{-1}) = focussum(t_{+1}) = I\text{–}disease \Rightarrow I\text{–}disease$,
$focussum(t_{-1}) = focussum(t_{+1}) = B\text{–}disease \Rightarrow B\text{–}disease$,
...
$focussym(t_0) = I\text{–}disease \Rightarrow I\text{–}disease$,
$focussym(t_0) = B\text{–}disease \Rightarrow B\text{–}disease$,
...

Table 2. Percentage of agreement between handcrafted post-processing and the meta-learned post-processing procedure in case there is a majority vote among the three overlapping class trigrams, and in case there is no such majority.

TASK	MAJORITY	NO MAJORITY
MED	99.16	30.80
GENIA	97.89	29.46

Given that the performance of the best learned combination procedure roughly equals that of the handcrafted post-processing rule, the interesting question is whether rule induction did in fact reinvent majority voting, or whether it produced an entirely different rule set that coincidentally results in comparable performance.

An inspection of the rules produced for MED offers a mixed view. For many classes, there are high-priority rules that dictate always believing the middle trigram, be it in various different formulations. For other classes, the highest-priority rule does first check for an overruling majority, before assigning the class suggested by the middle position in the middle trigram. In addition to these rules, which may be interpreted as voting-like behaviour, there are also more original rules that select the class suggested by, for example, the right trigram in favour of the one predicted by the middle trigram; or even rules that might be interpreted as small error correction rules.

For a more definite answer to the question whether the learned rules emulate a majority voting procedure, we computed two overlap metrics between the output of the handcrafted post-processing procedure and that of the meta-learned post-processing procedure: the first measures the percentage of agreement between the two in case there is a majority vote among the three overlapping class trigrams; the other measures the percentage of agreement in case there is no majority. The two metrics computed for both benchmark tasks are listed in Table 2. As can be seen, if there is a majority vote among the overlapping trigrams, there is almost exact agreement between the two different combination procedures. However, in the case of no majority, both methods agree on only 30 percent of the classifications.

These findings lead to the conclusion that the learned combination procedure does indeed implement majority voting. However, it differs from the original handcrafted post-processing rule in the way it deals with ties. This is hardly surprising since the information used for tie-breaking in the original voting rule – classifier confidence – is not available to the rule inducer generating the learned combination procedure.

4. Predicting Class Trigrams versus Probabilistic Output Sequence Optimisation

The class trigram method proposed by Van den Bosch and Daelemans (2005), and evaluated in a slightly modified form in the previous section, is one method for basing decisions for an individual token on the wider sequential context of this token. Both Van den Bosch and Daelemans (2005) and the current study show that predicting class trigrams is an effective method to improve upon a baseline classifier that classifies each token with respect to only a small local context. In order to evaluate the class trigram method with respect to more competitive reference scores, we also compared the method with another popular approach for improving sequence labelling: probabilistic sequence labelling techniques. In this section, three different methods based on this general approach are applied to the sample tasks: conditional markov models, maximum-entropy markov models, and conditional random fields.

4.1. Conditional Markov Model

Conditional markov models (CMM), as used for example by Ratnaparkhi (1996), supplement a standard maximum-entropy model with a feedback loop, in which a prespecified number of previous decisions of the classifier are fed back to the input as features for the current test instance. However, as maximum-entropy models do not simply predict a single most-likely class label, but rather model the entire conditional class probability distribution, classification of a token does not yield one partial labelling, namely, the partial labelling up to the current token followed by the current classification, but as many partial labellings as there are target labels, namely the partial labelling until the current token followed by any of the possible target labels.

As the use of a feedback loop makes the current classification depend on the results of previous classifications, each token in the sequence has to be classified in the context of each possible partial labelling up to that point. Clearly, this approach, if applied naively, gives rise to an exponential increase in possible partial labellings at each token. Therefore, CMMs employ a beam search to find the eventual best labelling. With beam search, at each point in time, only a prespecified number of partial labellings – those having highest probability – are considered for expansion, all the

Table 3. Comparison of generalisation performances of the MBL classifier predicting class trigrams, and each of the probabilistic methods. The best performances per task are printed in bold.

TASK	MBL	CMM	MEMM	CRF
MED	**67.7** ±1.07	59.7 ±1.07	60.3 ±1.13	63.4 ±0.95
GENIA	60.4 ±1.12	59.9 ±1.04	56.1 ±1.11	**62.8** ±1.08

other candidates are discarded.

When tested on the two sample tasks, CMM obtains the scores listed in the third column of Table 3. In comparison with the trigram-predicting MBL classifier, it performs considerably worse on MED, but similarly on GENIA.

4.2. Maximum-entropy Markov Model

A more recent probabilistic sequence labelling method is the maximum-entropy markov model (MEMM), proposed by McCallum et al. (2000). Derived from hidden markov models, MEMMs are modelled after a probabilistic state machine, in which, in the simplest case, a state corresponds to the output label of the previous token, and for each state, a separate conditional probability distribution determines the next state, that is, the output label for the current token, given the feature vector of this token. A slight modification of the Viterbi algorithm is used to determine the optimal path through the state machine given the input sequence.

The fourth column of Table 3 shows the performance of MEMM applied to the MED and GENIA tasks. On both tasks, MEMM is outperformed by the MBL classifier. For MED, MEMM's score is quite similar to that of CMM, but on GENIA, MEMM is outperformed by CMM as well.

4.3. Conditional Random Fields

Conditional random fields (CRF) (Lafferty et al., 2001) have been designed to resolve some of the shortcomings of MEMMs. The main difference lies in the number of probabilistic models used for estimating the conditional probability of a sequence labelling: MEMMs use a separate probabilistic model for each state, whereas CRFs have a single model for estimating the likelihood of an entire label sequence. The use of a single probabilistic model leads to a more realistic distribution of the probability mass among the alternative paths. As a result, CRFs tend to be less biased towards states with few successor states than CMMs and MEMMs.

On both sample tasks, CRF attains the highest scores of all three probabilistic methods tested; the last column in Table 3 shows the scores. Compared with MBL, CRF performs considerably worse on MED, but this order is reversed on GENIA, where CRF attains the best score.

5. Conclusion

Classifiers trained on entity chunking tasks that make isolated, near-sighted decisions on output symbols and that do not optimise the resulting output sequences afterwards or internally through a feedback loop, tend to produce weaker models for sequence processing tasks than classifiers that do. The two entity chunking tasks investigated in this paper are challenging tasks; not only because they demand the classifier to be able to segment and label variable-width chunks while obeying the syntax of the chunk analysis, but also because positive examples of labelled chunks are scattered sparsely in the data.

Following Van den Bosch and Daelemans (2005), this paper used a method based on predicting overlapping class trigrams to boost the performance of an MBL classifier on the two entity chunking tasks. Unlike the original work on class trigrams, however, the current study replaced the handcrafted post-processing rules by post-processing procedures learned automatically from labelled example data.

In a series of experiments, the two automatically learned post-processing procedures have been compared with a baseline unigram class predicting classifier, and with a class trigram predicting classifier using the original handcrafted post-processing rule. A meta-learner that simply tries to map the three overlapping class trigrams to a single class unigram was able to improve upon the baseline performance, but in comparison with the handcrafted post-processing rule, its performance was considerably worse.

The other meta-learned post-processing procedure used a description language that was more fine-grained, allowing the rules to refer to component symbols of the trigrams, as well as matches between them. With this representation language, a performance is attained that matches that of the handcrafted post-processing rule. Further analysis of the learned combination rules points out that they implement a voting procedure in quite the same way as the handcrafted

post-processing rule does. However, both methods do differ in the way ties are dealt with. The handcrafted rule bases its decision on classifier confidence, whereas the learned procedure contains separate tie-breaking rules for each different class. Overall, the results show that predicting class trigrams is a useful method to improve upon a baseline classifier predicting unigram classes, and that majority voting is sound method for combining the overlapping class trigrams produced by the base classifier.

In order to evaluate the class trigram method with more competitive reference scores, a number of probabilistic sequence labelling methods have been evaluated on the same entity chunking tasks. On the two benchmark sets, the class trigram method compares rather favourably with the probabilistic methods: on MED, it outperforms all three probabilistic methods by large margin; on GENIA, conditional random fields are the best performing method, MBL with class trigrams ending second, with a performance that is roughly similar to that of the conditional markov model.

These findings suggest that not only is the class trigram method able to improve upon a baseline classifier predicting only unigram classes, an optimised MBL classifier predicting class trigrams followed by a learned combination procedure also performs rather similarly to state-of-the-art probabilistic sequence labelling techniques.

Acknowledgements

The work of the first author is funded by the Netherlands Organisation for Scientific Research (NWO) as part of the NWO IMIX Programme.

References

Cohen, W. (1995). Fast effective rule induction. *Proceedings 12th International Conference on Machine Learning* (pp. 115–123). Morgan Kaufmann.

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

Daelemans, W., Zavrel, J., Berck, P., & Gillis, S. (1996). MBT: A memory-based part of speech tagger generator. *Proceedings of Fourth Workshop on Very Large Corpora* (pp. 14–27).

Daelemans, W., Zavrel, J., Van der Sloot, K., & Van den Bosch, A. (2004). *TiMBL: Tilburg memory based learner, version 5.1.0, reference guide* (Technical Report ILK 04-02). ILK Research Group, Tilburg University.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the 18th International Conference on Machine Learning*. Williamstown, MA.

McCallum, A., Freitag, D., & Pereira, F. (2000). Maximum entropy Markov models for information extraction and segmentation. *Proceedings of the 17th International Conference on Machine Learning*. Stanford, CA.

McCallum, A. K. (2002). Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Noreen, E. (1989). *Computer-intensive methods for testing hypotheses: an introduction*. John Wiley and sons.

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

Ratnaparkhi, A. (1996). A maximum entropy part-of-speech tagger. *Proceedings of the Conference on Empirical Methods in Natural Language Processing, May 17-18, 1996, University of Pennsylvania*.

Tateisi, Y., Mima, H., Tomoko, O., & Tsujii, J. (2002). Genia corpus: an annotated research abstract corpus in molecular biology domain. *Human Language Technology Conference (HLT 2002)* (pp. 73–77).

Van den Bosch, A. (2004). Wrapped progressive sampling search for optimizing learning algorithm parameters. *Proceedings of the 16th Belgian-Dutch Conference on Artificial Intelligence* (pp. 219–226). Groningen, The Netherlands.

Van den Bosch, A., & Daelemans, W. (2005). Improving sequence segmentation learning by predicting trigrams. *Proceedings of the Ninth Conference on Computational Natural Language Learning*. To appear.

Wolpert, D. H. (1992). Stacked Generalization. *Neural Networks, 5*, 241–259.