Constraint-Satisfaction Inference for Entity Recognition

Sander Canisius, Antal van den Bosch, and Walter Daelemans

Abstract One approach to QA answering is to match a question to candidate answers in a background corpus based on semantic overlap, possibly in combination with other levels of matching, such as lexical vector space similarity and syntactic similarity. While the computation of deep semantic similarity is as yet generally infeasible, semantic analysis in a specific domain is feasible, if the analysis is constrained to finding domain-specific entities and basic relations. Finding domain-specific entities, the focus of this chapter, is still not a trivial task due to ambiguities of terms. This problem, like many others in Natural Language Processing, is a sequence labelling task. We describe the development of a new approach to sequence labelling in general, based on the constraint satisfaction inference. The output of the machine-learning-based classifiers that solve aspects of the task (such as subsequently predicting the output of the label sequence) are considered as constraints on the global structured output analysis. The constraint-satisfaction inference method is compared to other state-of-the-art sequence labelling approaches, showing competitive performance.

1 Introduction

The Dutch word *arm* has two main senses, which in English are translated into *arm*, the body part, and *poor*. In the medical domain covered by the IMIX project, the *arm*

Sander Canisius
Netherlands Cancer Institute, Amsterdam, The Netherlands, e-mail: s.canisius@nki.nl

Antal van den Bosch
Tilburg center for Cognition and Communication, Tilburg University, Tilburg, The Netherlands, e-mail: Antal.vdnBosch@uvt.nl

Walter Daelemans
Computational Linguistics and Psycholinguistics Research Centre, University of Antwerp, Antwerp, Belgium, e-mail: Walter.Daelemans@ua.ac.be

meaning is obviously relevant: it is important for further processing, such as QA, to be able to detect when this sense is being used. The *poor* sense may be relevant to the domain as well, as it may be part of medical phrases such as *vitamin-poor diet*. Alternatively, it may be used in a non-medical sense in a medical text when it refers to the economic sense of *poor*. To summarise: detecting medically relevant concepts is not the mere detection of words from a gazetteer list (pre-collected lists of diseases, treatments, etc.), but rather a task that borrows some complexity from word sense disambiguation. Moreover, as many medical concepts are expressed in multiword expressions, such as *high fever* or *very low density lipoprotein*, there is also a challenge in finding the correct beginning and end of each expression.

Assuming that this aspect of automation requires a high-precision, high-recall solution to be used for the higher IMIX goal of medical question answering, we first define medical entity recognition as the focus problem, aiming to solve it with state-of-the-art natural language sequence processing methods. This chapter documents this particular effort. First, a limited set of thirteen entity types were identified that were relevant to the medical domain. Then, the IMIX medical encyclopaedia background corpus was annotated with these entities. A generic entity recognition method was developed, a hybrid of memory-based classification and constraint satisfaction inference, which was also applied to related benchmark problems in entity recognition as well as in purely syntactic chunking. The constraint satisfaction inference method is shown to attain state-of-the-art performance. However, entity recognition scores are not excellent. In this volume, Bouma, Fahmi, and Mur, in their chapter *Relation Extraction for Open and Closed Domain Question Answering*, further investigate the issue of how to employ the results of automatic entity recognition in QA.

The remainder of this chapter is structured as follows. Section 2 introduces sequence labelling as a special subclass of machine learning problems. In Section 3, a concise overview of previous machine learning approaches to sequence labelling is presented. Section 4 introduces a trigram-based sequence labelling method in which subsequences of trigram classes are predicted and simplistically resolved into output sequences. Next, Section 5 describes the constraint-satisfaction-based inference procedure. 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 Sections 6, 7 and 8, and discussed in Section 9. Conclusions are drawn in Section 10.

2 Sequence Labelling

Sequences are amongst the most versatile output structures in natural language processing. Many linguistic processing tasks can be seen as generating sequential outputs, either because the output naturally corresponds to a sequence, such as with POS-tagging, or because the targeted output structure is easily mapped to a

sequence, as is the case in, for example, named-entity recognition. This chapter focuses on a subclass of sequence prediction, referred to as sequence labelling.

In a sequence labelling task, both inputs and outputs are sequences. Typically, the aim is not simply to classify the complete input sequence according to some global property, but rather to recover some type of hidden structure, closely linked to the elements of the input sequence. Sequence labelling abstractly defines this hidden structure as a sequence of label assignments to each of the elements of the input sequence. Thus, the output sequence—also referred to as the label sequence—has the same length as the input sequence, and there is a one-to-one correspondence between the elements of both sequences in the sense that the *i*th element of the output sequence is the label of the *i*th element of the input. Labels that make up such label sequences are taken from a restricted label set, and only have significance for the target application. In the context of sequence labelling, they are treated simply as atomic symbols. In this respect, a naive interpretation of sequence labelling would simply rephrase it as a sequence of multiclass classification cases. However, as in any structured prediction task, it is assumed that dependencies amongst different elements of the output sequence are as important as dependencies between an element of the input and its label in the output. In this chapter two *structured prediction* techniques are explored, one simple and one more complex. These techniques attempt to model dependencies between input and output elements as well as amongst elements of the output sequence.

3 Related Work

Because of the wide applicability of sequence labelling as a processing task template, it is unsurprising that it has received considerable attention in machine learning. Many techniques for learning to predict complex output structures were originally developed in the context of sequence labelling. This is the case, for example, for *structured linear models*, which in recent years have been the most popular framework for sequence labelling. There have been several different implementations of linear models for sequence labelling (Lafferty et al, 2001; Collins, 2002; Altun et al, 2003), which are based on the same graphical model formalism, but differ in the learning algorithm used for parameter estimation. The underlying graphical model encodes the independence assumption, which states that a certain label in the output sequence only depends on the input and a fixed number of labels directly preceding it in the output. This number is referred to as the Markov order of the model, and most commonly equals one, although second-order models have been used as well (e.g. Sha and Pereira, 2003). Because of this assumption, efficient inference is possible for such linear models. Most notably, the Viterbi algorithm finds the optimal output sequence according to a linear model in $O(L^2 n)$ time, where L is the number of labels, and n is the length of the output sequence. On the negative side, features on output elements can only cover the small number

of preceding labels permitted by the Markov order, and consequently models are restricted in the types of structural dependencies that can be modelled.

A range of other machine learning methods have been applied to sequence labelling as well. Punyakanok and Roth (2001) apply the constraint satisfaction with classifiers (CSCL) framework to sequence segmentation, formulated in terms of a sequence labelling problem. Both Ratnaparkhi (1996) and McCallum et al (2000) proposed discriminative Markov-like models for sequence labelling. They have mostly been superseded by structured linear models. As a final example in this incomplete overview, an output kernel approach to sequence labelling is described by Cortes et al (2005).

4 A Baseline Approach

Many tasks in natural language processing are sequence tasks, due to the obvious sequential nature of words as sequences of phonemes or letters, and sentences and spoken utterances as sequences of words. However, many machine learning methods do not typically learn these tasks by learning to map input sequences to output sequences. Rather, the standard approach that fits any supervised classification-based machine learning algorithm is to encode a sequence processing task by windowing, in which input subsequences are mapped to single output symbols. A single output symbol is typically associated with one of the input symbols, for example the middle one in the window.

Figure 1 displays this simplest version of the windowing process; fixed-width subsequences of input symbols are coupled to one output symbol. To ignore that the

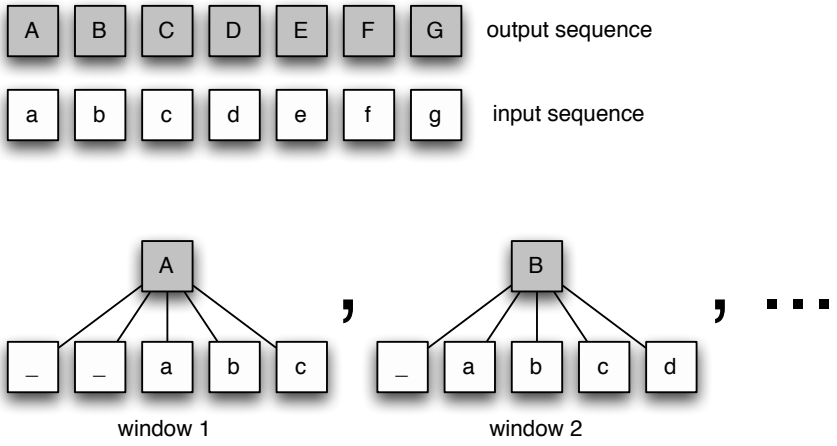

Fig. 1 Standard windowing process. Sequences of input symbols and output symbols are converted into windows of fixed-width input symbols each associated with one output symbol.

output forms a sequence is a problematic restriction, since it allows the classifier to produce invalid or impossible output sequences: for instance, it can make two neighbouring classifications in a sequence that are incompatible with each other, since it has no information about the other decision.

4.1 Class Trigrams

The restriction that classifications produce single output symbols is not intrinsic – the task may well be rephrased so that each input window is mapped to a sequence of output symbols. This directly prevents the classifier from predicting an invalid output sequence, since it will always produce sequences it has learned from training material. Van den Bosch and Daelemans (2006) propose to predict trigrams of labels (i.e. n-grams with $n = 3$) as a single atomic class label, thereby labelling three tokens at once.

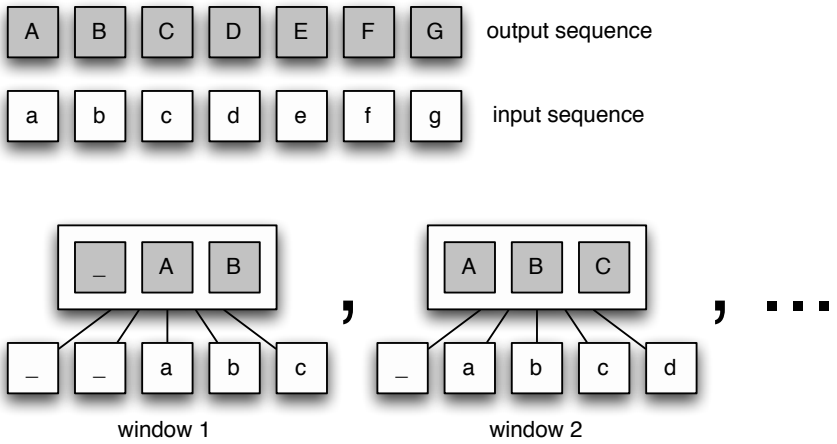

Fig. 2 Windowing process with n-grams of class symbols. Sequences of input symbols and output symbols are converted into windows of fixed-width input symbols each associated with, in this example, trigrams of output symbols.

Applying this general idea, Van den Bosch and Daelemans (2006) 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 chapter does attempt to retain predicted trigrams as part of the output sequence.

4.2 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 from their training data 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 the memory-based classifier was used as implemented in TiMBL[1] (Daelemans et al, 2009). 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. To facilitate the explanation of the inference procedure in Section 5, this chapter 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 are considered as given in the following example, and this specific instantiation will therefore be referred to 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 the notation $w(c,N(x))$ is used to refer to a specific instantiation of this function. Using these two functions, the nearest neighbour rule can be formulated as follows.

[1] `http://ilk.uvt.nl/timbl`

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

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

5 Constraint Satisfaction Inference

One disadvantage of the voting method explained in Section 4.1 is that it ignores the fact that predicted class trigrams are guaranteed to be syntactically correct according to the training data. It is also blind to the overall quality of the output sequence it generates, as the voting is a local process. Both deficiencies are aimed to be repaired by adding constraint satisfaction inference as a global procedure for producing sequential output.

Constraints over an output space of label sequences are defined as where the constraints model relevant global dependencies in the predicted label sequence in order to apply constraint satisfaction inference to sequence labelling. In this case, these dependencies are implicitly encoded by the predicted trigrams. 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 as much as possible predicted trigrams in the output label sequence.

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 which satisfy a given solution, 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 the 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 $W : C \rightarrow \mathbb{R}^+$ 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.

$$x^* = \arg\max_x \sum_c W(c)c(x)$$

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 chapter 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.

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

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; turning such a trigram into a constraint is done with the intention of having 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. This means 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 optimise the sum of the weights of the constraints that are satisfied.

Constraints can be directly 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, the confidence of the classifier is for a predicted class c_i is defined 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 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.

5.1 Solving the CSP

The output space of sequence labelling tasks is exponential in the length of the input sequence. Since the constraint satisfaction problem is constructed to only consider solutions that can be built from predictions made by the base classifiers, the output space searched by the constraint solver will typically be substantially smaller than this full output space. In spite of that, though, the worst-case complexity of this solution space remains exponential, be it with a smaller base. The constraint solver in the sequence labelling approach has to search this space.

This issue is, however, not unique to constraint satisfaction inference. It is faced by any inference-based sequence labelling approach. This is why solutions for inference for sequence labelling already exist. A popular approach is to use the Viterbi algorithm. Under the Markov assumption, finding the optimal solution is guaranteed. On the downside, the Markov assumption restricts the dependencies that are modelled. As a less restrictive alternative to the Viterbi algorithm, approximate search algorithms have been employed, such as beam search (Ratnaparkhi, 1996), or simulated annealing (Finkel et al, 2005). Viterbi is by far the most popular inference algorithm, and it could also serve as the basis of the constraint solver. However, the method allows for modelling unrestricted dependencies, not just the type of dependencies meeting the Markov assumption. This choice leaves only exhaustive or approximate search as possibilities. For the experiments in this chapter, an exhaustive search has been chosen. Even though, in the worst case, this leads to exponential-time inference, the search space resulting from the CSP formulation is assumed to be sufficiently small to make exhaustive search feasible.

6 Sequence Labelling Tasks

As mentioned earlier, natural language processing offers a wide array of tasks that can be seen as instances of sequence labelling. Input sequences may be words, sentences, or even documents. The output sequences correspond to a direct one-to-one labelling of the elements of the input sequence, or may for example encode a segmentation of the input sequence. While sequence segmentation tasks may be performed with special-purpose approaches (Carreras, 2005; Sarawagi and Cohen, 2005; Daumé III, 2006), they are most often reformulated as per-token sequence labelling tasks using the encoding scheme proposed by Ramshaw and Marcus (1995), generally referred to as the `IOB` encoding. According to this encoding, symbols assigned to input tokens signal whether the token is inside a segment (`I`), outside a segment (`O`), or at the start of a segment that was preceded by a segment of the same type ("between", `B`). A variant of `IOB` sometimes referred to as `BIO` uses the `B` to mark all tokens at the beginning of segments. Both `IOB` and `BIO` are used in the experiments reported below. Segment type labels are appended to this symbol to denote the current type of segment; `B-PP` which would mark the beginning of a prepositional phrase segment in syntactic chunking.

To illustrate the applicability of constraint satisfaction inference for a range of tasks, the results on three processing tasks are presented that can all be approached as sequence labelling tasks. All three tasks take the sentence as their domain, operating on sequences of word tokens; one is a syntactic task, and the remaining two are named entity recognition tasks. One of these entity recognition tasks is of direct interest to the IMIX project, namely the identification of medical entities in Dutch medical encyclopaedic text. The three tasks are introduced briefly in the following subsections.

6.1 Syntactic Chunking

In syntactic chunking, sometimes referred to as shallow parsing, the goal is to divide an input sentence into non-recursive syntactic base phrases, or chunks. Each base phrase is centred around some head word, which gives rise to the syntactic type of the phrase, and in addition includes some of the modifying words of the head. For example, a noun phrase consists of a head noun, and may also include some adjectives and determiners directly preceding, and syntactically modifying, the head word. As part of the chunking task, the exact boundaries of each chunk have to be determined, and in addition each chunk has to be labelled with its syntactic type. Figure 3 shows an example of a sentence and the base phrases that are to be recognised as part of the syntactic chunking task.

[$_{NP}$ Rockwell International Corp. **] [$_{NP}$** 's Tulsa unit **] [$_{VP}$** said **] [$_{NP}$** it **] [$_{VP}$** signed **] [$_{NP}$** a tentative agreement **] [$_{VP}$** extending **] [$_{NP}$** its contract **] [$_{PP}$** with **] [$_{NP}$** Boeing Co. **] [$_{VP}$** to provide **] [$_{NP}$** structural parts **] [$_{PP}$** for **] [$_{NP}$** Boeing **] [$_{NP}$** 's 747 jetliners **]** .

Fig. 3 Example sentence from the syntactic chunking task.

The standard benchmark for syntactic chunking is the data set created for the CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000). For this data set, chunks have been extracted from the full syntactic annotation of the Wall Street Journal part of the Penn Treebank (Marcus et al, 1993) and encoded in the aforementioned `BIO` notation. Besides the words and the chunks for each sentence, POS tags for those words have been obtained by running the POS tagger by Brill (1994) on the data. Using these predicted POS tags instead of the manually assigned tags available in the original corpus makes for a more realistic scenario, where the chunker has to cope with tagging errors in its input. The training data of the CoNLL-2000 data set corresponds to sections 15 to 18 of the WSJ corpus, while section 20 serves as test data. In addition, section 21 is frequently used as a development set, mainly for tuning learning algorithm parameters. This same train-test-development split has been used for the experiments reported in this chapter.

The features used for this task are all fairly standard. In a window of five tokens centred around the focus token, there are features for the word form, POS tag, and

a symbol encoding certain orthographical features of the word. In addition, in a window of three tokens centred around the focus token, conjunctions of pairs of consecutive words are included, and the same for their POS tags. Finally, again in a three-token window, conjunctions of word forms and their POS tag are encoded as features. Figure 4 illustrates these features in the context of an example sentence.

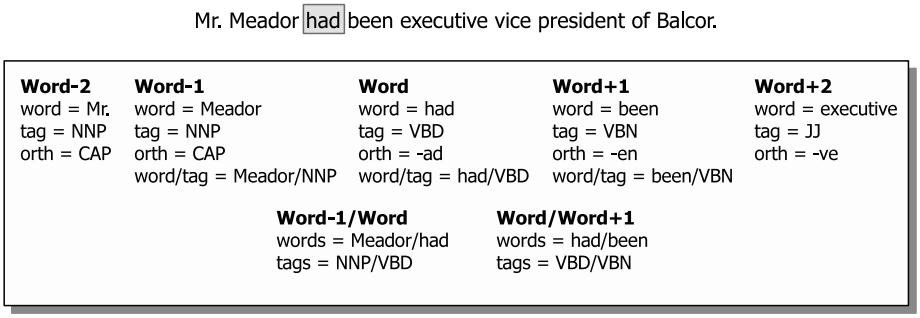

Fig. 4 Feature representation for the word "had" in the above sentence as used with the syntactic chunking task.

6.2 Named-Entity Recognition

An important subtask of information extraction is identifying names in running text, and characterising the type of real-world entity they refer to. Named-entity recognition, as it is called, has been the subject of several organised evaluations, such as MUC (Chinchor, 1995), CoNLL (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003), BioCreative (Hirschman et al, 2005), and ACE (Doddington et al, 2004). Each of these evaluations provided annotated data sets for several types of entities, and participating systems had to learn how to recognise them. Interestingly, the notion of a named entity can be defined as broadly or narrowly, as is suited for the task at hand. For example, in broadcast news texts, persons and organisations are relevant entities to discover. In contrast, those entities may not be worthwhile at all in biomedical texts, where references to proteins and viruses are more likely to be of interest.

The named-entity recognition task considered in this chapter has been defined as part of the CoNLL-2003 shared task (Tjong Kim Sang and De Meulder, 2003), from which the English data is used. Texts from this data set have been collected from the Reuters corpus (Lewis et al, 2004), which consists of general news stories, and have been annotated for named-entity mentions using the aforementioned `IOB` notation. The named entities to be recognised have been divided in four classes: people, locations, organisations, and a rather broad miscellaneous category, including such entity types as languages, events, and book titles. The task involves both identifying

the exact boundaries of each named-entity mention and assigning the correct entity type. Following the standard partitioning of the shared task data, the "test A" subset for development purposes was used, and the "test B" subset for the final evaluation.

[$_{\mathbf{ORG}}$ Interior] Minister [$_{\mathbf{PER}}$ Zbigniew Siemiatkowski] and [$_{\mathbf{PER}}$ Bernd Schmidbauer] , [$_{\mathbf{MISC}}$ German] intelligence co-ordinator in [$_{\mathbf{PER}}$ Helmut Kohl] 's chancellery, sealed the closer links during talks in [$_{\mathbf{LOC}}$ Warsaw] .

Fig. 5 Example sentence from the named-entity recognition task.

It is insightful to compare the named-entity recognition and syntactic chunking tasks. Both are sentence-level segmentation and labelling tasks. However, whereas in syntactic chunking, tokens that are not part of a chunk are exceptions, in named-entity recognition, the vast majority of tokens will not be part of a named-entity segment. For this reason, sequential correlation within label sequences can be expected to be different. The fact that a noun phrase is likely to be followed by a verb phrase can be a valuable clue to a learner. In contrast, observing that a named-entity is often followed by tokens that do not refer to an entity is almost stating the obvious. Nevertheless, sequential correlation is still an important factor in named-entity recognition as well. It may help in deciding that the two-token phrase *George Washington* is more likely to be a single `Person` entity, than a `Person` entity followed by a `Location`.

The feature set used for named-entity recognition includes features similar to those used for syntactic chunking, as displayed in Figure 4. It is extended with some additional features. First, features encoding affixes of lengths 2 and 3 of the word in focus are included. Second, syntactic chunk tags for the words in a three-token window are included. Third, again in a three-token window, features signalling the presence of the word in entity-specific gazetteer lists were added. The gazetteer lists used for this purpose were provided as part of the CoNLL-2003 shared task, and merely list the entities that are found in the annotated training data.

6.3 Medical Concept Chunking: The IMIX Task

The IMIX data set is yielded from a manually annotated Dutch-language medical encyclopaedia. The annotation offers labels for various medical entities, such as disease names, body parts, and treatments, forming a set of twelve entity types in total. Table 1 lists the twelve entity types and their frequency of occurrence in the encyclopaedia, as well as their average numbers of tokens. Three of the twelve entity types, *disease symptom*, *duration*, and *advice*, are typically several tokens long, which is to be expected given their more circumscriptive nature. The types *person feature* (such as gender, weight, age range), *treatment*, and *method of diagnosis* often span two tokens. The remaining six types are mostly single-token entities. The data has been split into training and test sets, resulting in 428,502 training examples (i.e.,

tokens, of which 128,182 tokens or 29.9% belong to a medical entity), and 47,430 test examples (with 14,314 or 39.8% of the tokens beloning to a medical entity).

Table 1 The twelve medical entity types in the IMIX data, their frequency of occurrence in the training data, and the average number of tokens per entity.

Entity type	Number of entities	Av. tokens per entity
Body part	21,509	1.2
Disease	16,037	1.3
Treatment	7,210	1.6
Disease symptom	6,636	2.2
Person	5,052	1.3
Bodily function	3,453	1.3
Duration	2,563	3.7
Disease feature	2,004	1.2
Microorganism	1,672	1.3
Method of diagnosis	1,447	1.5
Person feature	1,091	1.7
Advice	168	5.7

The feature set used for the domain-specific medical named-entity recognition is a simple encoding of the local context of each word. It consists of a seven-token window of words and POS tags centred on the token for which the label is predicted.

Een [$_{\textbf{disease}}$ cefaalhematoom] is een [$_{\textbf{disease}}$ bloeduitstorting] onder de [$_{\textbf{body-part}}$ hoofdhuid] . Bij [$_{\textbf{disease}}$ infantiel botulisme] kunnen in extreme gevallen [$_{\textbf{symptom}}$ ademhalingsproblemen] en [$_{\textbf{symptom}}$ algehele lusteloosheid] optreden.

Fig. 6 Example sentences from the IMIX concept chunking task.

7 Experimental Set-up

7.1 Evaluation

Arguably the purest way to evaluate sequence labelling performance is to measure the proportion of complete label sequences that are predicted correctly. This is an extremely strict measure that ignores the fact that even partly incorrect label sequences may still be usable in many applications. A more forgiving performance criterion counts the proportion of individual labels predicted correctly. As an advantage of token accuracy, label sequences that are almost, but not completely predicted correctly still contribute proportionally to the performance score. For some tasks—such as POS-tagging, which is not dealt with in this chapter—token

accuracy is the most natural evaluation measure. However, for many other tasks in natural language processing, token accuracy is rather uninformative. A typical example in this respect is named-entity recognition when approached as an `IOB` labelling task. The vast majority of tokens in the correct output have the `O` label; therefore, a classifier that always predicts `O` is likely to attain high token accuracy. However, in named-entity recognition, one is not interested in the tokens outside of named-entity segments, but only in those inside them. Token accuracy assigns too little importance to those tokens, and is therefore mostly unsuited for named-entity recognition.

The three tasks reported on in this chapter are in fact also concerned with sequence segmentation, rather than pure sequence labelling. Both sequence accuracy and token accuracy do not specifically measure the quality of the segments found. The most common metrics that do specifically measure segmentation and optional labelling quality are precision, recall, and $F_{\beta=1}$ (Van Rijsbergen, 1979). *Precision* corresponds to the proportion of predicted segments that are correct, where a segment will be considered correct if both its boundaries and its label exactly matches the true segment. *Recall* measures the proportion of true segments that have indeed been predicted correctly. $F_{\beta=1}$, finally, matches the harmonic mean of precision and recall.

7.2 Constraint Prediction

The choice for n-gram constraints still leaves the parameter n to be tuned, or at the very least, to be fixed in advance. Indeed, n can be seen as a genuine parameter of constraint satisfaction inference for sequence labelling; one that can be tuned for every new sequence labelling task that is to be performed. In this chapter, it was chosen not to do this, but instead decide on an $n = 3$ for all three tasks. Consequences of a choice of n include the following.

- The higher the value of n, the sparser the training data for the constraint predictor will be. In the extreme case, the label to be predicted matches the entire label sequence. This is generally not a viable option.
- n is also the theoretically maximal size of the micro-label domains. For a sequence of length T, this implies that the size of the worst-case output space is in the order of $O(n^T)$. Consequently, high values for n inevitably require approximate search in the output space. Gains that might potentially result from high values for n, may be lost as a result of only being able to use approximate search.
- Larger n-gram constraints also result in an increase of the number of, possibly conflicting, constraints covering a single micro-label. Although weighting the constraints by classification confidence should ensure that correct constraints are satisfied, too many incorrect constraints that conflict with the correct ones, may still overrule the latter.

For the above reasons, n is preferred not to be too high. On the other hand, choosing n too small will result in the loss of valuable structural information. Although there may certainly exist sequence labelling tasks for which 5-gram constraint satisfaction inference will be a viable option, or will even lead to better performance than trigrams, trigram constraints are chosen as the basis for all experiments in this chapter. An attractive consequence of choosing trigram constraints is that the solution space yielded by them is rather small, and therefore allows for efficient inference. Finally, as an interesting parallel with Markov-based sequence labelling approaches, trigram constraints can be seen as modelling an undirected first-order Markov assumption, where a label depends on the labels preceding and following it.

8 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 three tasks introduced above. Table 2 lists the performances of constraint satisfaction inference compared to the majority voting method, both 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.

Table 2 Performances (F-scores) 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.

Task	Baseline	Voting	CSI	Oracle
CHUNK	91.6	93.1	**93.8**	96.2
NER	76.5	82.5	**85.6**	88.9
IMIX	64.7	67.5	**68.9**	74.9

The column labelled "Oracle" in Table 2 represents the performance of the constraint-satisfaction inference method if it would be able, through an oracle, to choose the correct base classifier output among conflicting outputs. The "Oracle" scores thus represent an upper-bound score where all remaining errors are due to the base classifier.

The results show that both the majority voting method and constraint satisfaction inference outperform the naive baseline classifier. In turn, constraint satisfaction inference outperforms majority voting consistently. 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 naive baseline is 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 (2006), 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.

8.1 Comparison to Alternative Techniques

In the experiments reported on, constraint satisfaction inference has been compared with a naive baseline and the majority-voting-based sequence labelling technique in a set-up where the feature sets were the same for all techniques. Although this is arguably the most objective approach to such a comparison, it is certainly true that some methods might actually perform better with more or different types of features. For this reason, another interesting comparison is with other published work using the same data sets. The data sets for syntactic chunking and named-entity recognition have both been standardised as part of the CoNLL shared task, and consequently many additional results on those data sets are available. The top-performing systems for both tasks will be briefly discussed.

Table 3 Comparison of the performance of constraint satisfaction inference on syntactic chunking with other published results.

	Prec	Rec	$F_{\beta=1}$
Ando and Zhang (2005)	94.57	94.20	94.39
Zhang et al (2002)	94.28	94.07	94.17
Kudo and Matsumoto (2001)	93.89	93.92	93.91
CSI	**93.81**	**93.80**	**93.80**
Carreras (2005)	94.20	93.38	93.79
Kudo and Matsumoto (2000)	93.45	93.51	93.48

First of all, the top-performing systems for syntactic chunking are listed in Table 3. The CoNLL-2000 shared task on syntactic chunking took place before machine learning techniques for structured prediction gained widespread popularity. The best system at the time (Kudo and Matsumoto, 2000), using a recurrent sliding-window approach and an extensive set of features, attained an F-score of 93.48. Most other systems scored considerably lower. Structured prediction approaches published since then easily outperform those scores, as does the approach taken here. In fact, most such approaches attain similar scores, as can be seen when looking at the scores for Carreras (2005), Kudo and Matsumoto (2001), and

constraint satisfaction inference. Two systems perform substantially better, though in both cases, this performance gain can be attributed to additional information sources. Zhang et al (2002) use an enriched feature set that includes the output of a full syntactic parser. Without those features, their system reaches an F-score of 93.57. Ando and Zhang (2005) manage to improve performance by employing semi-supervised learning. Their fully supervised system achieves a performance of 93.60.

For named-entity recognition, the picture is slightly different. As can be seen in Table 4, the top-performing systems in the CoNLL-2003 shared task are still amongst the best published to date. These systems did not put extensive effort in structured prediction approaches, but rather used abundant sets of features. Possibly, good global coordination is not as essential for named-entity recognition as it is for syntactic chunking. The fact that most tokens that are part of entities belong to a single class allows carefully crafted local classifiers to perform at a high level for this task. In that light, constraint satisfaction performs rather well, given the limited effort invested in feature optimisation. As an illustration of the complexity of the top-performing systems, Florian et al (2003) combine four different classifiers, and use gazetteer lists comprising tens of thousands of words and even integrate the output of two other entity classifiers. Chieu and Ng (2003) also used large gazetteer lists and in addition, performed extensive feature engineering. Finally, as with syntactic chunking, the best scores published so far involve semi-supervised learning (Ando and Zhang, 2005).

Table 4 Comparison of the performance of constraint satisfaction inference on named-entity recognition with other published results.

	Prec	Rec	$F_{\beta=1}$
Ando and Zhang (2005)	-	-	89.31
Florian et al (2003)	88.99	88.54	88.76
Chieu and Ng (2003)	88.12	88.51	88.31
Klein et al (2003)	86.12	86.49	86.31
CSI	**85.88**	**85.29**	**85.58**
Zhang and Johnson (2003)	86.13	84.88	85.50
Carreras et al (2003)	84.05	85.96	85.00

9 Discussion

The experiments reported on 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. This section attempts to further analyse the behaviour of the inference procedure.

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. At 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 unlikely that such a huge number of candidate label sequences could be dealt with appropriately.

Table 5 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. This is an important observation, since it shows that the output space spanned by the predicted class trigrams is small enough to be searched exhaustively, which in fact was done for all three tasks. Exhaustive search, rather than for example Viterbi search, allows for constraints to cover arbitrary parts of the complete output label sequence. Although this feature was not employed in the current experiments, modelling of higher-level structural dependencies could prove beneficial in optimising entity recognition performance.

Table 5 The average number of potential output sequences that result from class trigram predictions made by a memory-based base classifier.

Task	Base acc.	Avg. # seq.
CHUNK	88.2	57.8
NER	92.3	19.1
IMIX	77.1	9.3

9.1 Other Constraint-based Approaches to Sequence Labelling

Using constraint satisfaction for global optimisation in sequence learning has been explored by others as well. As the following brief overview shows, constraints in these approaches do not stem from base classifiers, but are based on external knowledge.

Constraint Satisfaction with Classifiers (Punyakanok and Roth, 2001) performs the somewhat more specific task of identifying phrases in a sequence. Like the method described here, 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.

In a similar way to this method, classifier confidence estimates are used to rank solutions in order of preference. Unlike in this 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 this 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 this method, constraint satisfaction *is* the inference procedure. Nevertheless, arbitrary global constraints (both hard and soft) can easily be incorporated in this framework as well.

10 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 approaches to sequence labelling use a base classifier as part of a scoring function to guide the search through the output space of all possible label sequences. The constraint satisfaction inference approach presented in this chapter is different in the sense that the base classifier predictions are not only used to score candidate label sequences, but also to restrict the solution space that is explored during inference.

Constraint satisfaction inference builds on the class trigram method introduced by Van den Bosch and Daelemans (2006), 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 three sequence labelling tasks, covering both syntactic and semantic processing, constraint satisfaction inference has been shown to improve substantially on the performance achieved by a simpler inference procedure based on majority voting, which was proposed the original work on the class trigram method. In addition, the method was found to perform on a par with competing sequence labelling methods.

The work presented in this chapter shows there is potential for alternative interpretations of the classification and inference framework. Advantages of this method may be found in faster inference, no restrictions on the type of dependencies that can be modelled, and the fact that it can be used in combination with any type of base classifier. Sequence labelling, while an important class of machine learning problems in natural language processing, is not the only structured output domain to which the constraint satisfaction inference method can be applied. To illustrate this,

the same method was applied to syntactic parsing Canisius and Tjong Kim Sang (2007) and machine translation Canisius and Van den Bosch (2009).

In the larger framework of domain-specific QA, a proper recognition of domain-specific entities in background material with high precision and high recall is important for achieving improved results. The work of Bouma, Fahmi, and Mur (Chapter 9, this volume) complements the work by taking on the empirical question to whether domain-specific entity recognition aids QA.

References

Altun Y, Tsochantaridis I, Hofmann T (2003) Hidden markov support vector machines. In: Fawcett T, Mishra N (eds) Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003), pp 3–10

Ando R, Zhang T (2005) A high-performance semi-supervised learning method for text chunking. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, pp 1–9

Brill E (1994) Some advances in transformation-based part-of-speech tagging. In: Proceedings AAAI '94

Canisius S, Tjong Kim Sang E (2007) A constraint satisfaction approach to dependency parsing. In: Proc. of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, Prague, Czech Republic, pp 1124–1128

Canisius S, Van den Bosch A (2009) A constraint satisfaction approach to machine translation. In: Proceedings of the 13th Annual Conference of the European Association for Machine Translation (EAMT-2009), pp 182–189

Carreras X (2005) Learning and inference in phrase recognition: A filtering-ranking architecture using perceptron. PhD thesis, Universitat Politècnica de Catalunya

Carreras X, Màrquez L, Padró L (2003) A simple named entity extractor using adaboost. In: Daelemans W, Osborne M (eds) Proceedings of the seventh Conference on Natural Language Learning at HLT-NAACL 2003, pp 152–155

Chieu H, Ng H (2003) Named entity recognition with a maximum entropy approach. In: Daelemans W, Osborne M (eds) Proceedings of the seventh Conference on Natural Language Learning at HLT-NAACL 2003, pp 160–163

Chinchor N (1995) Named entity task definition. In: Proceedings of the Sixth Message Understanding Conference (MUC-6), pp 317–332

Collins M (2002) Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In: Hajic J, Matsumoto Y (eds) Proceedings of the ACL-02 conference on Empirical Methods in Natural Language Processing, pp 1–8

Cortes C, Mohri M, Weston J (2005) A general regression technique for learning transductions. In: Raedt LD, Wrobel S (eds) Proceedings of the Twenty-Second International Conference on Machine Learning (ICML 2005), pp 153–160

Daelemans W, Zavrel J, Van der Sloot K, Van den Bosch A (2009) TiMBL: Tilburg memory based learner, version 6.2, reference guide. Tech. Rep. ILK 09-01, ILK Research Group, Tilburg University

Daumé III H (2006) Practical structured learning techniques for natural language processing. PhD thesis, University of Southern California

Doddington G, Mitchell A, Przybocki M, Ramshaw L, Strassel S, Weischedel R (2004) The Automatic Content Extraction (ACE) Program–Tasks, Data, and Evaluation. In: Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04), pp 837–840

Finkel J, Grenager T, Manning C (2005) Incorporating non-local information into information extraction systems by Gibbs sampling. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), pp 363–370

Florian R, Ittycheriah A, Jing H, Zhang T (2003) Named entity recognition through classifier combination. In: Daelemans W, Osborne M (eds) Proceedings of the seventh Conference on Natural Language Learning at HLT-NAACL 2003, pp 168–171

Hirschman L, Yeh A, Blaschke C, Valencia A (2005) Overview of BioCreAtIvE: critical assessment of information extraction for biology. BMC Bioinformatics 6(S1)

Klein D, Smarr J, Nguyen H, Manning C (2003) Named entity recognition with character-level models. In: Daelemans W, Osborne M (eds) Proceedings of the seventh Conference on Natural Language Learning at HLT-NAACL 2003, pp 180–183

Kudo T, Matsumoto Y (2000) Use of support vector learning for chunk identification. In: Cardie C, Daelemans W, Nedellec C, Tjong Kim Sang E (eds) Proceedings of the Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop, pp 142–144

Kudo T, Matsumoto Y (2001) Chunking with support vector machines. In: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies, pp 1–8

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

Lewis D, Yang Y, Rose T, Li F (2004) RCV1: A New Benchmark Collection for Text Categorization Research. Journal of Machine Learning Research 5:361–397

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

McCallum A, Freitag D, Pereira F (2000) Maximum entropy Markov models for information extraction and segmentation. In: Langley P (ed) Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), pp 591–598

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

Ramshaw L, Marcus M (1995) Text chunking using transformation-based learning. In: 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. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, May 17-18, 1996, University of Pennsylvania

Roth D, Yih W (2005) Integer linear programming inference for conditional random fields. In: Proceedings of the 22nd International Conference on Machine Learning, ACM, p 743

Sarawagi S, Cohen W (2005) Semi-markov conditional random fields for information extraction. In: Advances in Neural Information Processing Systems, vol 17, pp 1185–1192

Sha F, Pereira F (2003) Shallow parsing with conditional random fields. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, pp 134–141

Tjong Kim Sang E (2002) Introduction to the conll-2002 shared task: Language-independent named entity recognition. In: Proceedings of CoNLL-2002, Taipei, Taiwan, pp 155–158

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

Tjong Kim Sang E, De Meulder F (2003) Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In: Daelemans W, Osborne M (eds) Proceedings of CoNLL-2003, Edmonton, Canada, pp 142–147

Van den Bosch A (2004) Wrapped progressive sampling search for optimizing learning algorithm parameters. In: Verbrugge R, Taatgen N, Schomaker L (eds) Proceedings of the Sixteenth Belgian-Dutch Conference on Artificial Intelligence, Groningen, The Netherlands, pp 219–226

Van den Bosch A, Daelemans W (2006) Improving sequence segmentation learning by predicting trigrams. In: Proceedings of the Ninth Conference on Natural Language Learning, CoNLL-2005, Ann Arbor, MI, pp 80–87

Van Rijsbergen C (1979) Information Retrieval. Buttersworth, London

Zhang T, Johnson D (2003) A robust risk minimization based named entity recognition system. In: Daelemans W, Osborne M (eds) Proceedings of the seventh Conference on Natural Language Learning at HLT-NAACL 2003, pp 204–207

Zhang T, Damerau F, Johnson D (2002) Text chunking based on a generalization of winnow. Journal of Machine Learning Research 2:615–637