Constraint Satisfaction Inference: Non-probabilistic Global Inference for Sequence Labelling

Sander Canisius and **Antal van den Bosch**
ILK / Language and Information Science
Tilburg University
Tilburg, The Netherlands
{S.V.M.Canisius,Antal.vdnBosch}@uvt.nl

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

Abstract

We present a new method for performing sequence labelling based on the idea of using a machine-learning classifier to generate several possible output sequences, and then applying an inference procedure to select the best sequence among those. Most sequence labelling methods following a similar approach require the base classifier to make probabilistic predictions. In contrast, our method can be used with virtually any type of classifier. This is illustrated by implementing a sequence classifier on top of a (non-probabilistic) memory-based learner. In a series of experiments, this method is shown to outperform two other methods; one naive baseline approach, and another more sophisticated method.

1 Introduction

In machine learning for natural language processing, many diverse tasks somehow involve processing of sequentially-structured data. For example, syntactic chunking, grapheme-to-phoneme conversion, and named-entity recognition are all usually reformulated as sequence labelling tasks: a task-specific global unit, such as a sentence or a word, is divided into atomic sub-parts, e.g. word or letters, each of which is separately assigned a label. The concatenation of those labels forms the eventual output for the global unit.

More formally, we can define a sequence labelling task as a tuple $(\mathbf{x}, \mathbf{y}, \ell)$. The goal is to map an input vector $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$ of tokens to an output sequence $\mathbf{y} = \langle y_1, y_2, \ldots, y_n \rangle$ of labels. The possible labels for each token are specified by a finite set ℓ, that is, $y_i \in \ell, \forall i$.

In most real-world sequence labelling tasks, the values of the output labels are sequentially correlated. For machine learning approaches to sequence labelling this implies that classifying each token separately without considering the labels assigned to other tokens in the sequence may lead to sub-optimal performance. Ideally, the complex mapping of the entire input sequence to its corresponding output sequence is considered one classification case; the classifier then has access to all information stored in the sequence. In practise, however, both input and output sequences are far too sparse for such classifications to be performed reliably.

A popular approach to circumvent the issues raised above is what we will refer to as the classification and inference approach, covering techniques such as hidden markov models and conditional random fields (Lafferty et al., 2001). Rather than having a token-level classifier make local decisions independently of the rest of the sequence, the approach introduces an inference procedure, operating on the level of the sequence, using class likelihoods estimated by the classifier to optimise the likelihood of the entire output sequence.

A crucial property of most of the classification and inference techniques in use today is that the classifier used at the token level must be able to estimate the likelihood for each potential class label. This is in contrast with the more common view of a classifier having to predict just one class label for an instance which is deemed most optimal. Maximum-entropy models, which are used in many classification and inference techniques, have this property; they model the conditional class distribution. In general, this is the case for all probabilistic classification methods. However, many general-purpose machine learning techniques are not probabilistic. In order to design inference procedures for those techniques, other principles than probabilistic ones have to be used.

In this paper, we propose a non-probabilistic inference procedure that improves performance of a memory-based learner on a wide range of natural-language sequence processing tasks. We start from a technique introduced recently by Van den Bosch and Daelemans (2005), and reinterpret it as an instance of the classification and inference approach. Moreover, the token-level inference procedure proposed in the original work is replaced by a new procedure based on principles of constraint satisfaction that does take into account the entire sequential context.

The remainder of this paper is structured as follows. Section 2 introduces the theoretical background and starting point of the work presented in this paper: the trigram method, and memory-based learning. Next, the new constraint-satisfaction-based inference procedure for class trigrams is presented in Section 3. Experimental comparisons of a non-sequence-aware baseline classifier, the original trigram method, and the new classification and inference approach on a number of sequence labelling tasks are presented in Section 4 and discussed in Section 5. Finally, our work is compared and contrasted with some related approaches in Section 6, and conclusions are drawn in Section 7.

2 Theoretical background

2.1 Class Trigrams

A central weakness of approaches considering each token of a sequence as a separate classification case is their inability to coordinate labels assigned to neighbouring tokens. Due to this, invalid label sequences, or ones that are highly unlikely may result. Van den Bosch and Daelemans (2005) propose to resolve parts of this issue by predicting trigrams of labels as a single atomic class label, thereby labelling three tokens at once, rather than classifying each token separately. Predicting sequences of three labels at once makes sure that at least these short subsequences are known to be syntactically valid sequences according to the training data.

Applying this general idea, Van den Bosch and Daelemans (2005) label each token with a complex class label composed of the labels for the preceding token, the token itself, and the one following it in the sequence. If such class trigrams are assigned to all tokens in a sequence, the actual label for each of those is effectively predicted three times, since every token but the first and last is covered by three class trigrams. Exploiting this redundancy, a token's possibly conflicting predictions are resolved by voting over them. If two out of three trigrams suggest the same label, this label is selected; in case of three different candidate labels, a classifier-specific confidence metric is used to break the tie.

Voting over class trigrams is but one possible approach to taking advantage of the redundancy obtained with predicting overlapping trigrams. A disadvantage of voting is that it discards one of the main benefits of the class trigram method: predicted class trigrams are guaranteed to be syntactically correct according to the training data. The voting technique splits up the predicted trigrams, and only refers to their unigram components when deciding on the output label for a token; no attempt is made to keep the trigram sequence intact in the final output sequence. The alternative to voting presented later in this paper does try to retain predicted trigrams as part of the output sequence.

2.2 Memory-based learning

The name memory-based learning refers to a class of methods based on the k-nearest neighbour rule. At training time, all example instances are stored in memory without attempting to induce an abstract representation of the concept to be learned. Generalisation is postponed until a test instance is classified. For a given test instance, the class predicted is the one observed most frequently among a number of most-similar instances in the instance base. By only generalising when confronted with the instance to be classified, a memory-based learner behaves as a local model, specifically suited for that part of the instance space that the test instance belongs to. In contrast, learners that abstract at training time can only generalise globally. This distinguishing property makes memory-based learners especially suited for tasks where different parts of the instance space are structured according to different rules, as is often the case in natural-language processing.

For the experiments performed in this study we used the memory-based classifier as implemented by TiMBL (Daelemans et al., 2004). In TiMBL, similarity is defined by two parameters: a feature-level similarity metric, which assigns a real-valued score to pairs of values for a given feature, and a set of feature weights, that express the importance of the various features for determining the similarity of two instances. Further details on both of these parameters can be found in the TiMBL manual. To facilitate the explanation of our inference procedure in Section 3, we will formally define some notions related to memory-based classification.

The function $N_{s,w,k}(x)$ maps a given instance x to the set of its nearest neighbours; here, the parameters s, w, and k are the similarity metric, the feature weights, and the number k of nearest neighbours, respectively. They will be considered given in the following, so we will refer to this specific instantiation simply as $N(x)$. The function $w_d(c, N(x))$ returns the weight assigned to class c in the given neighbourhood according to the distance metric d; again we will use the notation $w(c, N(x))$ to refer to a specific instantiation of this function. Using these two functions, we can formulate the nearest neighbour rule as follows.

$$\arg\max_{c} w(c, N(x))$$

The class c maximising the above expression is returned as the predicted class for the instance x.

3 Constraint Satisfaction Inference

A strength of the class trigram method is the guarantee that any trigram that is predicted by the base classifier represents a syntactically valid subsequence of length three. This does not necessarily mean the trigram is a correct label assignment within the context of the current classification, but it does reflect the fact that the trigram has been observed in the training data, and, moreover, is deemed most likely according to the base classifier's model. For this reason, it makes sense to try to retain predicted trigrams in the output label sequence as much as possible.

The inference method proposed in this section seeks to attain this goal by formulating the class trigram disambiguation task as a weighted constraint satisfaction problem (W-CSP). Constraint satisfaction is a well-studied research area with applications in numerous fields both inside and outside of computer science. Weighted constraint satisfaction extends the traditional constraint satisfaction framework with soft constraints; such constraints are not required to be satisfied for a solution to be valid, but constraints a given solution does satisfy, are rewarded according to weights assigned to them.

Formally, a W-CSP is a tuple (X, D, C, W). Here, $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of variables. $D(x)$ is a function that maps each variable to its domain, that is, the set of values that variable can take on. C is the set of constraints. While a variable's domain dictates the values a single variable is allowed to take on, a constraint specifies which simultaneous value *combinations* over a number of variables are allowed. For a traditional (non-weighted) constraint satisfaction problem, a valid solution would be an assignment of values to the variables that (1) are a member of the corresponding variable's domain, and (2) satisfy *all* constraints in the set C. Weighted constraint satisfaction, however, relaxes this requirement to satisfy all constraints. Instead, constraints are assigned weights that may be interpreted as reflecting the importance of satisfying that constraint.

Let a constraint $c \in C$ be defined as a function that maps each variable assignment to 1 if the constraint is satisfied, or to 0 if it is not. In addition, let $W \colon C \rightarrow \mathbb{R}^+$ denote a function that maps each constraint to a positive real value, reflecting the weight of that constraint. Then, the optimal solution to a W-CSP is given by the following equation.

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \sum_{c} W(c)c(\mathbf{x})$$

That is, the assignment of values to its variables that maximises the sum of weights of the constraints that have been satisfied.

Translating the terminology introduced earlier in this paper to the constraint satisfaction domain, each token of a sequence maps to a variable, the domain of which corresponds to the three candidate labels for this token suggested by the trigrams covering the token. This provides us with a definition of the function D, mapping variables to their domain. In the following, $y_{i,j}$ denotes the candidate label for token x_j predicted by the trigram assigned to token x_i.

$$D(x_i) = \{y_{i-1,i}, y_{i,i}, y_{i+1,i}\}$$

Constraints are extracted from the predicted trigrams. Given the goal of retaining predicted trigrams in the output label sequence as much as possible, the most important constraints are simply the trigrams themselves. A predicted trigram describes a subsequence of length three of the entire output sequence; by turning such a trigram into a constraint, we express the wish to have this trigram end up in the final output sequence.

$$(x_{i-1}, x_i, x_{i+1}) = (y_{i,i-1}, y_{i,i}, y_{i,i+1}), \forall i$$

No base classifier is flawless though, and therefore not all predicted trigrams can be expected to be correct. Nevertheless, even an incorrect trigram may carry some useful information regarding the output sequence: one trigram also covers two bigrams, and three unigrams. An incorrect trigram may still contain smaller subsequences, of length one or two, that are correct. Therefore, all of these are also mapped to constraints.

$$\begin{aligned} (x_{i-1}, x_i) &= (y_{i,i-1}, y_{i,i}), & \forall i \\ (x_i, x_{i+1}) &= (y_{i,i}, y_{i,i+1}), & \forall i \end{aligned}$$

$$\begin{aligned} x_{i-1} &= y_{i,i-1}, & \forall i \\ x_i &= y_{i,i}, & \forall i \\ x_{i+1} &= y_{i,i+1}, & \forall i \end{aligned}$$

With such an amount of overlapping constraints, the satisfaction problem obtained easily becomes over-constrained, that is, no variable assignment exists that can satisfy all constraints without breaking another. Only one incorrectly predicted class trigram already leads to two conflicting candidate labels for one of the tokens at least. Yet, without conflicting candidate labels no inference would be needed to start with. The choice for the weighted constraint satisfaction method always allows a solution to be found, even in the presence of conflicting constraints. Rather than requiring all constraints to be satisfied, each constraint is assigned a certain weight; the optimal solution to the problem is an assignment of values to the variables that optimises the sum of weights of the constraints that are satisfied.

Constraints can directly be traced back to a prediction made by the base classifier. If two constraints are in conflict, the one which the classifier was most certain of should preferably be satisfied. In the W-CSP framework, this preference can be expressed by weighting constraints according to the classifier confidence for the originating trigram. For the memory-based learner, we define the classifier confidence for a predicted class c_i as the weight assigned to that class in the neighbourhood of the test instance, divided by the total weight of all classes.

$$\frac{w(c_i, N(x))}{\sum_c w(c, N(x))}$$

Let x denote a test instance, and c^* its predicted class. Constraints derived from this class are weighted according to the following rules.

- for a trigram constraint, the weight is simply the base classifier's confidence value for the class c^*
- for a bigram constraint, the weight is the sum of the confidences for all trigram classes in the nearest-neighbour set of x that assign the same label bigram to the tokens spanned by the constraint
- for a unigram constraint, the weight is the sum of the confidences for all trigram classes in the nearest-neighbour set of x that assign the same label to the token spanned by the constraint

4 Experiments

To thoroughly evaluate our new inference procedure, and to show that it performs well over a wide range of natural-language sequence labelling tasks, we composed a benchmark set consisting of six different tasks, covering four areas in natural language processing: syntax (syntactic chunking), morphology (morphological analysis), phonology (grapheme-to-phoneme conversion), and information extraction (general, medical, and biomedical named-entity recognition). Below, the six data sets used for these tasks are introduced briefly.

CHUNK is the task of splitting sentences into non-overlapping syntactic phrases or constituents. The data set, extracted from the WSJ Penn Treebank, and first used in the CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000), contains 211,727 training examples and 47,377 test instances.

NER, named-entity recognition, involves identifying and labelling named entities in text. We employ the English NER shared task data set used in the CoNLL-2003 conference (Tjong Kim Sang and De Meulder, 2003). This data set discriminates four name types: persons, organisations, locations, and a rest category of "miscellany names". The data set is a collection of newswire articles from the Reuters Corpus, RCV1[1]. The given training set contains 203,621 examples; as test set we use the "testb" evaluation set which contains 46,435 examples.

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. The data have been split into training and test sets, resulting in 428,502 training examples and 47,430 test examples.

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.

PHONEME refers to grapheme-to-phoneme conversion for English. The sequences to be labelled are words composed of letters (rather than sentences composed of words). We based ourselves on the English part of the CELEX-2 lexical data base (Baayen et al., 1993), from which we extracted 65,467 word-pronunciation pairs. This pair list has been aligned using expectation-maximisation to obtain sensible one-to-one mappings between letters and phonemes (Daelemans and Van den Bosch, 1996). The classes to predict are 58 different phonemes, including some diphones such as [ks] needed to keep the letter-phoneme alignment one-to-one. The resulting data set has been split into a training set of 515,891 examples, and a test set of 57,279 examples.

MORPHO refers to morphological analysis of Dutch words. We collected the morphological analysis of 336,698 Dutch words from the CELEX-2 lexical data base (Baayen et al., 1993), and represented the task such that it captures the three most relevant elements of a morphological analysis: (1) the segmentation of the word into morphemes (stems, derivational morphemes, and inflections), (2) the part-of-speech tagging information contained by each morpheme; and (3) all spelling changes due to compounding, derivation, or inflection that would enable the reconstruction of the appropriate root forms of the involved morphemes.

Task	Baseline	Voting	CSInf	Oracle
CHUNK	91.9	92.7	93.1	95.8
NER	77.2	80.2	81.8	86.5
MED	64.7	67.5	68.9	74.9
GENIA	55.8	60.1	61.8	70.6
PHONEME	79.0	83.4	84.5	92.2
MORPHO	41.3	46.1	51.9	62.2

Table 1: Performances of the baseline method, and the trigram method combined both with majority voting, and with constraint satisfaction inference. The last column shows the performance of the (hypothetical) oracle inference procedure.

For CHUNK, and the three information extraction tasks, instances represent a seven-token window of words and their (predicted) part-of-speech tags. Each token is labelled with a class using the IOB type of segmentation coding as introduced by Ramshaw and Marcus (1995), marking whether the middle word is inside (I), outside (O), or at the beginning (B) of a chunk, or named entity. Performance is measured by the F-score on correctly identified and labelled chunks, or named entities.

Instances for PHONEME, and MORPHO consist of a seven-letter window of letters only. The labels assigned to an instance are task-specific and have been introduced above, together with the tasks themselves. Generalisation performance is measured on the word accuracy level: if the entire phonological transcription of the word is predicted correctly, or if all three aspects of the morphological analysis are predicted correctly, the word is counted correct.

4.1 Results

For the experiments, memory-based learners were trained and automatically optimised with wrapped progressive sampling (Van den Bosch, 2004) to predict class trigrams for each of the six tasks introduced above. Table 1 lists the performances of constraint satisfaction inference, and majority voting applied to the output of the base classifiers, and compares them with the performance of a naive baseline method that treats each token as a separate classification case without coordinating decisions over multiple tokens.

Without exception, constraint satisfaction infer-

[1] Reuters Corpus, Volume 1, English language, 1996-08-20 to 1997-08-19.

ence outperforms majority voting by a considerable margin. This shows that, given the same sequence of predicted trigrams, the global constraint satisfaction inference manages better to recover sequential correlation, than majority voting. On the other hand, the error reduction attained by majority voting with respect to the baseline is in all cases more impressive than the one obtained by constraint satisfaction inference with respect to majority voting. However, it should be emphasised that, while both methods trace back their origins to the work of Van den Bosch and Daelemans (2005), constraint satisfaction inference is not applied after, but instead of majority voting. This means, that the error reduction attained by majority voting is also attained, independently by constraint satisfaction inference, but in addition constraint satisfaction inference manages to improve performance on top of that.

5 Discussion

The experiments reported upon in the previous section showed that by globally evaluating the quality of possible output sequences, the constraint satisfaction inference procedure manages to attain better results than the original majority voting approach. In this section, we attempt to further analyse the behaviour of the inference procedure. First, we will discuss the effect that the performance of the trigram-predicting base classifier has on the maximum performance attainable by any inference procedure. Next, we will consider specifically the effect of base classifier accuracy on the performance of constraint satisfaction inference.

5.1 Base classifier accuracy and inference procedure upper-bounds

After trigrams have been predicted, for each token, at most three different candidate labels remain. As a result, if the correct label is not among them, the best inference procedure cannot correct that. This suggests that there is an upper-bound on the performance attainable by inference procedures operating on less than perfect class trigram predictions. To illustrate what performance is still possible after a base classifier has predicted the trigrams for a sequence, we devised an oracle inference procedure.

An oracle has perfect knowledge about the true label of a token; therefore it is able to select this label if it is among the three candidate labels. If the correct label is absent among the candidate labels, no inference procedure can possibly predict the correct label for the corresponding token, so the oracle procedure just selects randomly among the candidate labels, which will be incorrect anyway. Table 1 compares the performance of majority voting, constraint satisfaction inference, and the oracle after an optimised base classifier has predicted class trigrams.

5.2 Base classifier accuracy and constraint satisfaction inference performance

There is a subtle balance between the quality of the trigram-predicting base classifier, and the gain that any inference procedure for trigram classes can reach. If the base classifier's predictions are perfect, all three candidate labels will agree for all tokens in the sequence; consequently the inference procedure can only choose from one potential output sequence. On the other extreme, if all three candidate labels disagree for all tokens in the sequence, the inference procedure's task is to select the best sequence among 3^n possible sequences, where n denotes the length of the sequence; it is likely that such a huge amount of candidate label sequences cannot be dealt with appropriately.

Table 2 collects the base classifier accuracies, and the average number of potential output sequences per sentence resulting from its predictions. For all tasks, the number of potential sequences is manageable; far from the theoretical maximum 3^n, even for GENIA, that, compared with the other tasks, has a relatively large number of potential output sequences. The factors that have an effect on the number of sequences are rather complex. One important factor is the accuracy of the trigram predictions made by the base classifier. To illustrate this, Figure 1 shows the number of potential output sequences as a function of the base classifier accuracy for the PHONEME task. There is an almost linear decrease of the number of possible sequences as the classifier accuracy improves. This shows that it is important to optimise the performance of the base classifier, since it decreases the number of potential output sequences to consider for the inference procedure. Other factors affecting the number of potential output sequences are the length of the sequence, and the number of labels defined for the task. Unlike classifier accuracy, however, these two factors

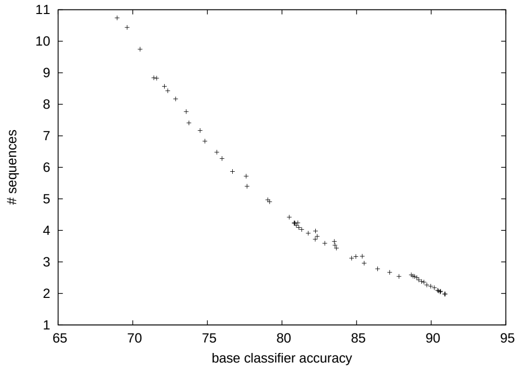

Figure 1: Average number of potential output sequences as a function of base classifier accuracy on the PHONEME task.

Task	Base acc.	Avg. # seq.
CHUNK	88.8	38.4
NER	91.6	9.0
MED	77.1	9.3
GENIA	71.8	1719.3
PHONEME	91.7	1.8
MORPHO	80.9	2.8

Table 2: The average number of potential output sequences that result from class trigram predictions made by a memory-based base classifier.

are inherent properties of the task, and cannot be optimised.

While we have shown that improved base classifier accuracy has a positive effect on the number of possible output sequences; we have not yet established a positive relation between the number of possible output sequences and the performance of constraint satisfaction inference. Figure 2 illustrates, again for the PHONEME task, that there is indeed a positive, even linear relation between the accuracy of the base classifier, and the performance attained by inference. This relation exists for all inference procedures: majority voting, as well as constraint satisfaction inference, and the oracle procedure. It is interesting to see how the curves for those three procedure compare with each other.

The oracle always outperforms the other two procedures by a wide margin. However, its increase is less steep. Constraint satisfaction inference consistently outperforms majority voting, though the difference between the two decreases as the base classifier's predictions improve. This is to be expected, since more accurate predictions means more majorities will appear among candidate labels, and the predictive quality of such majorities improves as well. In the limit –with a perfect base classifier– all three curves will meet.

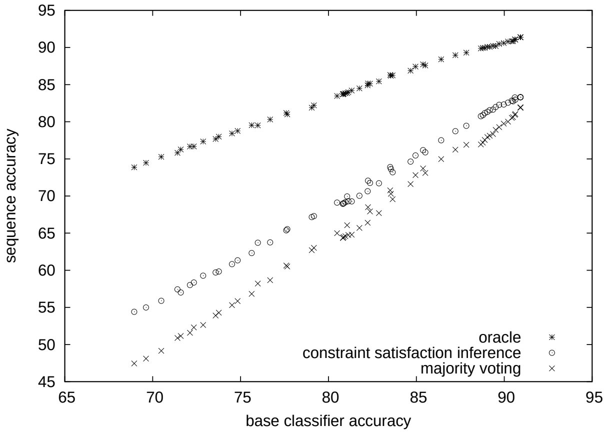

Figure 2: Performance of majority voting, constraint satisfaction inference, and the oracle inference procedure as a function of base classifier accuracy on the PHONEME task.

6 Related work

Many learning techniques specifically designed for sequentially-structured data exist. Given our goal of developing a method usable with non-probabilistic classifiers, we will not discuss the obvious differences with the many probabilistic methods. In this section, we will contrast our work with two other approaches that also apply principles of constraint satisfaction to sequentially-structured data.

Constraint Satisfaction with Classifiers (Punyakanok and Roth, 2001) performs the somewhat more specific task of identifying phrases in a sequence. Like our method, the task of coordinating local classifier decisions is formulated as a constraint satisfaction problem. The variables encode whether or not a certain contiguous span of tokens forms a phrase. Hard constraints enforce that no two phrases in a solution overlap.

Similarly to our method, classifier confidence estimates are used to rank solutions in order of preference. Unlike in our method, however, both the domains of the variables and the constraints are prespecified; the classifier is used only to estimate the cost of potential variable assignments. In our approach, the classifier predicts the domains of the variables, the constraints, and the weights of those.

Roth and Yih (2005) replace the Viterbi algorithm for inference in conditional random fields with an integer linear programming formulation. This allows arbitrary global constraints to be incorporated in the inference procedure. Essentially, the method adds constraint satisfaction functionality on top of the inference procedure. In our method, constraint satisfaction *is* the inference procedure. Nevertheless, arbitrary global constraints (both hard and soft) can easily be incorporated in our framework as well.

7 Conclusion

The classification and inference approach is a popular and effective framework for performing sequence labelling in tasks where there is strong interaction between output labels. Most existing inference procedures expect a base classifier that makes probabilistic predictions, that is, rather than predicting a single class label, a conditional probability distribution over the possible classes is computed. The inference procedure presented in this paper is different in the sense that it can be used with any classifier that is able to estimate a confidence score for its (non-probabilistic) predictions.

Constraint satisfaction inference builds upon the class trigram method introduced by Van den Bosch and Daelemans (2005), but reinterprets it as a strategy for generating multiple potential output sequences, from which it selects the sequence that has been found to be most optimal according to a weighted constraint satisfaction formulation of the inference process. In a series of experiments involving six sequence labelling task covering several different areas in natural language processing, constraint satisfaction inference has been shown to improve substantially upon the performance achieved by a simpler inference procedure based on majority voting, proposed in the original work on the class trigram method.

The work presented in this paper shows there is potential for alternative interpretations of the classification and inference framework that do not rely on probabilistic base classifiers. Future work may well be able to further improve the performance of constraint satisfaction inference, for example, by using more optimised constraint weighting schemes. In addition, alternative ways of formulating constraint satisfaction problems from classifier predictions may be explored; not only for sequence labelling, but also for other domains that could benefit from global inference.

References

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

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

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

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

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems*, pages 995–1001. The MIT Press.

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

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fields. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 737–744.

Yuka Tateisi, Hideki Mima, Ohta Tomoko, and Junichi Tsujii. 2002. Genia corpus: an annotated research abstract corpus in molecular biology domain. In *Human Language Technology Conference (HLT 2002)*, pages 73–77.

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

E. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

A. Van den Bosch and W. Daelemans. 2005. Improving sequence segmentation learning by predicting trigrams. In I. Dagan and D. Gildea, editors, *Proceedings of the Ninth Conference on Computational Natural Language Learning*.

A. Van den Bosch. 2004. Wrapped progressive sampling search for optimizing learning algorithm parameters. In R. Verbrugge, N. Taatgen, and L. Schomaker, editors, *Proceedings of the 16th Belgian-Dutch Conference on Artificial Intelligence*, pages 219–226, Groningen, The Netherlands.