Structured prediction for natural processing
Canisius, S.V.M.

Document version:
Publisher's PDF, also known as Version of record

Publication date:
2009

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright, please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 16. Aug. 2019
Structured prediction
for natural language processing
A constraint satisfaction approach

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Tilburg,
op gezag van de rector magnificus,
prof. dr. Ph. Eijlander,
in het openbaar te verdedigen ten overstaan
van een door het college voor promoties
aangewezen commissie
in de aula van de Universiteit
op vrijdag 13 februari 2009 om 14:15 uur

door

Sander Valentijn Maria Canisius

geboren op 25 mei 1981
te Born
Promotores:
Prof. dr. A.P.J. van den Bosch
Prof. dr. W.M.P. Daeleman

Leden van de beoordelingscommissie:
Prof. dr. H.C. Bunt
Prof. dr. E.J. Krahmer
Dr. L. Márquez
Prof. dr. E.O. Postma
Prof. dr. D. Roth

TiCC Dissertation Series no. 5

This research has been funded by the Netherlands Organisation for Scientific Research (NWO), as part of the IMIX program.

Cover artwork: Ivan Lodde (www.ivanilia.nl)

Copyright © 2009 S.V.M. Canisius

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or otherwise, without the prior permission of the author.

Preface

As I am writing the final words of this dissertation, the time has come to look back at the years that lie behind me and at the part various people have had in the completion of this dissertation.

First of all, I am happy to have worked with Antal van den Bosch. As my supervisor and promotor he has been a continuous source of good ideas and helpful feedback. I feel grateful that he has given me the space to develop my research in directions that I myself found most interesting. His cheerful and accessible personality has made our cooperation a pleasant one. Even though over the years I have witnessed the number of people demanding more and more of his time increase, he always managed to make time for a chat about work, life, and inevitably always about this one city we both know so well.

As my other promotor, Walter Daelemans has always been an enthusiastic supporter in the background. The geographical distance between our work places also created the objective perspective that proved beneficial in recognising the issues in my work that Antal and I overlooked. Furthermore, the times that we sat down for discussion more often than not resulted in clues about things to consider or existing work to check.

Both Antal and Walter I have to thank for reading the draft versions of this dissertation and for their many keen remarks on both textual and content matters. They are also the ones that started the IMIX Rolaquad project in the context of which this dissertation has been written. I thank all my colleagues of the IMIX project, in particular my colleague on the Rolaquad project Piroska Lendvai, for the pleasant cooperation on a project I have enjoyed working on greatly. In this context, I also wish to acknowledge the funding of the Netherlands Organisation for Scientific Research (NWO) that has made this research possible.

I wish to thank the members of the reading committee for taking the time to read this dissertation and for making the effort to attend my defence: Harry Bunt, Emiel Krahmer, Lluís Márquez, Eric Postma, and Dan Roth. I am fortunate that in one way or another all of them have crossed my path before and that each of those encounters have positively affected my skills as a researcher.

While being involved in research is already an exciting and fulfilling experience in its own right, I would not have enjoyed the past years as much as I did without my colleagues at Tilburg University. As not to risk omitting anyone, I will not
mention all of you by name, but you know who you are. I will keep fond memories of all our joint lunches, outings, and after-work drinks.

I have also had the pleasure to work with many talented researchers during my stay at the University of Illinois. Thank you, Dan Roth, for giving me this opportunity, and thanks to all the members of the Cognitive Computation group for welcoming me as a guest in their group, and for the many insightful discussions. So short before I started writing this dissertation, this visit has had an important impact on many parts of it.

Thanks are also due to Ivan Lodde for making the cover illustration of this dissertation. His illustration is a true finishing touch that will make me even more proud of my dissertation.

Finally, I cannot start to express my gratitude for those who have always been there for me, and always will be. My friends, and most of all, my family have been a perpetual source of support. I feel fortunate to be part of such a fun and loving family and to have such dear friends. Their encouragements, help, and companionship have always been invaluable, and definitely helped me get through the last phase of writing this dissertation.
# Contents

1 Introduction
   1.1 Natural language processing .......................... 1
   1.2 Machine learning for natural language processing ........ 3
   1.3 Structured prediction .................................. 5
   1.4 Research objectives .................................. 6
   1.5 Thesis outline ....................................... 8

2 Machine learning for structured prediction 9
   2.1 Supervised machine learning .......................... 9
   2.2 Structured prediction .................................. 11
      2.2.1 Multi-label classification .......................... 12
      2.2.2 Hierarchical classification ......................... 13
      2.2.3 Label sequence prediction .......................... 13
      2.2.4 Prediction of tree structures ....................... 14
      2.2.5 Joint prediction .................................... 15
      2.2.6 Discussion ......................................... 16
   2.3 Existing work ......................................... 16
      2.3.1 Recurrent architectures, pipelines, and Searn .......... 17
      2.3.2 Structured linear models ............................. 20
      2.3.3 Constraint satisfaction with classifiers ................ 23
      2.3.4 Output kernels ..................................... 25
      2.3.5 Discussion .......................................... 27
   2.4 Conclusion ............................................ 28

3 Structured prediction as constraint satisfaction 31
   3.1 The anatomy of a structured prediction problem ............ 31
   3.2 A toy problem: multi-label text categorisation .............. 32
   3.3 Towards constraint satisfaction inference for multi-label classification 34
      3.3.1 Micro-label prediction ............................... 35
      3.3.2 Macro-label prediction ............................... 36
      3.3.3 Meso-label prediction ............................... 38
3.3.4 Bounds on the performance of overlapping meso-label prediction ........................................ 40
3.3.5 An inference procedure for overlapping meso-labels ......................................................... 42
3.4 A general formulation of constraint satisfaction inference .................................................... 49
3.4.1 Output space ......................................................... 49
3.4.2 Base classifiers .................................................. 50
3.4.3 CSP formulation .................................................. 51
3.4.4 Solving the CSP .................................................. 52
3.5 Comparison with related work ................................................. 53
3.6 Conclusion ......................................................... 56

4 Sequence labelling .................................................. 59
4.1 Existing work .................................................... 60
4.2 Constraint satisfaction approach ...................................... 60
4.2.1 Solution space .................................................. 61
4.2.2 Constraints ..................................................... 61
4.2.3 Solving the CSP .................................................. 63
4.3 Sequence labelling tasks .................................................. 64
4.3.1 Syntactic chunking .............................................. 64
4.3.2 Named-entity recognition ...................................... 65
4.3.3 Letter-phoneme conversion ..................................... 67
4.3.4 Morphological analysis ........................................... 68
4.4 Experimental setup .................................................. 69
4.4.1 Evaluation ........................................................ 69
4.4.2 Constraint prediction ............................................. 70
4.5 Results .............................................................. 73
4.5.1 Comparison to alternative techniques ...................................... 76
4.5.2 Comparison to other published results ...................................... 78
4.6 Analysis of base classifier impact .................................... 79
4.7 Conclusion ........................................................... 82

5 Dependency parsing .................................................. 85
5.1 Existing work .................................................... 86
5.1.1 Transition-based models ........................................... 86
5.1.2 Graph-based models ............................................. 87
5.2 Constraint satisfaction approach ...................................... 88
5.2.1 Solution space .................................................. 88
5.2.2 Constraints ..................................................... 89
5.2.3 Solving the CSP .................................................. 93
5.3 Experimental setup .................................................. 93
5.3.1 Data ............................................................ 93
5.3.2 Evaluation ...................................................... 94
5.3.3 Constraint prediction ............................................. 95
5.4 Results ............................................................. 99
Contents

5.4.1 Dependency constraint ........................................ 100
5.4.2 Modifiers constraint ........................................... 101
5.4.3 Direction constraint ........................................... 102
5.4.4 Overall parsing performance ................................. 103
5.5 Conclusion .......................................................... 107

6 Machine translation ................................................. 109
6.1 Data-driven machine translation ................................. 110
6.2 Existing work ....................................................... 111
   6.2.1 Statistical machine translation ............................. 111
   6.2.2 Example-based machine translation ....................... 113
   6.2.3 Discriminative machine translation ....................... 114
6.3 Constraint satisfaction approach ................................ 115
   6.3.1 Solution space .............................................. 116
   6.3.2 Constraints .................................................. 120
   6.3.3 Solving the CSP ............................................. 121
6.4 Experimental setup ............................................... 123
   6.4.1 Data ......................................................... 123
   6.4.2 Evaluation .................................................. 123
   6.4.3 Constraint prediction ...................................... 124
6.5 Results ............................................................ 127
6.6 Conclusion ........................................................ 128

7 Conclusions .......................................................... 131
7.1 Discussion of the research objectives ......................... 131
7.2 How generic is constraint satisfaction inference? ........... 136

References ............................................................. 139

Summary ............................................................... 155

Samenvatting .......................................................... 159

TiCC dissertation series ............................................. 163
Chapter 1

Introduction

1.1 Natural language processing

The advent of computerised storage and processing of information brought with it the promise of instantaneous access to sources of knowledge previously only available on the shelves of libraries or in the minds of experts. Despite the impressive progress that has been made in only a short amount of time, this promise has thus far only partially been fulfilled. Successful computer-based information systems are typically the result of manual knowledge engineering involving both computer scientists and experts in the domain of the application. Thus, rather than instantaneous access, the route to the information is indirect, first requiring a costly engineering process. It is indirect, and most likely will remain so, because the primary means for communicating and storing information is in the form of written and spoken language. All but the most implicit of human knowledge has been and is being recorded in documents that are primarily aimed at human consumption; whether or not it is suited for automated processing is not a point of consideration. Consequently, the only way computers can facilitate access to this vast amount of readily available knowledge is making computers understand the language in which it is recorded. This is the goal of natural language processing.

For unrestricted language understanding, probably only human-level intelligence will suffice. It is widely accepted that large parts of the meaning of any language utterance is not encoded in the words that we write or say, but rather in the prior knowledge of the reader or hearer. The current direction in natural language processing, therefore, is not to aim at general understanding of language, but rather to define more restricted, more attainable subtasks. Many such subtasks involve mapping one representation of a language utterance to another representation.

Written text is one such representation; often, it is the input representation, that is, the representation mapped from. This is the situation for applications where some kind of understanding of a text is intended. Written text may also be the output representation, though. This would be the case if the aim is not to
understand, but rather to communicate some message. Even the combination of the two is possible. For example, if text is translated from one language to another, or if a summary of one or several documents is generated, both the input and the output representation is written text. The majority of processing tasks performed in natural language processing are not of the direct text-to-text kind though. Even the two aforementioned tasks may involve some internal processing in which more abstract representations are generated and used as part of the understanding of the input text and the generation of the output text. Typical natural language processing systems encompass several components, each responsible for mapping between representations, some textual, others more abstract.

Several common processing tasks will return in more detail further on in this dissertation; others will sometimes be referred to for illustration purposes. The following serves as a brief introduction to the areas of natural language processing to which we will return in later chapters.

**Phonological processing** is concerned with the mapping between written words and phonemic representations of their pronunciation. An important phonological processing task is *letter-phoneme conversion*, which may be used by text-to-speech conversion systems to recover the pronunciation of words that are not included in their pronunciation dictionary.

**Morphological processing** deals with the analysis of the internal structure of words into smaller meaning-bearing units called morphemes. A *morphological analysis* of a word divides it into its morphemes and can recover such phenomena as inflection, derivation, and compounding. Various degrees of morphological analysis may be relevant to such applications as information retrieval and spell checking.

**Syntactic processing** analyses the ways in which separate words combine to form sentences. Various forms of syntactic analysis, or parsing, are commonly performed in natural language processing. *Part-of-speech tagging* assigns grammatical word classes to each word in a sentence based on the words surrounding it. *Shallow parsing* segments sentences into non-overlapping syntactic constituents, and thus unlike part-of-speech tagging, groups related words together into larger syntactic units. *Full syntactic parsing* takes this even further and recovers all grammatical relations between words in a sentence.

**Information extraction** covers a multitude of tasks that involve the extraction of structured information from unstructured written text. *Named-entity recognition* aims to mark passages of text that refer to real-world entities, such as persons and locations in news texts, or genes and proteins in biomedical literature. *Relation extraction* is a typical follow-up task to named-entity recognition, in which semantic relations between previously found entities are detected, for example, a
lives in relation between a person and a country, or a relation denoting interaction between proteins.

**Machine translation** is one of the oldest applications of natural language processing, and arguably one of the most challenging. Automatically translating from one natural language to another involves correctly dealing with many of the subtleties—morphological, syntactic, semantic—of both languages, and thus subsumes many of the above tasks. It is also one of the areas in natural language processing—summarisation and paraphrasing being other examples—that is not only concerned with text understanding, but with text generation as well.

### 1.2 Machine learning for natural language processing

Natural language processing tasks, such as those introduced in the previous section, are typically defined in terms of what the input and the intended output look like. The tasks in themselves do not suggest any specific method for performing them. When performing syntactic parsing, that task has a well-defined kind of output, e.g. some kind of syntactic structure. However, the task in itself does not say anything about how to arrive at that structure when given a natural language sentence as input. Devising methods that actually implement this mapping, rather than simply describing it, is at the core of what natural language processing is about. Given an input representation and a desired output representation, there are still infinitely many ways of getting from the one to the other, and many have indeed been tried. Over time, certain trends can be observed in the general types of methods in use. In the last few decades, an important shift has taken place in natural language processing. Previously, the dominant strategy for implementing automatic linguistic mappings was based on manually constructing large sets of rules that describe the mapping between input and output, such is the case, for example, with context-free grammars for syntactic parsing (Chomsky, 1956; Earley, 1970), two-level morphology for morphological analysis (Koskenniemi, 1983), and extraction patterns for named-entity recognition (Appelt et al., 1993).

This traditional strategy comes with some severe disadvantages. It is costly, since manual labour is required extensively to craft well-performing rule sets. It is inflexible as well; e.g. grammars written for parsing Dutch are difficult to adapt for parsing French. Because of this, manual knowledge engineering has to be performed virtually from scratch for those two tasks, even though there are many obvious similarities between the two. These issues, which are inherent to knowledge-based approaches, caused a steady growth of the interest in statistical and machine learning methods for natural language processing. The accelerated development of computing resources and the increasing availability of large annotated corpora both contributed to the success of those methods. This shift from
knowledge-based to empirical methods mainly started in the 1980s (Sejnowski and Rosenberg, 1987; Garside et al., 1987; Church, 1988), although many of those early approaches trace back their roots to the statistical methods for automatic speech recognition developed in the 1970s (Baker, 1975; Jelinek et al., 1975). Since then, an increasing amount of tasks previously the domain of rule-based methods have been taken over by approaches based on machine learning.

Nowadays, a wide variety of different machine learning techniques are in use for dealing with an equally diverse set of linguistic processing tasks. The common factor among all of those is that they are built by learning to perform a task from examples. Instead of manually writing a procedural description of how to perform a mapping, a system is simply given examples of the input representation and the output representation it is supposed to generate for that input. For example, a data-driven syntactic parser is given large amounts of sentences together with their syntactic analysis. From those examples, it should learn to predict the output representation for new inputs, that is, inputs for which the true output form is unknown.

Though it is possible to devise a completely new learning technique specifically aimed at performing one processing task, mature general-purpose learning approaches have already been developed outside of the area of natural language processing. Those approaches, then, are not only applicable to linguistic processing, but also for such tasks as image recognition, fraud detection, and basically just any task for which suitable examples are available. Such generality is both a curse and a blessing. As for the latter, improvements in a general-purpose learning technique immediately benefit the performance of all applications in which that technique is employed. On the other hand, the generality of the learning techniques does give rise to an extra level of indirectness. To ensure applicability to a wide array of different tasks, input and output objects are necessarily more abstract. A learning algorithm is oblivious to such concepts as words, sentences, and parse trees. Therefore, mappings should be devised between the linguistic concepts of natural language processing, and the abstract input and output objects of the learning algorithm.

Apart from some exceptions, input objects are vectors of features, each describing some aspect of the true, task-specific input object. Typical features in natural language processing include “the current word is The”, “the current word starts with a capital letter”, or “there are 10 occurrences of the word ‘he’ in the input object”. Given input objects described in terms of such feature vectors, the learning task comes down to learning the relation between values of features and values of the output. Most learning algorithms operate in that way. As a recent development, kernels allow some learning algorithms to operate on task-specific input object descriptions. Internally, kernels map the objects to a high-dimensional feature space, which ensures that the same learning algorithm can be used as for traditional feature vectors.

Similar to the input space, output spaces are typically more generic than the task-specific output representations dealt with in natural language processing. For
1.3. Structured prediction

In the previous section, it was argued how there is an apparent mismatch between the output space of typical natural language processing tasks and that of traditional classification methods. The richness of the output structures in the former makes that there are many, sometimes infinitely many different output values. Learning to predict such values as a single classification simply becomes impossible, if only due to sparse training data. Splitting up the output structure and predicting it in separate parts makes the learning task more manageable, but doing so may result in loss of performance, as a result of not being able to take into account interactions within the output structure.

Until recently, two types of strategy with regard to this have been in use. The first is simply to ignore interactions internal to the output structure and try to make a local classifier as good as possible to compensate for the potential loss. While this may work in some cases, taking into account global information is often essential for good performance. A second strategy, therefore, explicitly models global phenomena within output structures. Most approaches following this strategy can be seen as performing a combinatorial optimisation in an output space that spans all possible outputs for a given input. In such approaches, there is a learning component that is responsible for learning the objective function, and a search, or inference, component that finds the output structure that maximises the objective function. Some areas in natural language processing already developed such techniques in an early stage. This has mainly been the case for those areas in which acceptable performance is simply not possible without doing so; for example, machine translation, and speech recognition. However, the techniques devised for those applications are not generally applicable to all linguistic processing tasks in which structure plays an important role. Structured prediction is an emerging field within machine learning that aims at designing generic techniques that explicitly model structural properties of the output space.

Currently a few frameworks exist for learning structured prediction tasks; for example, conditional random fields (Lafferty et al., 2001), and Searn (Daumé III, 2006). Nevertheless, structured prediction remains an issue with many questions still waiting for a definite answer. How do we best deal with the sparsity issues that inevitably result from rich output structures? What is the best way to search the output spaces, which are typically astronomical? Is it better to make restricting assumptions, so that optimal search becomes possible, or will approximate
search, allowing for more realistic assumptions, lead to better results? Finally, should the training procedure already take into account the effect of this search algorithm, or should it assume perfect search? Tentative answers to some of these questions are suggested by earlier studies, though a complete understanding of their implications has yet to be developed. In fact, Lafferty and Wasserman (2006) name structured prediction as one of three open challenges for machine learning research.

### 1.4 Research objectives

Even though some frameworks for structured prediction exist and have been applied successfully to some of the most challenging problems in natural language processing, there are still unanswered issues that might hinder widespread adoption of structured prediction. Our aim in this study is to explore alternative directions in structured prediction, and we do so by developing a new framework for structured prediction. In this new framework we aim to find satisfying solutions for some of the issues mentioned in the previous section. Therefore, we list the following three objectives for our study.

**Efficient inference without restricting assumptions**

Because of the richness of the linguistic representations involved in typical natural language processing tasks, the corresponding output spaces are potentially large. The complexity of search in those spaces, therefore, is an issue not to be taken lightly. There is no escaping the fact that any approach to structured prediction will have to take measures to restrict the cost of search at least to some extent. One option for doing so is by making restricting assumptions about the output space, for example, the Markov assumption. This option may enable efficient dynamic programming algorithms for optimal search under the given assumption. On the negative side, the expressiveness of the objective function is reduced, since it cannot take into account arbitrary features of the output structure. Another option is not to make any assumptions about the output space, but rather to rely on approximate search methods to explore the output space. As an obvious disadvantage, one loses the guarantee for optimality. However, the increased richness of the objective function might make up for this.

The aforementioned options are but two possibilities for dealing with the search problem in structured prediction. Both have been applied successfully to natural language processing tasks, but as explained, both have their own strengths and weaknesses. In this study, we want to explore yet another solution to the search problem that is less restricted by assumptions about the search space, and less prone to end up in local optima than approximate search methods.
1.4. Research objectives

Efficient training

Most approaches to structured prediction are somehow based on local models. For predicting a complete output structure, these local models are used to guide a search process that determines the globally optimal output. An important decision to make is how these local models should be trained. One option is to train locally, using conventional multi-class learning algorithms. This way, their role in the search process is ignored. The alternative is to train while already taking into account this role. Conditional random fields, for example, are trained by performing full inference as part of the training procedure. Searn—although it does not need an inference step for classification—requires several training iterations in which the full output structures are predicted and evaluated, after which appropriate parameter updates are made. Requiring search during training makes training expensive. Besides, when taking such a global view, it may be difficult to determine how different components contribute to the end result, and therefore credit assignment may be difficult.

Punyakanok et al. (2005) took a closer look at the necessity of inference as part of the training procedure, as opposed to local training, and only doing inference for classification. Their conclusion is that inference during training helps if the local classifications are difficult to learn in isolation. If, however, it is possible to train local classifiers sufficiently accurately, inference during training does not enhance performance any further. As already mentioned, inference during training yields a substantial overhead, and thus a structured prediction technique that does not require it is to be preferred. Therefore, the second objective of this study is to devise a structured prediction framework that does not require inference to be part of training.

Compatibility with existing learning techniques

For traditional learning scenarios, such as classification, many different techniques have been developed over the years. Each of those techniques tends to have its own strengths and weaknesses, and it is assumed that no technique exists that is superior in every aspect (Wolpert, 1996, 2001). Because of this, a wide variety of different learning approaches have traditionally been in use in natural language processing. In contrast, the area of structured prediction is dominated by linear classification models, such as maximum-entropy models and support vector machines. Mainly, this is because many structured prediction approaches trace back their roots to the work of Lafferty et al. (2001) on conditional random fields, which used a linear classification model. In the ideal case, though, structured prediction would offer the same flexibility with respect to learning algorithms as traditional classification. As a final objective of our study, we therefore aim for a structured prediction framework that is compatible with any existing learning technique.
1.5 Thesis outline

The remainder of this dissertation is organised as follows. Chapter 2 serves as an introduction to the area of structured prediction. The first half of the chapter is spent defining the notion of a structured prediction task, and explaining why conventional classification approaches fall short on such tasks. Next, an overview of prior work on structured prediction is presented. In this overview, we focus on approaches that are generically applicable to many different structured prediction tasks. This excludes approaches that could certainly be characterised as performing structured predictions, but are only aimed at one specific natural language processing task. Such approaches, if relevant, will be described in later chapters.

Chapter 3 describes the main research product of this dissertation. It presents constraint satisfaction inference, a generic framework for structured prediction. For introducing and explaining the most important concepts behind the framework, it is introduced by applying it to a relatively simple kind of structured prediction task, multi-label document categorisation. Following this introduction by example, the main concepts are summarised and restated in more general terms, so as to allow them to be applied to other structured prediction tasks as well.

Whereas Chapter 3 lays the conceptual foundation for the rest of this dissertation, the subsequent Chapters 4, 5, and 6 provide the empirical support of the dissertation. To verify the generic applicability of constraint satisfaction inference it is applied to sequence labelling, dependency parsing, and machine translation, respectively. Although this set constitutes a diverse range of different linguistic processing tasks, they all share the characteristic that their intended output has a complex structure, which makes that structured prediction techniques are needed to perform them accurately.

Chapter 7 returns from the task-specific focus of previous chapters to a bird’s-eye view of structured prediction in general. We look at the objectives formulated in this introduction chapter, and summarise how constraint satisfaction inference as implemented in the previous chapters meets those. We conclude by discussing the generality of our approach, and in that respect, comparing our framework to other approaches to structured prediction.
Chapter 2

Machine learning for structured prediction

2.1 Supervised machine learning

Machine learning is the subfield of artificial intelligence that aims to achieve intelligent behaviour not through carefully engineered human knowledge, but by learning the knowledge implicitly stored in data. In its most general form, it is concerned with finding and modelling the regularities in those data. More focused, machine learning is typically employed to cluster similar objects, or to learn a function, e.g. a real-valued function or a decision function. The former is the typical aim of unsupervised learning. The data comprises a raw collection of objects, and a learning algorithm finds patterns shared by certain objects, and groups the objects according to those. In this dissertation, we focus on supervised learning. In this setting, the data not only correspond to raw objects. The objects are explicitly labelled with a target value, which the learning algorithms should learn.

To formulate the supervised learning setting more specifically, we assume input objects \( x \) are taken from a space \( \mathcal{X} \), and every such input object corresponds to a target value \( y \), which is taken from an output space \( \mathcal{Y} \). It is the task of a learning algorithm to come up with a model that knows how to map arbitrary input objects to their correct target values. For this, it is given access to a set of labelled examples \( \{(x_1, y_1), \ldots, (x_m, y_m)\} \subseteq \mathcal{X} \times \mathcal{Y} \). Learning this mapping successfully involves finding regularities in the relation between features of the input on the one hand, and the output values on the other hand. For doing so, various different strategies exist, and consequently, many different types of models and learning algorithms have been developed. Though the underlying strategies differ, their goal is not so different. Good prediction models are the ones that not only learn to label the training instances correctly, but in addition, generalise well to instances not encountered in the training data. How well a learned model will really generalise depends on a number of factors, though. One of them is the effort
invested by the learning algorithm, for example, to prevent overfitting. Others include the general difficulty of the learning task, and the informative value of the features. Finally, the shape and size of the output space, possibly in combination with the amount of training data, is also an important factor in generalisation performance.

Focusing somewhat more on the nature of the output space, we can distinguish certain classes of supervised learning problems that each have specific types of output spaces. In regression problems, for example, the output space is a continuous real-valued space, typically $\mathbb{R}^N$ for some $N \geq 1$. Labelled examples in such problems combine a feature description of the input and a numeric output value. Due to the continuous nature of the output space, no training set can ever include an example for each possible output value. At the same time, though, this continuity allows the model to predict a numeric value by somehow interpolating from what it has seen in its training data. As a result, every output value can be predicted, even those that do not appear in the training data.

In classification problems, another class of supervised learning problems, the output space is a discrete one. In most cases, the labels—also referred to as classes—assigned to instances are symbolic rather than numeric. Binary classification, the situation where there are only two classes, is often distinguished as a special subclass of classification problems. The restricted output space makes this setting an attractive subject for theoretical analysis. Linear models, for example, originally devised for binary classification, are a particularly well-studied class of classification models. Various different optimisation criteria and algorithms have resulted in some of the best-performing and best-understood learning techniques to date, such as support vector machines (Boser et al., 1992), perceptrons (Rosenblatt, 1958; Freund and Schapire, 1999), and logistic regression (Berger et al., 1996). While the binary output space of these techniques may seem a problematic restriction, there are methods with which any multi-class problem can be reduced to a number of binary decision tasks. The simplest of those methods, one-versus-all classification, trains a binary classifier for each of the different output classes and predicts the class whose classifier predicts positively with the highest confidence. Other multi-class to binary conversion schemes include pairwise classification (Hastie and Tibshirani, 1998), and error-correcting output codes (Dietterich and Bakiri, 1995).

As an alternative to the approaches that reduce multi-class classification to a binary setting, other methods exist that can learn a multi-class classification problem directly. Examples of such methods include decision trees (e.g. Quinlan, 1986, 1993), and $k$-NN classifiers (Cover and Hart, 1967; Aha et al., 1991). In addition, generalised linear classifiers, which will be discussed in more detail later on in this chapter, allow for direct multi-class classification with linear models. Direct learning of multi-class classification models, as opposed to reducing the task to several binary classification problems, might allow the learning algorithm to better exploit the interactions among the various classes.

While there is no theoretical limitation to the number of classes learned by
a multi-class classifier, there is a rather important practical one. Learning to recognise a class requires a certain amount of training examples. Computational learning theory shows that the sample complexity, i.e. the amount of training data required to successfully learn a model, grows with the complexity of the model. Thus, the larger the number of classes to be recognised, the more training data will be needed. Unfortunately, additional training data for supervised learning is expensive to obtain. Moreover, the number of training examples one can use is always finite, yet if the output space to be learned has an infinite size, only infinitely many training examples would suffice.

This practical limitation of multi-class classification has implications for what can and cannot be dealt with as a single classification task. Focusing on natural language processing, there are many tasks that involve learning the mapping between complex input and output structures. For example, in syntactic parsing, the aim is to learn the mapping from natural language sentences to syntactic parse trees. Conceptually, one would like to see such a tree as the target value to be learned, and therefore the output space as the infinite space of all possible parse trees. With the aforementioned relation between output space complexity and training data requirements in mind, though, it is easy to see that conventional multi-class learning strategies cannot cope with such an output space. The example of syntactic parsing is but one of the many tasks in natural language processing for which this issue arises. In many other domains, for example bioinformatics and computer vision, complex output structures occur frequently as well. For this reason, extending existing machine learning methods to perform complex prediction tasks is an important research problem.

2.2 Structured prediction

Before giving a survey of existing work on structured prediction, let us first try to provide some insight into what is actually meant when we speak about structured prediction. As a first way to define structured prediction, we can show how it differs from more traditional supervised learning problems. Two such problems, classification and regression, have been discussed in the previous section. Both are concerned with predicting the value of a single output variable. For classification this output is symbolic, for regression it is a continuous value. The learning task in both scenarios involves learning a relation between features of the input and the potential output values. Predictions according to the learned model are based on correlations between features of the instance and each of the possible target values.

Structured prediction is similar to traditional classification in the sense that both operate in a symbolic output space. Yet, it would be naive to regard structured prediction problems as multi-class learning tasks that happen to involve huge numbers of classes. Unlike the classes of conventional classification, the output values of structured prediction tasks have a complex internal structure.
Syntactic parse trees are not simply atomic values, but are built from syntactic relations connecting the words of a sentence. Likewise, the pronunciation of a word is in fact a sequence of phonemes, rather than a single symbol. Thus, instead of thinking of structured prediction as mapping an instance to a single output value, it makes more sense to regard it as the joint prediction of several values, namely the parts of the complex output structure. In abstract terms, we will treat the target value of a structured prediction task as a vector of discrete symbols, i.e. $y \in \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_n$, where the length $n_y$ of this vector is not a property of the task, but of the classification case at hand.

When learning a structured prediction model, it is still important to find relations between features of the input and output values, but additionally, effort should be spent on modelling the structural correlations among the variables making up the complex output structure. This additional modelling step is what distinguishes structured prediction most from multi-class classification. In the latter, every predicted value is considered independent of all other predictions; hence, there is no need to model interactions among them. In the former, ignoring those interactions may negatively affect the prediction accuracy, or even result in invalid output structures; for example, a syntactic parse that is not a valid tree, or a phoneme sequence that contains phoneme subsequences that can never occur.

Moving on from this abstract definition of structured prediction, there are a multitude of different forms of structured prediction problems that occur in practical situations. Natural language processing is of particular interest for structured prediction, since a vast majority of problems in this area have to do with structure in one way or another. In the following, a number of structured prediction tasks are introduced and illustrated by problems from natural language processing.

### 2.2.1 Multi-label classification

A typical document classification scenario asks to predict a category for a given document. The document can be about politics, sports, health, etc. In the simplest case, document categories are mutually exclusive and thus, document classification becomes standard multi-class classification. Often however, it is preferable for document categories not to be mutually exclusive. A document about sports-related injuries should be classified as dealing with both sports and health; a text about doping regulations might even be classified as dealing with all three categories mentioned. In the case in which more than a single category can be assigned to an instance, multi-class classification does not cover the problem anymore. The new problem is referred to as multi-label classification.

Conceptually, the move from multi-class classification to multi-label classification is an easy one. Instead of selecting one label out of a set of possibilities, the goal is to select a subset of possible labels that are relevant for the input. It can be cast as a number of binary classification tasks, one for each of the possible labels. The decision to make for each label, then, is whether or not it applies to the current input. However, in multi-label classification tasks, the outcomes of
these decisions are strongly intercorrelated. For example, some labels are likely to occur simultaneously, others may be mutually exclusive. Therefore, it is best to base individual decisions not only on the input, but also on the predictions for other labels. This perfectly fits the definition of structured prediction.

### 2.2.2 Hierarchical classification

In the same way that mutually-exclusive categories may be too harsh an assumption for some document classification tasks, it may also be inappropriate to consider all classes equally different. A text about cycling and another text about soccer treat different topics, and therefore may have to be classified as such. However, if compared to yet another text about music, it is often the case that the two former texts are to be considered more similar—both are about sports—than the latter. In this example, document categories form a hierarchy, where documents can belong to the same category on one level, but actually have different categories according to lower levels. The fact that two different categories can still share some features that set it apart from yet another category can be exploited by explicitly modelling the path from the top of the hierarchy down to the bottom-level category. Doing so means that an input will be classified according to each level of the hierarchy; obviously, those classifications are correlated.

There is a straightforward way to employ traditional classification techniques to tackle hierarchical classification. This involves training separate classifiers for each node in the hierarchy that classify an instance as one of the categories directly beneath their node. For predicting the hierarchical label of a new instance, one starts classifying at the top of the hierarchy using the relevant classifier. The prediction of this classifier is used to select the next classifier, which classifies the instance according to its own position in the hierarchy. This process is repeated iteratively until a leaf node is reached. If predictions at the higher levels are easier to make than those at lower levels, this strategy will do fairly well. For many tasks this will indeed be the case. However, what if it happens that two categories are easy to set apart on a lower level, but an incorrect higher level classification makes that the final category is still predicted incorrectly? In the strategy just described, high confidence at lower levels of the hierarchy cannot affect decisions at higher levels. On the other hand, if all levels are predicted jointly, that will be possible.

### 2.2.3 Label sequence prediction

Sequences are one of the most important output structures in natural language processing. Many linguistic processing tasks are naturally cast as mapping a linguistic input to a sequence of symbolic labels. If we want to predict the pronunciation of a written word, the conventional way to do this is by generating a sequence of phonemes. Similarly, in part-of-speech tagging, the output of the task is a sequence of grammatical word categories. What makes both the afore-
mentioned tasks good candidates for a structured prediction approach is the fact that not only is there a relation between letters and the corresponding phoneme, or words and their matching part-of-speech tag; there is also strong interaction among, mostly subsequent, phonemes, or part-of-speech tags.

As an illustration of this interaction, consider the task of predicting the correct grammatical category for the word “spoke”. Prior probability might suggest the word should be tagged as a verb. Possibly, that is even the only tagging for the word “spoke” found in the training data. However, if the previous word is known to be a determiner, tagging “spoke” as a noun is a much better option. In order to see this, the prediction model should know that finite verbs do not tend to follow determiners, while nouns do. In other words, the model should include the interaction between output labels.

In addition to processing tasks that are obviously concerned with predicting label sequences, it is also fairly common to reformulate certain tasks to a sequence labelling task. Sequence segmentation tasks are concerned with dividing an input sequence into several segments, possibly also labelling those segments. Again, there are many applications for this in natural language processing. First of all, considering sentence segmentation tasks, recognising low-level syntactic phrases—known as shallow syntactic parsing—or detecting references to real-world entities, are important sequence segmentation tasks. Relevant applications of segmentation can also be found at the word level, such as recognising the morphemes in a word form.

The classic conversion scheme for sequence segmentation tasks to a sequence labelling problem is due to Ramshaw and Marcus (1995). Each element of the input sequence, e.g. each word of a sentence, is mapped to a symbol that denotes whether that element starts a new segment, continues a segment, or is not part of a segment. If labelling of the segments is also part of the task, the segment label is simply appended to the predicted class label.

### 2.2.4 Prediction of tree structures

Next to sequences, trees are arguably the most important output structures in natural language processing. In particular, most grammar formalisms use tree structures to encode recursive grammatical structures; this is the case, for example, for constituent and dependency structures. Tree-structured encodings are also popular for representing semantic analyses, such as predicate-argument structures.

As an example, consider syntactic dependency parsing, which will be revisited in Chapter 5. A dependency analysis of a sentence links each word via a grammatical dependency category to another word in the sentence; it is said that the former word modifies the latter. It may make sense to predict such dependencies individually, but not taking into account other parts of the output risks ending up in an invalid output. In dependency parsing, but more generally in all tree prediction tasks, the most important type of interaction among output variables
is a direct consequence of the mathematical definition of trees: a valid tree does not have any cycles. For dependency parsing, this means that a word cannot directly or indirectly modify itself. Thus, what may look like a good dependency relation in isolation may not be a valid contribution to the dependency tree, since it would result in a cycle. Joint prediction of all relations will ensure that the best local predictions are found given the tree constraint.

2.2.5 Joint prediction

A typical phenomenon in natural language processing is that a single input, say a sentence, is analysed to recover multiple types of linguistic structure, and those analyses are interrelated. For example, in information extraction, one may want to recover both entities and relations between entities. The interaction here is straightforward: in order to find relations between entities, it should be known what those entities are. Entity and relation extraction is an example where both pieces of information are of practical interest. In another scenario, one may want to recover additional structure because it makes another processing task easier. For example, syntactic parsing is known to be easier if we already know the part-of-speech tags of the words. Similarly, semantic role labelling is often performed using the syntactic analysis of a sentence. In these two examples, the intermediate information that is produced, i.e. part-of-speech tags and syntactic analyses, are not the desired end result, but the abstraction they provide makes the real task easier to perform.

The typical approach to such interrelated processing tasks is to perform the predictions along a pipeline, in which processing components produce output that subsequently serves as input for following processing components. This way, later processing components can base their predictions on the input, as well as on analyses produced by earlier steps. So, an information extraction pipeline can consist of an entity recogniser followed by a relation detector. A pipeline for semantic analysis connects a part-of-speech tagger, a syntactic parser, and a semantic role labeller, in that order.

A drawback of the standard pipeline model results from the fact that the processing components are ordered sequentially. If a relation detector follows an entity recogniser in a pipeline, the latter only starts after the relation detector has finished. Because of that, the relation detector can use information produced by the entity recogniser, but it is not possible for the entity recogniser to base its predictions on information produced by the relation detector. Perhaps the information about a certain semantic relation between two entities allows the entity recogniser to make better classifications, but semantic relations are only predicted after the entity recogniser is finished. A semantic role labeller may have strong evidence for a semantic link between two words, but cannot predict it, because the syntactic parser did not produce the required syntactic link.

A structured alternative to the pipeline architecture would perform two or more processing steps simultaneously. For example, recognising entities and de-
tecting relations between them at the same time, rather than first doing the one and only then the other. The same could be done for syntactic parsing and semantic role labelling. Because the two related analyses are predicted jointly, information is exchanged in both directions.

### 2.2.6 Discussion

The above survey is merely a selection of classes of processing tasks that can benefit from a structured prediction approach. Despite not being an exhaustive overview, it covers a large part of the tasks dealt with in natural language processing. More importantly, it shows how all these tasks involve the joint prediction of multiple output variables, whether they be the individual levels of a hierarchically-structured category label, the elements of a label sequence, or the interdependent analyses produced by an information extraction system. This, above all, is what makes structured prediction different from conventional classification. It is certainly possible to treat each of the output variables as separate classification cases; however, as we have illustrated in the previous subsections, recognising and exploiting the correlations between their values can be expected to yield better results.

The above means that learning for structured prediction involves more than only learning the relations between input features and output values. It is at least as important to learn the relations among the various output variables, since that is what allows for joint prediction of their values. Learning approaches specifically aimed at structured prediction should somehow incorporate these extra modelling requirements in their learning and classification strategies. At the same time, they should ensure that the learning task remains feasible. For example, learning the relations between all input features and all output variables, as well as the relations between all output values can become infeasible, especially if the number of output variables is not restricted, as for example in sequence labelling, where the number of output variables equals the length of the input sequence.

Existing structured prediction approaches each provide their own solutions to these issues. The next section provides an overview of the most important of those approaches.

### 2.3 Existing work

Complex structures are so ubiquitous in natural language processing that it is almost difficult to find processing tasks that do not have to deal with structured prediction in one way or another. For this reason, early approaches to what is now known as structured prediction can be traced back to the early days of machine learning for natural language processing. The classic example is NETtalk (Sejnowski and Rosenberg, 1987), a neural network that was trained to convert written English words to their pronunciation in terms of phoneme sequences. Both
input and output have a complex, sequential structure, although the mapping of letters to phonemes clearly has relatively local properties. In NETtalk, the structured prediction problem was solved by sliding a window over the English input word, at each position focusing on just one letter of the word, and predicting the phoneme corresponding to that letter. The output sequence is constructed simply by concatenating all predictions.

Notwithstanding several obvious flaws in the approach, NETtalk’s sliding window approach has become standard in all sequence processing tasks. More generally even, the intuition that complex inputs and complex outputs can be broken apart into smaller windows on the complete structure and classified on a more local level is at the basis of many if not all modern structured prediction techniques. As Dietterich (2000) observes, this strategy can actually be seen as an application of the divide-and-conquer paradigm. Most existing approaches to structured prediction predict an output structure in three steps: (1) divide the original problem (input structure) into smaller subproblems, (2) solve (conquer) the subproblems, and (3) merge the solutions (predictions) for the subproblems into a global solution for the original problem.

Looking at NETtalk, we see that the division step is implemented by the sliding window method, and that the conquer step corresponds to a classification with the neural network. The third step, merging the local predictions into the global output structure, is a simple concatenation of the locally predicted phonemes. In other words, NETtalk blindly follows the decisions of the local classifier without considering whether those make sense in the context of the other predicted phonemes. This way, it ignores the correlation among output variables, which we have seen in Section 2.2 to be defining for structured prediction problems. Improved ways of merging local classifications into a globally optimal solution can be seen as the main contribution of many of the recent approaches to structured prediction.

The remainder of this section is spent giving an overview of the most important methods for structured prediction. Since machine learning tasks in which structure plays an important role have been around longer than approaches specifically aimed at structured prediction, much work exist that could be said to perform structured prediction. Here, we restrict ourselves to those methods that can be applied to a broad range of structured prediction problems, rather than only to a single task that happens to have a structured output space.

### 2.3.1 Recurrent architectures, pipelines, and Searn

As explained in Section 2.2, for good structured prediction, it is important not only to base a prediction on features of the input, but to look at other parts of the output as well. Suppose we are performing a sequence labelling task, such as part-of-speech tagging. A typical strategy would be to classify words from left to right, thereby building the output sequence incrementally. When classifying a word, features are extracted from the input sentence. If those are the only features
the learning algorithm has access to, though, the resulting classifier will not be able to relate its current prediction to other parts of the output structure. This is in fact the approach of NETtalk, which we have explained to be suboptimal for structured prediction.

An extension to this strategy allowing for better structured predictions also extracts features from the partial output sequence predicted so far, i.e. all part-of-speech tags corresponding to the words to the left of the current word. For example, two features corresponding to the two preceding part-of-speech tags are added to the feature vector for a word. Other features may be added that signal whether or not a verb has been seen in any of the preceding words. In general, this strategy can be described by stating that the feature extraction process has access not only to the complete input, but also to a history of previous predictions. A sequence prediction approach that extracts parts of its features from its prediction history is referred to by Dietterich (2002b) as a recurrent sliding window strategy.

Recurrent sliding windows have long been a popular strategy for sequence labelling; among others, Daelemans et al. (1996), Ratnaparkhi (1996), and Kudo and Matsumoto (2000) describe systems based on this approach. In no way is the approach limited to sequence labelling, however. Nivre et al. (2004) present an algorithm for data-driven dependency parsing that also uses history-based features. More specifically, the predictions are driven by a deterministic shift-reduce parsing algorithm. It is made deterministic by having a classifier decide on the next parsing action. Making those decisions, it looks at features extracted from the input sentence and from the parsing stack, which contains partial parses built by the algorithm so far.

We use the term recurrent architecture to refer to an approach where a single classifier incrementally predicts a complex output structure, and while doing that, has access to its own previous predictions. The pipeline model described in Section 2.2.5 does not fit this definition, yet the general strategy is not so different. In recurrent architectures, features are extracted from previous predictions of the same classifier; in pipelines, some features given to one classifier have been predicted by another classifier, earlier on in the pipeline. This similarity makes that most of the following is also applicable to pipeline models.

Inference

Recurrent architectures improve structured prediction by taking into account the partial output predicted so far. While this is a substantial improvement compared to ignoring other parts of the output altogether, the fact that the remaining part of the output is ignored may still lead to suboptimal predictions in some cases. For example, Dietterich et al. (1995) report that for performing English letter-phoneme conversion, predicting the phoneme sequence from right to left with a recurrent sliding window actually leads to better results than a more conventional left-to-right moving window. For the English letter-phoneme conversion task, phonemes to the right of the current prediction appear to be more informative
than those to the left. Reversing the order of prediction, in this case, ensures that the most informative output labels are available as features. Ideally, one would like to take into account the output labels on both sides of the current prediction, though.

Instead of reversing the direction of the sliding window, Ratnaparkhi (1996) describes an extension to the recurrent sliding window strategy that integrates a search algorithm in the prediction process. So far, we assumed that partial outputs are extended by deterministic classifier predictions. In other words, once the classifier has predicted a certain part of the output structure, this decision is never revisited. Another way to look at this approach is as performing a greedy search through the output space. Replacing the greedy search by a more thorough search algorithm allows to postpone the actual prediction of an output label until some or even all future predictions have been made.

Many classification methods do more than just predicting the most-likely class for an instance. Rather, they generate a distribution of output classes. A conventional recurrent sliding window strategy only uses the highest-ranked class in this distribution, but the deterministic prediction can also be turned into a non-deterministic one by using a non-greedy search algorithm. The classifier then predicts each class with a certain confidence, and a search state—a partial output structure—is scored in terms of the confidences of all predictions that led to that state. If the remainder of an output sequence suggests that a lower-ranked prediction is actually the best one in the context of the rest of the sequence, the non-deterministic classification process can still prefer that prediction over the one that was actually ranked highest.

In this setup, any search algorithm can be used, though the choice is not an arbitrary one. Depending on the assumptions made by the algorithm, the base classifier should adhere to certain restrictions. For example, if a Viterbi algorithm with a first-order Markov assumption is used, the base classifier can only extract history features from the decision directly preceding the current one. Non-dynamic programming algorithms, such as beam search or A* search do not have such restricting assumptions.

**Training**

The prediction process of recurrent architectures is essentially nothing more than a sequence of multi-class classifications. In fact, the only difference between a recurrent architecture and NETtalk’s architecture is the use of features that encode part of the prediction history. Therefore, training for recurrent architectures can be as simple as employing conventional multi-class learning algorithms. Training examples are created by simulating the prediction process, albeit with a classifier that always predicts the correct class. As a result, the history features in the training data reflect the true value of those features.

Recurrent architectures trained according to the above procedure are an attractive option for structured prediction. They manage to achieve fairly good per-
formance, and—given that the method works with standard multi-class classifiers—are easy to implement. On the downside, however, several studies have found this training strategy to be vulnerable to an issue known as the label bias problem. It arises when history features are so informative that, if their correct value is known, other features are almost unnecessary for predicting the correct class. The learning algorithm, which is only given the correct values of those features, will overestimate the importance of history features. In the extreme case, previous predictions completely override the information carried by the input, which can cause problems if those previous predictions are incorrect. Bottou (1991) first observed the label bias problem in recurrent architectures. Pipelines may also suffer from this same phenomenon; Van den Bosch (1997) describes the negative consequences of it as error propagation.

Label bias can only be prevented by altering the way that base classifiers are trained. Cohen and Carvalho (2005) report that adding noise to the history features in the training data, causes a maximum-entropy learner to downweight those features, resulting in a reduction of the number of errors. While this strategy does prevent the learner from overfitting on the history features, it also distorts valuable training data. For pipelines, Van den Bosch (1997) proposes adaptive training: rather than learning from data that contains the true values for features for the output of previous processing steps, adaptive training learns from the values that are actually predicted by those previous steps.

Searn (Daumé III, 2006) is a framework for structured prediction that can be seen as a generalisation of adaptive training that is applicable to recurrent architectures. The training examples that the classifier is trained on are iteratively improved so as to better reflect the value of the history features that the classifier will encounter during prediction. In any given iteration, the history features of a training example are generated by a policy. At first, an optimal policy simply produces the true value of the history features—this is equivalent to standard multi-class learning for recurrent classifiers. In subsequent iterations, the policy of the previous iteration is interpolated with a classifier trained on the data generated in the previous iteration. Daumé III reports that for four different sequence labelling tasks, optimal performance is reached in 5 to 15 iterations.

2.3.2 Structured linear models

Linear models are a family of classification models that subsume popular machine learning methods such as perceptrons, support vector machines, and maximum-entropy models. In their standard form, they are restricted to performing binary classification tasks, in which \( y \in \{-1, +1\} \). It is assumed there exists a mapping \( \psi : X \rightarrow \mathbb{R}^n \) that embeds input objects into an \( n \)-dimensional feature space. Essentially, the model is a hyperplane in feature space that separates positive instances from negative ones. Thus, for classification one takes the sign of the dot product...
product of the feature vector and this hyperplane.

\[ \hat{y} = \text{sgn}(\langle w, \psi(x) \rangle) \]

Although, as said, standard linear models can only perform binary classification, they can be generalised, so that they can also be applied to learning tasks where the output space is larger than two classes. This is the case, for example in multi-class classification, where \( y \in \{1, 2, \ldots, N\} \). In such generalised linear models, the feature map is a joint map \( \psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^n \) over both input and output, and the classifier predicts the class \( y \) that maximises the value of the discriminant function.

\[ \hat{y} = \arg \max_y \langle w, \psi(x, y) \rangle \]

From multi-class classification, it is only a small step to structured prediction. Conceptually, at least. Rather than a scalar value, the output value is now a vector.

\[ \hat{y} = \arg \max_{y \in \mathcal{Y}_1 \times \mathcal{Y}_2 \times \ldots \times \mathcal{Y}_n} \langle w, \psi(x, y) \rangle \]

Enumerating and scoring every possible solution in the output space is intractable. Fortunately, that is only necessary if every pair of output variables were interdependent. If, however, interactions are known to exist only between some output variables, more efficient algorithms may be used for performing the maximisation.

Lafferty et al. (2001) propose the use of graphical models as a compact probabilistic formalism to represent the dependencies between the output variables explicitly in terms of an independence graph. Such an independence graph \( G = (V, E) \) is an undirected graph composed of nodes \( v \in V \) that correspond to input and output variables. The edges \( e \in E \) encode the direct conditional dependencies between variables, that is, if two variables \( v_i \) and \( v_j \) are conditionally independent, then \( (v_i, v_j) \notin E \). From this local independence information encoded in the graphical model, more global independence assumptions can be derived. Most importantly in the context of structured prediction, given the values of its neighbours, the value of a random variable \( v_i \) is conditionally independent of all other variables in the graph.

Now consider a subset of variables \( C \subseteq V \), such that for every pair of variables in \( C \) there is an edge connecting them. In graph theory this is called a clique. According to Hammersley and Clifford (1971), the score function of a solution candidate can be decomposed in terms of the cliques of the independence graph.

\[ \hat{y} = \arg \max_{y \in \mathcal{Y}_1 \times \mathcal{Y}_2 \times \ldots \times \mathcal{Y}_n} \sum_{c \in C(G)} \langle w, \psi(x, y_c) \rangle \]

(2.1)

Here, \( y_c \) is the vector corresponding to the output variables indexed by the clique \( c \), and \( C(G) \) returns all cliques of the graph \( G \).
Chapter 2. Machine learning for structured prediction

Inference

The relevance of the clique-based decomposition of structured linear models is that the costly enumeration of the structured output space is not necessary. Scores can be computed for each clique separately, and output structures are built from those cliques. Still, inference in general independence graphs is computationally hard. Nevertheless, for certain classes of graphical structures, efficient inference algorithms do exist. Such is the case if the independence graph has a chain structure, which is typically the case for sequence labelling tasks. Then, the Viterbi algorithm is a polynomial-time algorithm for optimal inference. For tree-structured output objects, dynamic programming parsing algorithms can be used for inference.

As said, for general graphical structures, no polynomial-time algorithm exists for exact inference. The junction tree algorithm (Lauritzen and Spiegelhalter, 1988) performs exact inference in general independence graphs. In addition, many approximate algorithms have been developed or used for inference in general graphs, such as loopy belief propagation (Pearl, 1988) and Gibbs sampling (Geman and Geman, 1984).

Training

Regardless of whether a linear model is trained for binary classification, multi-class classification, or for structured prediction, the goal of the training algorithm is to find a good hyperplane \( w \). What exactly constitutes a good hyperplane is a question that cannot be answered uniformly. Therefore, various different learning algorithms exist for finding such a hyperplane. What is actually considered a good hyperplane may differ for each algorithm. For example, logistic regression searches for a hyperplane that maximises the likelihood of the training data, while support vector machines maximise the margin between the hyperplane and the training examples. For now, we ignore such learning strategy-specific issues and continue treating the general class of linear models.

Interestingly, the reformulation of linear models for multi-class classification, as well as for structured prediction, does not require major changes to the learning algorithms. Any algorithm that can be used for training binary linear classifiers can also be applied in this generalised linear setting. Support vector machine training has been adapted for the multi-class domain by both Weston and Watkins (1998) and Crammer and Singer (2001), who propose two slightly different alternative formulations. For support vector machines for structured prediction, also two alternative formulations exist (Taskar et al., 2004; Tsochantaridis et al., 2005). Maximum-entropy models (Berger et al., 1996) are a multi-class version of logistic regression, and conditional random fields (Lafferty et al., 2001) generalise the model for structured prediction. Collins (2002) generalises perceptron training for both multi-class classification and structured prediction.

An important property of all training algorithms for structured linear models
is that they require inference during training. Most training algorithms make
several passes through the training data, and at each pass, each instance has
to be classified, and thus, inference has to be performed for each instance. In
addition, depending on the exact training objective, additional statistics have to
be collected. Logistic regression optimises the likelihood of the training data.
Therefore, it needs to turn scores into probabilities. This is done with a softmax
function, which requires a normalising constant, the partition function, that equals
the sum of scores of all output structures. Support vector optimisation aims
to maximise the difference in score between the true output and the highest-
scoring incorrect output. Therefore, the training algorithm needs to perform
2-best inference. Even in the case where reasonably efficient inference algorithms
exist, the need for inference during training remains a severe penalty for structured
linear models.

2.3.3 Constraint satisfaction with classifiers

Structured prediction is concerned with predicting the values of several interde-
pendent variables. Recurrent architectures and structured linear models try to
learn those dependencies from data. Often in natural language processing, the
actual types of dependencies between variables come down to constraints on the
values that variables have simultaneously. If that is the case, recurrent architec-
tures and structured linear models learn those constraints implicitly. However,
if the types of constraints on the output structure are known in advance, it is
wasteful trying to learn them; one could also choose to enforce these constraints
explicitly on the classifier predictions.

As an example, consider a sequence segmentation task such as named-entity
recognition. Typically, named-entity recognition is performed as a sequence la-
labelling task predicting \texttt{BIO} codes for the words in a sentence. In this coding
scheme, \texttt{B-PER} signals the beginning of a reference to a person entity, \texttt{I-PER}
the continuation of such a segment, and \texttt{O} a word outside a segment. According
to this scheme, it is illegal to predict a subsequence of tags \texttt{O I-PER}, since segments
should always start with a \texttt{B-} code. Another example of structural constraints, in
the context of a pipeline architecture for entity recognition and relation extrac-
tion, is found in a study of Roth and Yih (2004). They observe that semantic
relations between entities are often restricted in the types of entities they take.
Examples of this in their study include a \texttt{killed} relation that involves two person
entities, and a \texttt{lives_in} relation that covers a person and a location entity.

Constraint satisfaction with classifiers (CSCL) (Punyakanok and Roth, 2005;
Punyakanok et al., 2005) is a framework for structured prediction that employs
constraint satisfaction techniques in combination with explicitly formulated struc-
tural constraints, such as the above examples, that model the dependencies be-
tween output variables. Local classifiers estimate the actual values of the various
output variables, but are mostly oblivious to the structural constraints. These
are dealt with separately, in an inference step that follows classification.
Chapter 2. Machine learning for structured prediction

Inference

Inference in the CSCL framework faces the task of finding the output structure that satisfies the constraints and at the same time is maximally likely given the predictions made by the classifiers. To this end, the inference step is formulated as a constraint satisfaction problem. This ensures that any solution that is considered is a valid solution according to the domain constraints. The specific constraint satisfaction formalism used in CSCL associates scores with variable-value assignments. In short, the inference step performs the following optimisation.

\[
\hat{y} = \arg \max_{y \in C(y)} \sum_{i=1}^{n_y} f_i(x, y_i)
\]  

(2.2)

The search for solutions is restricted to those parts of the output space \( y \in C(y) \) that satisfy all constraints. Local score functions \( f_i: X \times Y_i \rightarrow \mathbb{R} \) give a confidence estimate for the assignment of a certain value to each one of the output variables given the input. These scores are extracted from the classifier predictions.

Unrestricted combinatorial search for solving this problem is computationally intractable. For sequential outputs, Punyakanok and Roth (2001) reduce the constraint satisfaction problem to a shortest path problem in a directed graph. This problem is solvable in polynomial time. More general output spaces can be dealt with by converting the constraint satisfaction problem to an integer linear program, for which high-performance solving algorithms are readily available (Roth and Yih, 2004).

Training

As shown in Equation 2.2, the global score function of CSCL is decomposed into local score functions. These local score functions correspond to the confidence estimates of multi-class classifiers that are trained to predict the value of a single output variable. The exact details of the training procedure of those classifiers leave open two strategies. One can choose to train the classifiers without considering the role of inference and the constraints enforced in it, or inference can already be taken into account while training. Punyakanok et al. (2005) compare these two training strategies for CSCL. They refer to the former strategy as learning plus inference (L+I), and the latter as inference-based training (IBT).

In the learning plus inference strategy, training aims to minimise the loss on individual predictions of output variable values. This is equivalent to standard multi-class learning, and thus, standard learning algorithms for multi-class classification can be used.

In the inference-based training strategy, not the quality of individual predictions but that of the global structure is optimised. Therefore, the constraint satisfaction based inference procedure is performed during training, in the same way as for structured linear models, and classifiers are trained in such a way that
they perform well in the context of inference, regardless of whether they would do so as well without inference. Punyakanok et al. implement this strategy with an adapted perceptron algorithm, which only updates parameters if the predictions after inference are incorrect.

The study finds that both strategies have their merits depending on the exact characteristics of the learning task. If the hypothesis space of the local classifiers is broad enough to successfully learn the target class, L+I outperforms IBT. On the other hand, if the local classifications cannot be learned accurately by the local models, IBT may perform better than L+I; however, sufficient amounts of training data is crucial for that.

2.3.4 Output kernels

Whereas structured output is still largely an open research problem, reflected in the large diversity of approaches being applied to it, there is a remarkably broad agreement on how to treat structured inputs: by using kernels. Essentially just a method to enable linear classifiers to learn tasks that are not linearly separable, kernels are often used to map arbitrary input structures to a high-dimensional feature space, where it can be dealt with by linear models. Formally, a kernel is a function \( k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) implicitly computing the dot product of its two arguments in some high-dimensional feature space. The result of that dot product can be interpreted as measuring the similarity between the input arguments. As a consequence of this, we can express the similarity between input types having any arbitrary structure, as long as we can construct a mapping—defined by a feature map \( \Phi : \mathcal{X} \rightarrow \mathbb{R}^F \)—of the input structure to an \( F \)-dimensional feature space.

Particularly relevant for machine learning for structured data are several kernels that have been defined over structured domains. These kernels include n-gram kernels (Leslie et al., 2002; Paass et al., 2002), which map sequences to feature space, tree kernels (Collins and Duffy, 2001), for operating on trees, and convolution kernels (Haussler, 1999). Interestingly, the latter type of kernel is a rather general framework for constructing kernels over structured data; it describes how a kernel for a structured object can be defined in terms of kernels over its parts. Having an appropriate kernel for the type of input structure of a learning problem, kernelisable linear models can easily be used for processing the structured input objects.

Although best-known for their ability to learn with structured inputs, kernels have also been applied as a method for predicting structured output. To this end, the structured target values of a learning problem are mapped into feature space through application of the feature map of the appropriate kernel. The result of this mapping is a numeric vector, representing the original structured output. Unlike the original output value, however, we can now employ conventional learning algorithms for multi-variate regression to learn a mapping from inputs to the feature representation of the structured output.

The application of kernels for predicting structured data was first explored by
Weston et al. (2002) as part of the kernel dependency estimation (KDE) framework. Their work has been further elaborated upon by Cortes et al. (2005), who describe a simpler formulation of KDE, and also explain how input-output constraints can be integrated in the approach. Along similar lines, Geurts et al. (2006) adapt regression tree learning for kernelised output, resulting in output kernel trees (OK3). Szdemak et al. (2005) use an output kernel approach to hierarchical classification.

Pre-image search

The regression models trained in the above studies map input instances to a vector representation of the output structure. This vector representation is the mapping of the true output structure in feature space, and as such is not the desired end result. Therefore, the remaining step is to reconstruct the output structure from its representation in feature space. This is called the pre-image problem.

If the feature map of the kernel is invertible, solving this problem is trivial. Schölkopf and Smola (2002) show that this is the case for polynomial kernels of odd degree, sigmoid kernels, and RBF kernels. In the general case, though, there may not always be a closed-form solution. Even more problematic, there is not even the guarantee that every vector in the feature space corresponds to a valid structure in the output space. This will especially be the case with the mostly discrete output spaces of structured prediction. Only a part of the continuous feature space has an actual pre-image, and regression errors may cause a vector without such a pre-image to be predicted. Rather than finding the exact pre-image, the goal is changed to finding an approximate pre-image. Then, the pre-image search problem is to find the value $y$ in the output space that, when mapped to feature space, is closest to the predicted output vector.

$$\hat{y} = \arg \min_{y \in Y} \| \Phi(y) - f(x) \|^2$$  \hspace{1cm} (2.3)

One way of solving this is through general numerical optimisation (Burges, 1996). This approach is rather expensive, though, and also risks only finding a local optimum of the above equation. Alternatively, Bakır et al. (2004) propose learning to reconstruct pre-images from their vector representation in feature space. Unlike the previously mentioned approach, this method is also applicable to problems where the pre-image is a discrete structure. As a critical precondition for this approach, it has been found that the distribution of pre-images in the training data should reflect the distribution in the test data. Cortes et al. (2005) formulate the pre-image problem for the $n$-gram kernel as finding an Euler circuit in a graph, which enables a linear-time solution to the pre-image problem.
2.3. Existing work

Training

The use of kernels makes that complex output structures are mapped to numeric vectors. Because of that, the learning task is that of standard multi-variate regression, for which many learning algorithms exist. Examples are kernel ridge regression (Saunders et al., 1998), support vector regression (Vapnik, 1995), and kernel matching pursuit (Vincent and Bengio, 2002). Importantly, learning is completely decoupled from pre-image search. That is, in order to learn the regression model, no feedback of the pre-image search is required. This is different from structured linear models and CSCL with inference-based training, where inference is a crucial component of training, and as such may turn into a severe bottleneck.

In spite of the advantage of not requiring pre-image search during training, training for output kernels risks to be costly in other aspects. Depending on the choice of regression technique, learning might require many expensive matrix inversions, or may not end up with sparse solutions. For the latter problem, Cortes et al. (2005) propose a greedy approximation of kernel matching pursuit.

2.3.5 Discussion

Our survey of structured prediction approaches shows that there are both similarities and differences in the way the approaches tackle the structured prediction task. Here, we summarise the most important of them, and with these observations in mind, revisit the objectives formulated in Chapter 1.

The most apparent commonality among the approaches is that most of them require the optimisation of a global score function during prediction (as illustrated by Equations 2.1, 2.2, and 2.3). In structured linear models, this takes the form of inference on the underlying graphical model; constraint satisfaction with classifiers solves a constraint satisfaction problem to arrive at the best prediction; and output kernel approaches need to perform a pre-image search to find the output structure of which the feature space mapping is closest to the predicted vector. Since the typical output space of structured prediction tasks has a size exponential in the number of output variables, this inference step, in its most general form, is computationally hard. In many cases, though, appropriate assumptions can be made that enable the inference problem to be solved in polynomial time. Nevertheless, this remains a serious bottleneck. Recurrent architectures can also be extended with an inference step, but can also function without. Moreover, Searn is claimed to remove the need for inference during classification.

Whereas inference is performed during prediction in most of the approaches, more differences exist as to whether or not inference is part of the training procedure as well. Training of structured linear models always involves inference. As explained in Section 2.3.3, constraint satisfaction with classifiers may be used with both types of training, but depending on the characteristics of the learning task, one may be preferable over the other. Output kernel approaches need a pre-
image search as part of prediction, but for learning the regression model, no such step is necessary. Finally, recurrent classifiers do not necessarily need inference for prediction; however, if Searn is used for training the classifier, then training does involve several iterations in which full structured prediction of the training data is performed.

To conclude, we recapitulate the requirements that the various approaches put on the underlying learning techniques. All frameworks discussed in this section can be seen as extensions of more traditional learning methods to the structured domain, whether they be based on classification or regression. Still, some of them allow for more flexibility in the choice of the underlying learning method than others. Structured linear models, by definition, hinge on linear classifiers as the underlying classification model. This model class still allows for many different learning trade-offs, yet many other learning frameworks are incompatible with the approach. Output kernel approaches are based on regression rather than classification models; however, given this restriction, any regression method that can learn to predict vector output values is usable. Recurrent classifiers, including Searn-trained classifiers, and the constraint satisfaction with classifiers framework are compatible with any multi-class learning method. If no inference is performed, the former can even function without confidence estimates.

2.4 Conclusion

In this chapter, we have elaborated on some of the concepts touched upon briefly in Chapter 1. We have explained how many processing tasks in natural language processing have an output space that is outside the scope of conventional classification techniques. Until recently, most of those tasks were nevertheless performed using those conventional classification techniques, mostly following the prototypical NETtalk strategy of predicting the parts of the complex output separately. In recent years, special-purpose structured prediction approaches have emerged and managed to improve significantly on the performance of those traditional approaches. The survey of those approaches given in this chapter is by no means exhaustive; however, the methods presented do provide good insight into the central issues that structured prediction involves.

Our discussion of existing approaches to structured prediction helps putting the three research objectives formulated in Chapter 1 in perspective. Structured prediction methods face an output space of exponential size in the number of output variables. Therefore, exhaustive search for finding the optimal output structure is a realistic option with small numbers of output variables only, yet typical natural language processing tasks involve many output variables. The inference phase will have to deal with this in order to ensure efficient predictions. We saw that making restricting assumptions is a way to ensure efficient inference. A potential disadvantage of such assumptions is that they limit the types of dependencies that can be modelled, and as a result, may complicate the
learning task. Our goal is to attain efficient inference, without making restricting assumptions.

The complexity of training procedures for structured prediction models may also be affected by the magnitude of the output space. This is the case for approaches that employ inference-based training, since inference is performed, possibly several times, for all training examples in order to learn the prediction model. Inference-based training is a requirement for some methods, such as structured linear models and Searn, while others can do without it; for example, the constraint satisfaction with classifiers framework and output kernel approaches. Our objective is to decouple inference from training.

Finally, as already discussed, some approaches restrict the range of underlying classifiers that can be used with it. Because of this, some machine learning methods that have been shown to be successful in learning linguistic processing tasks may nevertheless become irrelevant in the context of structured prediction problems. Our final objective states that our approach should be independent of the type of underlying classifier.
Chapter 3

Structured prediction as constraint satisfaction

3.1 The anatomy of a structured prediction problem

If nothing is known about the internal structure of a complex output value, the prediction problem can only be treated as one gigantic multi-class prediction problem. While learning algorithms exist that can scale fairly well to large numbers of classes, this approach will ultimately break down due to sparsity issues. In many cases however, the value to be predicted is not simply a black box, but known to be built up from smaller parts, the composition of which shows some degree of regularity. A part-of-speech tag sequence, for example, may simply be regarded as one atomic object without any internal structure, but there is an obvious way to split up that large sequence into separate part-of-speech tags, each of which can be linked to one of the words in the input sentence. If such subdivision of the complex output value is possible, structured prediction methods can be devised that predict the output structure based on predictions of those smaller parts.

In order to develop a general framework that will be applicable to many different structured prediction scenarios, it is necessary to use a representation that abstracts away from the many surface forms of structured predictions, i.e. a representation that is uniform, no matter whether a label sequence, a tree structure, or something even more complex is predicted. For this purpose, we adopt the terminology introduced by Altun (2005).

The macro-label $\mathbf{y} \in \mathcal{Y}$ is the top-level view of the structured object. When performing part-of-speech tagging, the macro-label of a sentence is the complete sequence of part-of-speech tags. If the task is syntactic parsing, the macro-label of that same sentence may be a tree, e.g. a dependency or constituent structure. The macro-label of an input is the closest match to the conventional class label in multi-class classification, i.e. for each input object, there is one (macro-)label,
and two (macro-)labels are either the same or completely different. As noted before, only dealing with a structured prediction problem on the macro-label level does not allow for structured prediction solutions; the only remaining option is conventional multi-class classification. However, macro-labels are different from the conventional class labels of multi-class classification; they have an internal structure. The subparts of a macro-label are referred to as its micro-labels. Again taking the example of part-of-speech tagging, the complete part-of-speech tag sequence is the macro-label, whereas the individual part-of-speech tags are the micro-labels.

The terms macro-label and micro-label make it possible to refer to the complex surface structure (macro-label), as well as to the internal structure and its subparts (micro-labels) using a uniform terminology. To unite these two views, the macro-label \( y \) is defined to be a vector \( (y_1, \ldots, y_n) \in Y_1 \times \ldots \times Y_n \). The vector elements \( y_i \) then correspond to the micro-labels. Note that, for the sake of generality, viewing a macro-label as a vector is deliberately left abstract. It may be trivial to think of a label sequence as a vector, but it may be less so to think of a tree as such. Nevertheless, it is assumed that every complex object structure can be mapped to a vector representation. As part of this mapping, it may be necessary to define the relation between certain vector elements. This mapping however, is task specific and will not be specified any further in this introduction.

### 3.2 A toy problem: multi-label text categorisation

In this chapter, the constraint satisfaction inference (CSI) framework for structured prediction is introduced. Before formalising the terminology that defines the framework, the intuition behind the method is introduced stepwise by illustrating the application of constraint satisfaction inference to a toy problem. Unfortunately, given the inherent complexity of structured prediction problems, real toy problems for structured prediction are either non-existent, or too artificial and constructed to be of explanatory interest. That being said, multi-label classification may arguably be considered one of the easier classes of prediction tasks with a structured output space. For illustrating our new approach to structured prediction, we therefore chose a multi-label text categorisation scenario. Referring to this as a toy problem does not do right to its relevance. In fact, it is has been the focus of numerous scientific studies (e.g. Elisseeff and Weston, 2002; Boutell et al., 2004; Ghamrawi and McCallum, 2005). The goal in the context of this chapter however, is not to optimise constraint satisfaction inference for attaining new state-of-the-art results, but rather to explain the concepts that form the basis of the constraint satisfaction inference method.

In text categorisation, the content of a textual document is analysed and a relevant category label is predicted that reflects the topic of the document according to a certain taxonomy. The potential applications of text categorisation are
3.2. A toy problem: multi-label text categorisation

numerous, for example spam filtering and managing news feeds. In the former example, only two categories exist, denoting whether or not a message is spam; in the latter, many more or less fine-grained categories may be distinguished, for example, politics, science, sports, etc. If category labels are mutually exclusive, and thus a document can have one category only, the categorisation task is an instance of standard multi-class classification. It is easy to see that in the spam filtering example, categories should be mutually exclusive. Often, however, mutually-exclusive text categories are too restrictive. One could easily imagine this to be the case for the news feed example, where the boundaries between categories are not always clear-cut. Multi-label text categorisation deals with the situation where a document may actually belong to more than one category, and the task is to select the subset of category labels that are relevant to the document. Relaxing the mutual-exclusiveness of category labels moves text categorisation from multi-class classification to structured prediction, which makes it part of the application domain of constraint satisfaction inference.

One of the standard text collections for text categorisation research is the Reuters News Corpus for Text Classification (Reuters-21578). The corpus comprises a collection of 21,578 Reuters newswire stories that have been labelled with 135 topic labels. The ModApte split is a popular subset of the full Reuters corpus that considers only the 10 most frequent topics, and consists of only those documents that belong to at least one of those topics. The data set thus obtained has 8,242 documents: 5,924 for training, 2,318 for testing. The vast majority of documents in this subset has been labelled with a single topic; yet, documents exist with up to 4 topic labels. Hence, topic labels are not mutually exclusive, which makes the categorisation task a multi-label classification task.

Table 3.1 lists the 10 most frequent document topics that define the ModApte split. As said, most of the documents in this subset are labelled with only one of those topic labels. Given our focus on structured prediction, we are mostly interested in those documents that have more than one topic. Table 3.2 shows the distributions of topic counts per document for both the training and test set. In both subsets, approximately 10% of the documents have been labelled with more than one topic. Looking further at how many combinations are formed with the 10 topics, we found that in the training data, there are 39 different macro-labels, while in the test data, there are 27 different macro-labels. Interestingly, 4 of those 27 macro-labels do not exist in the training data.

In the next section, we describe experiments with various approaches to the Reuters multi-label text categorisation task. These experiments will show that explicitly modelling the structural properties of the output space of multi-label classification is key to good performance on the task.
Table 3.1  List of the ten most frequent topics in the Reuters-21578 text classification data. The ModApte split, used for the experiments in this chapter, is a subset of the full corpus that only includes documents that are labelled with at least one of these topics.

<table>
<thead>
<tr>
<th># Topics</th>
<th>Training documents</th>
<th>Test documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5,346</td>
<td>2,123</td>
</tr>
<tr>
<td>2</td>
<td>505</td>
<td>170</td>
</tr>
<tr>
<td>3</td>
<td>68</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.2  Distribution of the number of topics assigned to a document in the training set and in the test set.

3.3 Towards constraint satisfaction inference for multi-label classification

Instead of directly presenting the constraint satisfaction inference approach to multi-label classification, the following subsections will present a series of approaches to the task. The first of those are relatively straightforward applications of conventional classification techniques. We analyse the flaws of these simpler approaches, and based on that, work towards constraint satisfaction inference, a framework intended to resolve those flaws. This style of presentation shows how constraint satisfaction inference differs from more straightforward approaches, but will also illustrate how constraint satisfaction inference attempts to keep what is good about the simpler approaches, and only increases complexity where necessary.

As a first step towards a structured approach to multi-label text categorisation, the prediction task is formulated in terms of the terminology introduced in Section 3.1. Intuitively, multi-label classification is thought of as predicting a number of labels for one input object. Formally, though, a structured prediction approach assumes the output to be a single, be it complex object: the macro-label. A document that has been categorised as trade as well as wheat has a single macro-label, which encodes that those two categories are relevant, while all other categories are not. Similarly, for a document that actually only has one category, interest, the macro-label expresses that interest is a relevant category, while all other categories are not. With this definition, multi-label classification is defined in terms of one single output object.

The internal structure of the macro-label is defined by the micro-labels, which
in this case, straightforwardly reflect the topics. Each micro-label $y_i$ can be seen as a boolean variable that signals the presence or absence of the $i$th topic in the topic set of the document, and thus $Y_i = \{0,1\}$. The macro-label, then, is a bit vector covering ten micro-labels, one for each topic: $Y = Y_1 \times \ldots \times Y_{10} = \{0,1\}^{10}$.

Given the macro-label and micro-label views of the output as just defined, two approaches to our multi-label text categorisation task present themselves almost automatically. The first is to train a binary classifier for every micro-label; the second is to train one multi-class classifier for the complete macro-label. Both approaches are still within the domain of conventional classification, and are not expected to perform optimally. On the other hand, it is insightful to see how far conventional classification techniques bring us, and then introduce more structured strategies to improve on that. Experiments with these two approaches are presented in the next two subsections, after which the lessons learned from the experiments lead us to propose a novel approach, which is introduced in the remaining subsections.

To perform the experiments of the following subsections, several classifiers need to be built that map input documents to some kind of multi-class output space, differing for each of the experiments. Since the structured multi-label predictions will be constructed from the output of these classifiers, the latter will be referred to as base classifiers. Following one of the objectives of this study, the structured prediction framework presented in this chapter allows for any machine learning algorithm to train such base classifiers, as long as it is capable of learning multi-class classification tasks. For the experiments that follow, we will use the $k$-nearest neighbours ($k$-NN) method for training base classifiers—other learning methods would have been possible as well though. The feature set for representing input documents corresponds to a set-of-words representation of the content of the documents from which stop words and words occurring 5 times or less in the complete training data are removed.

### 3.3.1 Micro-label prediction

The most straightforward approach to multi-label classification is to train binary classifiers for each micro-label. A positive prediction of a micro-label classifier sets the corresponding bit in the macro-label vector to 1, a negative prediction sets it to 0. This approach is in fact a common one for classification tasks with classes that are not mutually-exclusive (e.g. Joachims, 1998; Yang and Liu, 1999; Boutell et al., 2004). It is sometimes complemented with a confidence threshold, so that only micro-label classifiers that make a positive prediction with a confidence greater than the threshold cause the corresponding macro-label bit to be set. Tuning this threshold parameter, empirically or analytically, may make micro-label prediction—which is not specifically suited for multi-label classification at all—a bit more suited for multi-label classification. In our experiment, we do not use such threshold tuning, and simply follow the positive or negative predictions of the base classifiers.
Table 3.3 Performance of various machine learning approaches to the multi-label text categorisation task.

<table>
<thead>
<tr>
<th></th>
<th>Macro-$F_{\beta=1}$</th>
<th>Micro-$F_{\beta=1}$</th>
<th>Subset acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro-label</td>
<td>77.59</td>
<td>89.51</td>
<td>84.12</td>
</tr>
<tr>
<td>Macro-label</td>
<td>78.66</td>
<td>89.87</td>
<td>87.06</td>
</tr>
<tr>
<td>Meso-label</td>
<td>78.04</td>
<td>89.82</td>
<td>85.16</td>
</tr>
<tr>
<td>Oracle</td>
<td>82.87</td>
<td>92.12</td>
<td>87.83</td>
</tr>
<tr>
<td>Saboteur</td>
<td>73.98</td>
<td>87.57</td>
<td>82.79</td>
</tr>
<tr>
<td>CSI$_{meso}$</td>
<td>78.82</td>
<td>90.01</td>
<td>84.99</td>
</tr>
<tr>
<td>CSI$_{meso,count}$</td>
<td>79.69</td>
<td>90.40</td>
<td>86.71</td>
</tr>
</tbody>
</table>

To apply micro-label prediction to the text categorisation task, ten different training sets are created, one for each topic. In each training set, documents that have the corresponding topic as one of their topics are labelled as positive instances, while all remaining documents become negative instances. Subsequently, binary classifiers are trained on these training sets and used to predict the micro-labels of the test documents. The results of this experiment are reported in Table 3.3, which lists macro- and micro-averaged $F_{\beta=1}$ and subset accuracy, the percentage of completely correct macro-labels.

With a subset accuracy of 84.12, the performance is already quite high, even though micro-label prediction does nothing to take into account the structural properties of the output space deemed important for multi-label classification. The most likely cause of this is the fact that all ten document topics are sufficiently present in the training data. This only illustrates that, even in structured prediction tasks, well-trained classifiers may be able to make reasonably accurate micro-label predictions without taking into account the values of other micro-labels. Nevertheless, since there may be interaction among topics assigned to the same document, it is likely that micro-label prediction misses crucial information as a result of not taking this interaction into account. Such information may, for example, suggest that if one category is assigned to a document, another category is also likely to apply to that same document. Also, certain categories may still be mutually exclusive, so that two of those categories will never apply to the same document. Micro-label prediction fails to recognise, let alone exploit such interactions. The approach presented next improves on this situation.

3.3.2 Macro-label prediction

The main drawback of micro-label prediction is that it fails to exploit the interactions that exist between the categories that are assigned to the same document. Since the learners that build the micro-label predictors do not learn the ways
3.3. Towards constraint satisfaction inference for multi-label classification

in which categories interact, this is a fundamental deficiency of the approach. A more structured approach should learn the ways in which categories interact, preferably without explicitly being given additional knowledge about the interactions. Macro-label prediction does exactly that; it learns all the interactions implicitly, no matter whether they constrain the possible topic combinations, or only affect their frequencies. The approach is rather simple: learn to predict the macro-labels directly, as if they were atomic multi-class labels. In spite of its apparent simplicity, macro-label prediction has the potential to learn any kind of interaction that might exist between document categories. We illustrate this by discussing two important types of dependencies.

1. Mutually-exclusive micro-labels
   Although topics in the Reuters-21578 training data are not mutually exclusive in general, combinations of topics that never appear in practice do exist. For example, no document exists that has both category labels acq and interest; also the combination of money-fx and ship is never encountered in the data, as are some other combinations. Provided that the training data being used are sufficiently large and that they are representative of the documents tested on, it is a reasonable assumption that unseen test documents will not belong to such combinations of topics either. This observation can be seen as a constraint on the output space, and as such one could use it to extend micro-label prediction, so that the predicted micro-labels adhere to these macro-label constraints. In order to so, these constraints will first have to be made explicit. With macro-label prediction, there is no need to make them explicit. Instead, they are captured implicitly by the approach: if no macro-label where both acq and interest are active has been seen in the training data, a conventional multi-class classifier will not predict one, since it will only consider classes seen during training.

2. Co-occurrence of micro-labels
   Some micro-labels may make the presence of another micro-label for the same document more, or less likely. For example, it may be the case that documents with the topic corn have an increased likelihood of also having the topic grain. Again, to a certain extent, such knowledge could be collected from the data and combined with the output of micro-label predictors to produce better macro-labels, but as it was the case for mutual exclusiveness, that will have to be done explicitly. In contrast, macro-label prediction learns co-occurrence interactions implicitly. If the presence of corn increases the likelihood of grain, the relative frequency of the topic grain among training examples with macro-labels that also contain the topic corn will be higher than its frequency among all training examples. Since learning algorithms for multi-class classification tend to be sensitive to prior probabilities of class labels, co-occurrence frequencies will be learned as a side effect.
To apply macro-label prediction, a single training set is created, in which
documents are labelled with a symbol that is the combination of all topics relevant
to the document. A multi-class classifier is then trained to directly predict the
macro-label of a document. The performance of this approach on the test set is
reported in Table 3.3. It can be seen that the expected benefits of direct macro-
label prediction indeed contribute to an improved performance. With a subset
accuracy of 87.06, and $F_{\beta=1}$ scores also well above those of micro-label prediction,
it clearly outperforms micro-label prediction.

Notwithstanding the advantages of macro-label prediction, there is one flaw
that makes it impractical for general structured prediction. Learning to predict
complete macro-labels with conventional classification techniques inevitably runs
into sparsity issues. As the number of macro-labels may grow exponentially with
the number of micro-labels, many macro-labels will probably occur only a few
times if at all in the training data. Earlier, we mentioned that there exist four
different macro-labels in the test data that do not occur a single time in the
training data. Even for those macro-labels that do have a reasonable presence
among training examples, it may still be difficult to successfully learn the relation
between input features and macro-labels. Macro-labels that are not represented in
the training data at all even pose an insolvable problem for macro-label prediction,
they will simply never be predicted. While for multi-label classification with a
small number of micro-labels the sparsity problem could arguably be fixed with
more training data, this can hardly be claimed to be the case for e.g. part-of-
speech tagging, or syntactic parsing, since there can never be enough training
data to cover every possible macro-label for those tasks.

### 3.3.3 Meso-label prediction

The macro-label prediction experiment indicates that multi-label document cate-
gorisation can be learned as a single multi-class learning task. In fact, macro-label
prediction beats micro-label prediction in terms of classification performance.
However, there is no denying that for many realistic applications, macro-label
prediction ultimately breaks down due to sparsity issues. For the current docu-
ment categorisation task, the worst-case size of the output space is $2^{10} - 1$; it turns
out to be only 39. As could be observed in the macro-label prediction experiment,
this output space is learnable. A particularly favourable property is that the size
of the output space depends on the number of categories only, and thus given the
task, is fixed; the size of the input document plays no role here. Such favourable
conditions do not hold in general, though. Even the still relatively easy structured
prediction task of part-of-speech tagging has an output space that grows expo-
nentially with the length of the input sentence. The large number of class labels
in the training data will already make it difficult to accurately train a classifier
for recognising those labels. An even bigger problem, though, is the fact that
many valid part-of-speech tag sequences are not even in the training data, and
thus, can never be predicted by a macro-label prediction approach.
3.3. Towards constraint satisfaction inference for multi-label classification

<table>
<thead>
<tr>
<th></th>
<th>corn</th>
<th>crude</th>
<th>earn</th>
<th>grain</th>
<th>interest</th>
<th>money-fx</th>
<th>ship</th>
<th>trade</th>
<th>wheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>acq</td>
<td>0.009</td>
<td>0.015</td>
<td><strong>0.251</strong></td>
<td>0.026</td>
<td>0.021</td>
<td>0.033</td>
<td>0.009</td>
<td>0.021</td>
<td>0.014</td>
</tr>
<tr>
<td>corn</td>
<td>0.002</td>
<td>0.024</td>
<td>0.110</td>
<td>0.002</td>
<td>0.002</td>
<td>0.000</td>
<td><strong>0.002</strong></td>
<td>0.022</td>
<td></td>
</tr>
<tr>
<td>crude</td>
<td>0.041</td>
<td>0.004</td>
<td>0.003</td>
<td><strong>0.006</strong></td>
<td>0.001</td>
<td>0.002</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>earn</td>
<td>0.062</td>
<td>0.045</td>
<td>0.073</td>
<td>0.028</td>
<td>0.052</td>
<td>0.030</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>grain</td>
<td>0.005</td>
<td>0.007</td>
<td>0.001</td>
<td>0.002</td>
<td><strong>0.138</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>interest</td>
<td>0.041</td>
<td>0.002</td>
<td>0.001</td>
<td>0.002</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>money-fx</td>
<td>0.004</td>
<td>0.000</td>
<td>0.004</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ship</td>
<td>0.001</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trade</td>
<td>0.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4 Mutual information values of the ten most frequent topics of the Reuters-21578 document categorisation task. The values printed in bold face type correspond to the topic pairs selected for the meso-label prediction approach.

If macro-label prediction is—for all but the simplest of cases—too difficult a task to learn, and micro-label prediction not powerful enough to perform well on structured prediction tasks, it might be worthwhile to try classification on yet another level; a level in between micro-labels and macro-labels. The target values of this approach will be referred to as meso-labels. Where a micro-label corresponds to one element of the output vector and a macro-label is the full vector, a meso-label $y_S$ is a vector of which the elements $S \in 2^{\{1,\ldots,n_y\}}$ are a subset of the elements of the macro-label. In comparison with macro-labels, meso-labels capture fewer dependencies—only those between the micro-labels they cover—but since they are less sparse, it may be easier to learn them accurately.

If document categories are to be predicted jointly, the question arises which labels to predict jointly, that is, which meso-labels classifiers will be trained for. Let us assume for now that overlapping meso-labels, i.e. meso-labels that cover one or more of the same categories, are not allowed. By making this assumption, the problem of what to do with overlapping but conflicting meso-label predictions is circumvented. As yet another simplification, we will only consider meso-labels with two elements. Note that this is not a fundamental restriction, but it will simplify the choice of meso-labels.

As the original motivation for jointly predicting categories was to capture interactions between categories, it makes sense to try to construct meso-labels that cover categories that are conditionally dependent. We estimated the dependencies between document categories by computing the mutual information values of all pairs of document categories, and listed those in Table 3.4. From the values in the table, one can indeed conclude that there is strong interaction between certain document categories. The **acq** and **earn** are most-strongly correlated; in fact, they are almost mutually exclusive. There is also fairly strong interaction between **corn** and **grain**, and between **wheat** and **grain**. Such pairs of categories with high mutual information are logical candidates for meso-labels. The assumption that meso-labels do not overlap prevents us from choosing both
corn+grain and wheat+grain as meso-labels, though. Our selection of meso-labels is the set of topic pairs that has the maximum sum of mutual-information values, while adhering to the non-overlap constraint; these meso-labels correspond to the mutual-information values printed in bold face type in Table 3.4.

Applying the meso-label prediction approach involves the creation of five different training sets, one for each meso-label. Documents in the training set for a certain meso-label are labelled with a symbol that encodes the presence of the two topics covered by the meso-label. The resulting learning task is a multi-class task, with at most four classes. To predict the topics for a test document, the five resulting classifiers are applied, and the macro-label is constructed by a simple combination of the five predicted meso-labels.

The performance of this meso-label prediction approach can be seen in Table 3.3. Comparing it with the performance of micro-label prediction, it is better in terms of all three measures. In comparison with macro-label prediction, though, meso-label prediction falls short considerably, which suggests that meso-label prediction is a worse approach to structured prediction; it most certainly is for multi-label classification. One could argue that meso-label prediction has the crucial advantage that, unlike macro-label prediction, it will scale to larger output spaces, yet the performance difference on this toy problem is too big for this meso-label prediction approach to be considered a serious contender for being a generic structured prediction approach.

The most likely explanation for the superior performance of macro-label prediction is that it apparently captures more label interactions than meso-label prediction does. More specifically, macro-label prediction captures interactions that cannot possibly be covered by meso-label prediction, as a result of the requirement that meso-labels do not overlap. Dropping this requirement might potentially improve the performance of meso-label prediction, but it also introduces a new problem: what to do with meso-label predictions that suggest a different value for the same category? This is, in fact, similar to the inference problem faced by many of the structured prediction approaches presented in Chapter 2, i.e. building a consistent output from possibly conflicting local predictions. All approaches presented so far in this chapter, could ignore the issue of inference, since simply combining all predictions always ended up being a valid macro-label. This is no longer the case if overlapping meso-labels are predicted. For now, let us postpone the inference issue and first have a look at the potential of overlapping meso-label prediction.

3.3.4 Bounds on the performance of overlapping meso-label prediction

From the mutual-information values shown in Table 3.4, it can be seen that both grain and corn, and grain and wheat would be useful meso-labels to predict, since both have high mutual information. However, because of the requirement that meso-labels do not overlap, only one of those two could actually be used
3.3. Towards constraint satisfaction inference for multi-label classification

Figure 3.1 Illustration of the oracle and saboteur inference strategies for determining the upper and lower bound performances of inference on overlapping meso-label predictions. Both strategies have perfect knowledge about the true topics of the document—in this case, corn and grain. The oracle will use this knowledge to predict the correct value for a micro-label if the available meso-labels enable that; the saboteur uses it to predict incorrect micro-label values whenever possible.

in the previous experiment. As a result, valuable information was wasted. One potential fix for this is to join all three labels in one meso-label of length three. There is some information in most label pairs, though, and in the extreme case we would simply end up predicting macro-labels. In the next experiment, a different approach is followed. Dropping the requirement of non-overlapping meso-labels, all pairs of categories are turned into meso-labels and classifiers are trained for them.

Thus, the general procedure is similar to that of the previous section, with the exception that now, 45 different classifiers are trained, one for each pair of topics. When classifying a document with all of those, instead of one prediction, we now obtain 9 predictions for each topic. Conflicts are bound to occur among those predictions. To resolve those, an inference procedure needs to decide which predictions to follow, and which to ignore. Such an inference procedure is presented in the next subsection. First, we describe two imaginary inference strategies that allow us to determine the upper and lower bound performance of any inference strategy applied to the predicted meso-labels. For the upper bound, we devise the oracle strategy; for the lower bound, the saboteur strategy. Both approaches are explained below, and illustrated graphically in Figure 3.1.

An oracle inference strategy

The oracle method knows what categories a document does and does not belong to, but when predicting the final value of each category, it can only choose from those values predicted by the underlying meso-label predictors. For example, if the oracle knows that the current document belongs to the acq category, and at least one of the meso-label predictors made a positive prediction for this category,
the inference procedure also makes a positive prediction, regardless of whether or not there was also a conflicting negative prediction. In another situation, if the underlying meso-label predictors only made negative predictions for \texttt{acq}, the oracle predicts negative as well, even though it knows the true value for this category to be different. The performance of this oracle inference procedure gives an upper bound on what any inference procedure can attain given a set of meso-label predictions.

Table 3.3 shows the oracle inference performance on the document classification task. Comparing it with the performance of the non-overlapping meso-label predictor shows that at least the upper bound of overlapping meso-label prediction is higher than that of the non-overlapping variant. In addition, the upper bound is also higher than the performance of the macro-label predictor. This means that, given a good inference procedure, the approach might do better than meso-label prediction, and possibly even macro-label prediction.

**A saboteur inference strategy**

In the same way that it is insightful to know the upper bound of inference, it is also interesting to have a look at the lower bound. To this aim, we devise a saboteur inference procedure. As the oracle, the saboteur knows about the correct categories for a document; however, it will predict the incorrect value for a category if and only if there is at least one meso-label that makes an incorrect prediction for that category. In other words, only if all predicted meso-labels suggest the correct value for a micro-label, will it actually be predicted correctly.

The performance of the saboteur inference method is shown in Table 3.3. The main observation to be made is that the lower bound of an inference procedure on top of overlapping meso-label predictions is below the performance of the micro-label predictor. So, whereas the oracle method tells us that overlapping meso-label predictions have the potential of leading to good structured predictions, the performance of the saboteur method shows that not all inference methods will suffice to meet this potential. The actual inference procedure will still have to invest substantial effort to distinguish good meso-labels from bad ones. The next subsection introduces an inference procedure that should meet that requirement.

### 3.3.5 An inference procedure for overlapping meso-labels

In previous sections, we decomposed the full multi-label prediction task into several local prediction problems. Doing so, we assume that the local predictions are sufficiently easy to be learnable as a conventional multi-class classification task. For predicting the full multi-label classification, first each of the local classifiers is applied to the input, resulting in a set of local predictions. Combining these local predictions into a globally consistent macro-label is the task of an inference procedure. Inference for micro-label prediction and meso-label prediction with labels that do not overlap is a trivial combination of the local predictions. If,
3.3. Towards constraint satisfaction inference for multi-label classification

however, the local predictions are not necessarily consistent among one another, as is the case when predicting overlapping meso-labels, the inference procedure has an additional task to assess what local predictions it should follow, and which ones to ignore. To this end, it should estimate the reliability of local predictions, and combine meso-labels into a consistent macro-label that is as reliable as possible. This is typically expressed as the optimisation of a compatibility function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$, that estimates how compatible a certain target structure is with the given input:

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} f(x, y)$$

To apply this to our current case, the input $x$ is a document, for which we have predicted a set of overlapping meso-labels. Ideally, we would like a meso-label prediction to be an instruction to use the two suggested micro-labels in the eventual macro-label. However, it is expected that there will be conflicting predictions, so some trade off will have to be made as to which meso-labels are, and which are not used in the eventual prediction. For this, we need to estimate the reliability of a meso-label prediction. The following two intuitions form the basis for our inference procedure.

1. Local base classifiers that make the meso-label predictions associate a confidence score with their prediction. This score can be used to order conflicting meso-labels in terms of expected quality. If two meso-labels are in conflict, it is preferable to keep the higher scoring one.

2. Suppose a meso-label suggests, for example, that micro-label $y_i = 1$ and $y_j = 0$, and other, partially overlapping meso-label predictions exist that also suggest $y_i = 1$ or $y_j = 0$. Then, those meso-labels can be seen as supporting the suggestions made by the former, and thereby, increasing its reliability.

Moreover, the above two intuitions are mutually enforcing, so that supporting meso-labels that were themselves predicted with high confidence, boost reliability even more. To formalise the above, we will formulate the inference procedure as a weighted constraint satisfaction problem, a formalism which will be introduced first.

**Constraint satisfaction**

In constraint satisfaction (Monatanari, 1974; Mackworth, 1991) the goal is to find values for a set of variables that satisfy certain constraints. A constraint satisfaction problem (CSP) is defined as a tuple $(\mathcal{V}, \mathcal{D}, \mathcal{C})$. Here, $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is a finite set of variables. $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n\}$ is a set of domains for the variables in $\mathcal{V}$, such that each variable $v_i$ is allowed to take on a value of its domain $\mathcal{D}_i$. While a variable’s domain dictates the values a single variable is allowed to take on, the constraints of a CSP specify which simultaneous value combinations over
Chapter 3. Structured prediction as constraint satisfaction

A constraint $c \in \mathcal{C}$ is a pair $(S, R)$, where $S \subseteq V$ is the scope of the constraint, and $R$ is an $|S|$-ary relation that contains the allowed value assignments. It is said to be satisfied if the value of the variables in the scope of the constraint are contained by the relation. This formal definition allows for any constraint on the CSP to be defined. For practical purposes, though, constraints are often denoted more informally; for example, by stating that all variables should have different values, the sum of variable values equals a certain constant, or the structure encoded by the variables is a valid tree.

A solution to a CSP is a joint assignment of values to every variable in $V$, such that (1) each variable $v_i \in V$ is assigned a value that is a member of its domain $\mathcal{D}_i$, and (2) every constraint $c \in \mathcal{C}$ is satisfied.

In a traditional CSP, all constraints are hard constraints, that is, only solutions that satisfy all constraints are valid solutions. Several extensions to the traditional CSP exist that introduce some notion of soft constraints, i.e. constraints that are not required to be satisfied for a solution to be valid, although satisfying certain constraints may make the solution more desirable. Typically, such extensions provide a means to specify the importance of soft constraints, so that solutions that satisfy important soft constraints, but violate certain less important ones are preferred over solutions that violate many important constraints. Many such extended constraint formalisms can be cast as a specialisation of semiring-based constraint satisfaction (Bistarelli et al., 1997), a general framework for specifying constraint satisfaction problems with soft constraints. The definition of a constraint is altered slightly, such that it corresponds to a pair $(S, g)$. $S$ is still the scope as it is for hard constraints; however, the relation $R$ is replaced by a function $g$ that is defined on $S$ and associates a value to variable assignments.

The extension we will be using is a weighted constraint satisfaction approach. Candidate solutions to a weighted constraint satisfaction problem (WCSP) are scored according to the sum of weights of the constraints that they satisfy, and the highest scoring solution is selected.

$$\hat{T} = \arg\max_{T \in \mathcal{D}_1 \times \ldots \times \mathcal{D}_n} \sum_{(S, g) \in \mathcal{C}} g(T_S)$$

Note that this framework still allows for hard constraints, in addition to soft constraints. In the above equation, hard constraints can be thought of as assigning the value $-\infty$ to all non-allowable tuples.

General solving of constraint satisfaction problems is intractable. Without going into too much detail, though, we note that many years of research has resulted in many solving algorithms, both of the optimal and approximate kind (cf. Tsang, 1993; Dechter, 2003). Furthermore, subclasses of the general constraint satisfaction problem, defined by restrictions on domains or constraint structure, have been identified that allow for more efficient solving algorithms.
3.3. Towards constraint satisfaction inference for multi-label classification

Figure 3.2  The constraint satisfaction inference approach to multi-label classification. First, base classifiers predict overlapping meso-labels. From those predictions, a weighted constraint satisfaction problem is formulated, which is solved by a WCSP solver to obtain the macro-label prediction. Note that while the output space of this simplified example contains six macro-labels, the solution space of the WCSP contains only two macro-labels, since the meso-label predictions only disagree on the value of \textit{wheat}.

Constraint satisfaction inference

To use weighted constraint satisfaction as the implementation formalism for our inference procedure, we will redefine the inference procedure as a weighted constraint satisfaction problem. This CSP is defined over the output space of the prediction task. The micro-labels $y_1, \ldots, y_n$ of the original problem naturally map to the variables $v_1, \ldots, v_n$ of the CSP—for notational convenience we will therefore use the same symbols to refer to both the micro-labels and the variables of the CSP, i.e. $V = \{y_1, \ldots, y_n\}$. Weighted soft constraints will be added to the CSP to find a solution that is optimally compatible with the predicted meso-labels. More specifically, the CSP is formulated as follows.

- **Domains.** Recall that the solution space $Y$ of the structured prediction task corresponds to the space of possible macro-labels, which in turn relates to the micro-label spaces, such that $Y \subseteq Y_1 \times \ldots \times Y_n$. The solution space of a constraint satisfaction problem is defined similarly in terms of the domains of the variables, $D_1 \times \ldots \times D_n$. Since we reformulate the inference procedure as a constraint satisfaction problem, it may be tempting to equate these two concepts, that is, $D_i = Y_i$. Doing so, however, we would ignore the predictions made by the base classifiers. To see this, consider the information we have about the value of a micro-label $y_i$. There are 9 different meso-label predictions that cover, and thus suggest a value for that micro-label. If all agree that this value should be, for example, 1,

\begin{itemize}
  \item \textit{Domains.} Recall that the solution space $Y$ of the structured prediction task corresponds to the space of possible macro-labels, which in turn relates to the micro-label spaces, such that $Y \subseteq Y_1 \times \ldots \times Y_n$. The solution space of a constraint satisfaction problem is defined similarly in terms of the domains of the variables, $D_1 \times \ldots \times D_n$. Since we reformulate the inference procedure as a constraint satisfaction problem, it may be tempting to equate these two concepts, that is, $D_i = Y_i$. Doing so, however, we would ignore the predictions made by the base classifiers. To see this, consider the information we have about the value of a micro-label $y_i$. There are 9 different meso-label predictions that cover, and thus suggest a value for that micro-label. If all agree that this value should be, for example, 1,

\footnote{Note that the macro-label space is a subset of the product of micro-label spaces, rather than necessarily equal to it. Problem-specific constraints might exclude certain solutions. That being said, for general multi-label classification, the macro-label space does in fact equal the product of micro-label spaces.}
then there is strong evidence for that value, and thus we assume there is no further uncertainty to be resolved by the inference procedure; in that case, \( \mathcal{D}_i = \{1\} \subset Y_i \). As a result, the solution space to be searched by the constraint solver will be smaller than the complete solution space of the structured prediction task.

- **Constraints.** To try to retain predicted meso-labels as part of the output structure, we turn them into soft constraints. Doing this is straightforward: a prediction \((v_i, v_j)\) for the meso-label \(y_{\{i,j\}}\) gives rise to a constraint \(y_i, y_j = v_i, v_j\); it is weighted with the classifier confidence, which we will assume to be a number between 0 and 1, although we do not assume any probabilistic interpretation. Obviously, there is a high risk that incorrect constraints are added to the CSP, which happens when a prediction error occurs. Nevertheless, there are two reasons why it is expected that this strategy will not harm the inference. First, as we have seen in Section 3.3.3, the majority of meso-label predictions is in fact correct. Secondly, even if an incorrect meso-label is turned into a constraint, there are still 8 overlapping constraints, which are likely to conflict with the erroneous constraint. These two observations, in combination with the fact that all predicted constraints are soft constraints, and thus, are not required to be satisfied, should favour solutions in which incorrect constraints are left unsatisfied.

Performing multi-label classification of a document then proceeds, as illustrated in Figure 3.2, in the following steps. First, the meso-label predictors predict joint micro-label pair classifications. After that, a weighted constraint satisfaction problem is formulated, deriving both the constraints and the variable domains from the predicted meso-labels as explained above. Finally, a standard WCSP solver is used to obtain the joint multi-label prediction for the document.

Applying this procedure results in the scores listed in the row labelled CSI\(^\text{meso}\)meso in Table 3.3. First of all, it can be seen that the inference procedure operating on overlapping meso-labels performs better than the non-overlapping meso-label approach in terms of both macro-averaged and micro-averaged \(F_{\beta=1}\). This gain should mostly be attributed to the extra inference step. Both approaches have an output scope of only two micro-labels, but nevertheless the inference step is able to recover more global information. Looking at the \(F_{\beta=1}\) scores for macro-label prediction, we see that constraint satisfaction inference performs slightly better than that approach as well.

In spite of the performance improvement with respect to the \(F_{\beta=1}\) measures, we can also see that the subset accuracies of both non-overlapping meso-label prediction and macro-label prediction are higher than the accuracy attained with constraint satisfaction inference. A closer inspection of the predictions made by the latter reveals that the improvement in \(F_{\beta=1}\) is the result of a much better precision; yet this precision comes at the cost of recall. In other words, the inference step makes the system more careful in making positive micro-label predictions.
3.3. Towards constraint satisfaction inference for multi-label classification

This pays off in precision, but causes a substantial degradation in terms of recall and subset accuracy. Apparently, the inference step still cannot recover the global information required for improving both precision and recall. Indeed, even though the base classifier predictions are combined in a complex way, they only directly model pairwise dependencies between micro-labels. Higher-order dependencies are modelled indirectly at best, whereas in macro-label prediction, they are just learned directly by the classifier. Ideally, we would like to integrate more sources of global structural information in the inference procedure, so that additional dependencies are modelled. Fortunately, the effort we put into formulating our inference procedure as a constraint satisfaction problem allows us to do that.

Enriching the inference procedure with additional constraints

Originally, we conceptualised meso-labels as a literal subpart of the output structure, and turning them into constraints was merely an implementation convenience to facilitate resolving conflicts. However, if we are actually going to redefine meso-labels as constraints on the output, as opposed to literal subparts, our inference framework suddenly becomes more flexible in terms of what information can be used, and as a result becomes considerably more powerful. Note that our new definition is actually a generalisation of the old one. Constraints can still be predicted that correspond to subparts of the output; the transformation of those subparts into constraints has been described above.

However, we can also try to predict more abstract constraints on the output. Let us add a new constraint to the document classification system. This new constraint will be a constraint on the number of categories that are assigned to the document, and can formally be defined as follows.

\[ \text{Constraint:} \quad \sum_{i=1}^{n_y} y_i = m \]

Obviously, there is no way to know what value \( m \) this constraint should enforce in advance, since it is dependent on the document itself, and that is exactly one of the difficulties of multi-label classification. However, we do know that we can learn to recognise the number of topics to be assigned to a document fairly reliably. Learning to predict the number of topics for a document can be seen as a generalisation of learning to predict the macro-label for a document, a task for which we already trained a classifier successfully. Thus, a new classifier is added to the system that is responsible for predicting the exact form of the constraint. Given this classifier, we create constraints from all classes in the predicted class distribution, and thus not only from the class that is deemed most likely, as we did for the meso-label constraints. It is important to emphasise that doing so will not have an effect on the composition of the domains of the constraint satisfaction problem, and consequently, will not negatively affect the run-time of the constraint solver. Therefore, we can move the uncertainty resolution of this classifier to the inference procedure without this leading to a search cost penalty.
From the performance reported in the bottom row of Table 3.3 it can be seen that adding the new constraint type has positive effects. It outperforms the old inference procedure, and mainly does so by restoring the balance between precision and recall. This benefits the subset accuracy as well: it is much better than the accuracy attained with non-overlapping meso-label predictions. In terms of both $F_{\beta=1}$ scores, constraint satisfaction inference also convincingly outperforms macro-label prediction. The fact that, in terms of subset accuracy only, it still performs slightly worse than macro-label prediction can be explained by recognising that the macro-label predictor has been optimised explicitly for this metric. However, this advantage will not persist for tasks with larger output spaces, whereas it can be expected that constraint satisfaction inference will also perform accurately in such larger spaces.

With the introduction of a new constraint predictor it has been shown that the inference procedure can successfully integrate predictions about different aspects of the output and have those compete and cooperate with one another. The improvement resulting from adding the new constraint suggests that the extra information is indeed helpful for resolving the uncertainty left by the local meso-label predictions.

Enriching the inference procedure with knowledge-based constraints

An additional advantage of constraint satisfaction inference is its ability to easily integrate background knowledge. For some tasks, external sources of knowledge may be available, from which constraints can be extracted accurately. If that is the case, training a classifier for that constraint is wasteful and unnecessary. The constraints can simply be added to the CSP, together with the constraints derived from meso-label predictions.

For example, Punyakanok et al. (2008) demonstrate the power of globally enforced constraints as part of an inference procedure in the context of semantic-role labelling. Constraints added to the system include (1) a verb cannot have two arguments of the same type, (2) a verb has at least one argument, and (3) certain verbs disallow some argument types. Linguistic constraints are also employed in various studies on anaphora resolution (cf. Mitkov, 1997, 1999). Agreement constraints between coreferring arguments, for example, are perfectly suited for such an approach. According to such constraints, the system could enforce that two noun phrases that corefer agree in number and in gender. As a final example, Daya et al. (2004, 2008) present a learning approach to the identification of roots of Hebrew words, and show that integrating linguistic constraints on word formation in Hebrew causes a substantial performance improvement in comparison with a pure classification-based approach without such constraints.
3.4 A general formulation of constraint satisfaction inference

In the previous section, we introduced the mechanisms behind constraint satisfaction inference applied to a concrete example task. For clarity, the explanation so far has stayed close to the terminology and properties of multi-label classification. The constraint satisfaction inference framework, however, is intended to be applicable to any structured prediction task. In this section, we will therefore revisit the important principles that we previously discussed in the specific setting of multi-label document categorisation. This time, we will discuss them in more general terms, so that we can apply them to new structured prediction tasks, as will be done in the remainder of this thesis.

To apply constraint satisfaction inference to new structured prediction tasks, a number of steps need to be taken. Essentially, these steps describe how a WCSP instance is formulated based on the characteristics of the output space of the task, and information learned and predicted by a set of underlying base classifiers. Some of these steps may require task-specific design choices for which no general guidelines can be provided. Other steps may more easily borrow from related work in structured prediction or constraint satisfaction. In brief, constraint satisfaction inference comprises the following four steps.

1. Define the output space;
2. Apply trained base classifiers to predict constraints on the output space;
3. Construct a constraint satisfaction problem from the base classifier predictions;
4. Solve the constraint satisfaction problem.

The following four subsections explain each of the above steps in more detail. They provide the guidelines for the design of the task-specific implementations of the following chapters.

3.4.1 Output space

This step comes down to providing a task-specific description of the macro-label and micro-label concepts. Defining the macro-label is trivial as it simply corresponds to the target value of the task. The heart of defining the solution space therefore lies in the definition of micro-labels. Some structured output spaces are easily encoded in terms of micro-labels; examples are the output spaces of the multi-label classification task of this chapter and the sequence labelling task of Chapter 4. Other output spaces, such as those of dependency parsing in Chapter 5 and machine translation in Chapter 6 may be less easily thought of as defined by a finite set of micro-labels, but nonetheless, it is assumed that each output space
Chapter 3. Structured prediction as constraint satisfaction

is expressed in terms of micro-labels. The importance of having a well-defined micro-label concept is that it links the output space of the prediction task to the solution space of the constraint satisfaction problem that is to be predicted: a structured output space defined by the micro-labels \( y_1, \ldots, y_n \) gives rise to a constraint satisfaction problem with variables \( V = \{ y_1, \ldots, y_n \} \).

Unfortunately, no further guidelines can be provided for defining the output space that cover the general case. Each structured prediction task may face a different target value structure, and thus, defining the solution space of constraint satisfaction inference is only possible with respect to a concrete task. It is for that reason that each of the following application chapters contains an explicit definition of the micro-labels to be predicted as part of the task.

Central to defining a micro-label \( y_i \) is describing its micro-label space \( Y_i \). Again, this is a task-specific procedure. Here, we restrict ourselves to stating that the micro-label space \( Y_i \) is a feature of the prediction task in general, and does not provide the definition of the variable domain \( D_i \). The latter is constructed from base classifier predictions as is discussed later on.

### 3.4.2 Base classifiers

Even though constraint satisfaction inference constitutes a novel approach to the kinds of structured prediction tasks studied in this dissertation, conventional machine learning classifiers are still at the basis of the approach. Like in any other supervised machine learning task, these classifiers are given labelled training data from which they learn a model that can classify new input objects. That being said, though, it is important to point at the actual role that the base classifiers play in the context of the constraint satisfaction inference framework.

In Sections 3.3.1, 3.3.2, and 3.3.3, which describe non-structured approaches, the classifiers are used to predict the target value, or parts of it directly. This is in fact the common thing to do in most applications of machine learning, and it even happens in most structured approaches described in Chapter 2. In constraint satisfaction inference, however, the role of the base classifiers is slightly different. Instead of directly predicting a part of the target structure, base classifiers predict a weighted constraint satisfaction problem, the solution to which is the actual target value. The exact details as to how such a CSP is constructed from base classifier predictions is deferred to the next section. Here, we already provide a brief overview.

Formulating a constraint satisfaction problem involves giving a definition of the tuple \((V, D, C)\). In the previous subsection, it has already been discussed how the set of variables \( V \) is constructed. For that, no base classifier predictions are consulted, the variables simply follow from the structure of the prediction task. In contrast, both the set of variable domains \( D \) and the set of constraints \( C \) are partly or entirely based on base classifier predictions.

A constraint satisfaction inference approach may comprise one or several different base classifiers. Each of those may contribute domain values or constraints
to the CSP, some do only one of the two, others do both. With respect to this, we define two features of base classifiers.

**Domain-defining** A *domain-defining* classifier is a base classifier that predicts class labels from which domain values are extracted. The meso-label predictors in the example task of this chapter are domain-defining; they directly predict values for one or more micro-labels. The topic count predictor, in contrast, is not domain-defining. The number that is predicted cannot be translated into a domain value for one or several of the micro-labels. Note that the only requirement for a domain-defining classifier is that a domain value can somehow be extracted from the predicted class labels. This includes class labels that encode more information than only the required domain value. For example, the meso-label predictor for the meso-label $y_{\{i,j\}}$ is both domain-defining for $y_i$ and for $y_j$.

**Constraint-defining** A *constraint-defining* classifier is a base classifier that predicts values from which constraints are extracted. This is the case for all classifiers in the multi-label classification task of this chapter. Both the 45 meso-label predictions and the topic count predictions are directly added as constraints to the CSP.

As said, these two features are not mutually exclusive; a base classifier can be both at the same time, but can also be only one of the two.

For constructing the base classifiers to be used in constraint satisfaction inference, we used the $k$-NN algorithm, but as said, any other multi-class learning algorithm could have been used instead. In the following chapters, we will continue using $k$-NN—or memory-based learning—algorithms. Our implementation of these algorithms (Daelemans et al., 2007) has many configurable parameters, the values of which may have significant effects on the classification performance. We optimise these parameters automatically based on the training data of the task using the algorithm proposed by Van den Bosch (2004).

### 3.4.3 CSP formulation

The procedure for formulating a constraint satisfaction problem given a set of base classifier predictions does not differ substantially from the explanation given in Section 3.3.5. Here, we give a more general description.

- **Domains.** In Section 3.3.5, we explained that a domain $D_i$ simply matches the micro-label space $Y_i = \{0, 1\}$, unless all base classifiers that cover the micro-label $y_i$—i.e. the domain-defining classifiers for $y_i$—agree on either one of the two possible values, in which case the domain will only contain that single value. To generalise this to prediction tasks involving micro-label domains with sizes greater than two, we will build a domain $D_i$ from the
union of all suggested values. Thus, let $U_i$ be the set of all domain-defining base classifier predictions for micro-label $y_i$. Then, $D_i = \bigcup_{u \in U_i} u_i$.

- **Constraints.** Extracting constraints from constraint-defining base classifier predictions is mostly straightforward, since constraints can simply be encoded in the predicted class labels, and the constraint weight can be derived from the base classifier confidence. For the general case, we do not formulate any specific requirements for constraint formulation from base classifier predictions. While it is easiest to think of one classification resulting in one constraint, one could also consider extracting several constraints from a single prediction. This is what we did for the constraint on the number of topics used in the document categorisation example, for which all class labels in the predicted class distribution are turned into constraints, weighted by the score they receive in the distribution. As another example, we could have opted for extracting two additional constraints on the value of a single micro-label only from the predicted meso-labels.

### 3.4.4 Solving the CSP

The result of the previous two steps is a standard weighted constraint satisfaction problem. Solving this problem provides us with the actual structured target value. In Section 3.3.5, we briefly referred to the existence of many general-purpose constraint solving algorithms. One of the advantages of choosing the constraint satisfaction formalism as the implementation formalism for the inference is that it allows us to borrow from the extensive research on constraint satisfaction, and instantly apply these high-performance solving algorithms to structured prediction tasks. General-purpose constraint solving remains a computationally hard problem, though. The size of the solution space is exponential in the number of variables, and without any task-specific knowledge that allows for more efficient search, finding the optimal solution requires exponential time.

This general case may not always apply to any structured prediction task, however. Many tractable subclasses of constraint satisfaction problems can be identified. Such subclasses are characterised by certain restrictions on the scope or structure of constraints. For example, constraint satisfaction problems with only binary domains and binary constraints are known to be tractable. The same is true for problems in which all constraints are binary and form a tree over the variables. Constraint satisfaction problems that fall within one of such subclasses may be solved more efficiently. It is similar to the Markov properties in graphical models, which allow for efficient dynamic programming inference algorithms.

Unfortunately, by restricting the types of constraints of the underlying constraint satisfaction problem, we limit the expressiveness of the model, in the same way that Markov assumptions do for structured linear models. According to one of our three research objectives, our structured prediction framework should allow for inference without restricting assumptions, and thus without restrictions
on the types of constraints. As said above, allowing this inevitably requires us to use exponential-time solving algorithms. However, it is important to note that we construct the solution space of the constraint satisfaction problem to represent uncertainty only if really required. As a result, exhaustive, exponential-time search of the solution space may actually be feasible in some cases.

3.5 Comparison with related work

With constraint satisfaction inference, we have presented a generic framework for structured prediction. As illustrated by the survey in Chapter 2, a few more such frameworks exist, which makes it relevant to compare our framework with those existing alternatives. Since the problem setting, that of structured output spaces, and the basis, traditional machine learning, is similar for all frameworks, it is only to be expected that similarities are found between frameworks. The most obvious similarity is the divide-and-conquer approach underlying most approaches to structured prediction. However, following the objectives formulated in Chapter 1 we made certain design choices that are notably different from the approach of some of the current frameworks. Here, we will point out those differences. In this discussion, we focus on the two frameworks that share most similarities with our proposal, namely constraint satisfaction with classifiers, and structured linear models.

To facilitate the comparison, Figure 3.3 illustrates the application of the three approaches to a simplified multi-label classification task. The left-hand side of the figure shows the predictions of the base classifiers and the output scopes they relate to. The right-hand side lists the members of the output space that are evaluated during inference. In the task, three possible labels, A, B, and C, can be assigned to an instance. The labels A and B are mutually exclusive, and every instance has at least one label. As can be seen in the figure, this task is dealt with differently by the three approaches, although similarities exist as well.

Constraint satisfaction with classifiers

The most apparent similarity between the CSCL framework and constraint satisfaction inference is that the constraint satisfaction formalism is at the basis of inference. In both approaches, the variables of the CSP correspond to the micro-labels of the structured output space, constraints make sure that global structural properties are enforced in the solution, and weighted classifier predictions ensure the link between input and output. Differences exist, though, in the exact formulation of the constraint satisfaction problem, and in the role of the base classifiers.

First of all, there is a difference in how the domains of the constraint variables are composed. In CSCL, domains exactly match the corresponding micro-label space, i.e. $D_i = Y_i$. In constraint satisfaction inference, we strive for domains that are subsets of the output space, thus $D_i \subseteq Y_i$. To achieve this, the base
Figure 3.3 Comparison of three structured prediction approaches applied to a fictitious multi-label classification task. In this task, the labels A and B are mutually exclusive, and every instance has at least one label.
classifiers are actively involved in composing the variable domains. As can be seen in Figure 3.3, the effect of this strategy is that the solution space searched by constraint satisfaction inference may be much smaller than the original output space of the task, which is the space searched by CSCL. This difference in domain composition comes from a different way of resolving uncertainty. In CSCL, the base classifiers score candidate micro-label assignments, but they do not restrict them—all uncertainty is resolved during inference. Constraint satisfaction inference, on the other hand, leaves most of this to the base classifiers, only if two overlapping predictions disagree, is uncertainty retained, and left to the inference procedure to resolve.

Constraint formulation is also different in the two approaches. In CSCL, the constraints are formulated and fixed in advance. They are general domain constraints, such as in Figure 3.3, where constraints enforce the mutual exclusiveness of $A$ and $B$, and the minimum label count of one. In contrast, in CSI, the constraints are predicted by the base classifiers based on the current input. These constraints are not expected to be valid domain constraints in general; for example, the constraint $[A, B = 1, 0]$ in Figure 3.3 will be incorrect for many instances, but one of the base classifiers suggested it to be a relevant constraint for the current input. Another important difference is that in CSI, only satisfied constraints contribute to the score of a candidate solution, whereas in CSCL, micro-label assignments are scored.

**Structured linear models**

Constraint satisfaction inference also bears a resemblance to structured linear models worth pointing out. In the latter, the output structure is decomposed based on the cliques of an underlying graphical model. A global scoring function combines local score estimates that are computed on the level of those cliques. Figure 3.3 illustrates such an approach for multi-label classification, similar to the Collective Multi-Label with Features (CMLF) model of Ghamrawi and McCallum (2005). The cliques correspond to pairs of micro-labels. Essentially, the meso-labels in constraint satisfaction inference and cliques are the same concept, namely a joint label covering several micro-labels. Our meso-labels are also predicted, and assigned a score by local models. The constraint satisfaction problem combines weighted meso-label constraints into a consistent output structure that optimises the sum of local scores, which makes it similar to the global scoring function of structured linear models.

The primary difference is that the clique scores of structured linear models are actually score distributions over every possible clique configuration, i.e. joint micro-label assignments. Thus, a clique always receives some credit, whatever its configuration. Meso-label constraints in CSI, on the other hand, are just constraints that enforce a single configuration: the one predicted by the base classifier. Consequently, label assignments that do not satisfy a constraint, receive no credit at all.
More importantly, and as pointed out in the discussion with respect to CSCL, meso-label predictions not only score micro-label assignments, they also restrict them, by composing the domains of micro-labels. Structured linear models, similarly to CSCL, consider every possible value of micro-labels and do not use the underlying model to restrict the range of values. Once more, this difference is the result of a different approach to uncertainty resolution. CSI prefers resolving it in the base classifier, structured linear models always leave it to the inference. The importance of this difference is that efficient inference is possible without having to make restricting assumptions as is required in structured linear models. As an illustration of this, in Figure 3.3, the structured linear model approach considers every one of the eight possible outputs; constraint satisfaction inference, evaluates only two candidates. This way, exhaustive search of the output space is cheaper, which may allow for exhaustive search in tasks where structured linear models cannot do that.

### 3.6 Conclusion

In this chapter we have introduced constraint satisfaction inference, a generic framework for structured prediction. In accordance with the research objectives formulated in Chapter 1, the approach differs from existing alternatives in several aspects. These objectives state that the framework should allow for efficient inference without restricting the types of dependencies that can be modelled, have an efficient training procedure that does not involve an inference step, and finally, be independent of the underlying learning technique.

The second goal is met, because no inference is required as part of the training procedure. Base classifiers are trained to predict constraints, which comes down to standard multi-class learning. This also meets the third objective: independence of underlying learning technique, as any multi-class learning algorithm is usable within the framework. Efficient inference is targeted through a different way of handling uncertainty in micro-label assignments. Most structured prediction frameworks leave all uncertainty to the inference to resolve. On the other hand, constraint satisfaction inference leaves most of the uncertainty to the base classifiers to resolve. As a result, the search space of the inference procedure is likely to be considerably smaller than in other approaches. The potential weakness of this choice is that local classifiers may not have sufficient knowledge to resolve uncertainty in a way that is globally sound. However, we assume that even local classifiers are correct most of the time. By having multiple predictions cover the same micro-labels, we propose a different method for handling uncertainty. That is, only if two overlapping predictions disagree, do we make the inference procedure resolve the uncertainty.

The multi-label text categorisation task performed in this chapter only served as an introductory example. Constraint satisfaction inference is intended to be applicable to any task with a structured output space. Applying constraint satis-
faction inference to new structured prediction tasks involves two steps. First, the output space has to be defined. The formulation in macro-labels and micro-labels as used in this chapter is a convenient formalism for doing that. Secondly, constraints have to be conceived that capture the important structural dependencies. Base classifiers have to be trained for predicting these constraints given an input instance. At least one of the base classifiers should be domain-defining, but in addition, several constraint-defining base classifiers can be added. With these two ingredients, everything is at hand to build a constraint satisfaction problem given an input instance. Solving it gives the desired output structure. In the next three chapters, this procedure is applied to three challenging linguistic processing tasks with structured output spaces.
Sequences are among the most versatile output structures in natural language processing. Despite the apparent simplicity of sequences, many linguistic processing tasks can be seen as generating sequential outputs. Either because the output naturally corresponds to a sequence, such as with part-of-speech tagging, or because the targeted output structure is easily mapped to a sequence, as is the case for example in named-entity recognition. Other tasks that can be solved by predicting sequences include text-to-speech conversion, morphological parsing, and document segmentation. 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. Natural language processing comprises many input units that naturally fit the criterion of sequential structure. For example, documents are sequences of sentences, sentences are sequences of tokens, and words are sequences of letters. Often, the aim is not simply to classify such inputs 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 the same way that multi-class classification treats its classes as atomic symbols. In this respect, a naive interpretation of sequence labelling would simply rephrase it as a sequence of multi-class classification cases. However, as in any structured prediction task, it is assumed that dependencies among different elements of the output sequence are as important as dependencies between an element of the input and its label in the output. For this reason, it is expected that only structured prediction techniques can be truly successful at
4.1 Existing work

Because of the wide applicability of sequential outputs, it is unsurprising that sequence prediction has received wide attention in machine learning. In fact, many structured prediction techniques were first applied to sequence labelling, and have since then been extended to cope with general structured outputs. For this reason, most of the work covered in Chapter 2 is relevant for 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. Although there have been several different implementations of linear models for sequence labelling (Lafferty et al., 2001; Collins, 2002; Altun et al., 2003), they essentially only differ in the learning algorithm used for parameter estimation. The underlying graphical structure for all techniques mentioned is the same. This structure 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 defined by 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 same number of preceding labels, and consequently models are restricted in the types of structural dependencies that can be modelled.

Although structured linear models have been most popular for sequence labelling, a vast array of other machine learning methods have been applied to sequence labelling as well. Most of those have already been reviewed in their general form in Chapter 2. 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).

4.2 Constraint satisfaction approach

To apply constraint satisfaction inference to sequence labelling, we follow the steps outlined in Chapter 3. First, the output space is defined in terms of macro-labels, micro-labels, and how these correspond to the actual sequence to be predicted.
4.2. Constraint satisfaction approach

Then, constraints are defined over this output space that model relevant global dependencies in the predicted label sequence. Finally, we briefly discuss the choice of constraint solving algorithm for the experiments in this chapter.

4.2.1 Solution space

The definition of the solution space of sequence labelling poses no difficulties. It is easy to see that a macro-label corresponds to a complete label sequence, and that each label in this label sequence has a matching micro-label. An interesting difference with the macro-label composition of Chapter 3 is that in sequence labelling there is no fixed number of micro-labels. For the multi-label classification task described in that chapter, there were always exactly ten micro-labels; one for each of the possible categories. In contrast, for sequence labelling, the number of micro-labels does not depend on how many categories there are, but rather on the length of the input sequence. In many sequence labelling tasks, this length will differ per case, and consequently the number of micro-labels will differ as well. Looking at natural language sentences as a typical type of input sequence in natural language processing, the sequence length may be as small as one or two tokens, or as long as a few dozen tokens. Another difference with respect to the multi-label categorisation task is that in sequence labelling, micro-labels are ordered, whereas in multi-label categorisation they are not. It is exactly this ordering that is exploited by many if not all learning approaches for sequence labelling. Rather than modelling the interaction among all possible micro-labels, as was done for multi-label categorisation, only the interaction between nearby micro-labels is modelled. We adopt a similar assumption as the basis for the constraints.

4.2.2 Constraints

For the design of the constraints over the output space, the same locality assumption as used by most other sequence labelling approaches is adopted. Only constraints covering nearby micro-labels are added to the inference, which reflects that only nearby micro-labels are assumed to be interdependent. This locality assumption is known to work quite well for many sequence labelling tasks, although it remains a simplification of reality.\footnote{As will be explained later, the approach to sequence labelling presented in this chapter may allow for more global interactions to be modelled rather easily.} More specifically, constraints will be defined over meso-labels consisting of a few consecutive micro-labels. It still needs to be decided how many such micro-labels to include, but let us for now postpone this question and simply adopt the meso-label predictions proposed by Van den Bosch and Daelemans (2005). In this study, trigrams of output labels are predicted for each input token, such that the middle label of the trigram corresponds to the target label of the input token. While not intended as constraints in the original study, these trigram predictions are in fact meso-labels in the same way.
Figure 4.1  Constraint satisfaction approach to part-of-speech tagging on the basis of trigram constraints. In this example, the overlapping predictions only leave uncertainty about the tags of “declined” and “comment”, the domains corresponding to the other words contain a single value only.

As the joint category predictions of Chapter 3 were. The predictions are both domain-defining and constraint-defining, and therefore, both the domains of the micro-labels and the constraints over them can be extracted from the predicted trigrams.

As an illustration, consider the part-of-speech tagging example depicted in Figure 4.1. Following the approach by Van den Bosch and Daelemans (2005), trigrams of part-of-speech tags are predicted for each word in the input sentence. A consequence of predicting such trigrams for all words is that they overlap. Effectively, each part-of-speech tag, but the first and last of the sentence, is predicted three times: once as the right part of the trigram corresponding to the preceding word, once as the middle part of the trigram predicted for the word itself, and once as the left part of the trigram corresponding to the next word. Those three predictions may agree on the part-of-speech tag for a given word, but classification errors will often result in some degree of disagreement. If there is agreement among all three predictions, this constitutes strong evidence that the part-of-speech tag agreed upon is indeed the correct tag. Alternatively, disagreement among two or even all three predictions suggests uncertainty that requires more global sources of evidence to resolve it. The latter is the task of the inference procedure, which will simply consider all output sequences containing any one of the conflicting predictions.

From the informal description above, a more specific definition of the search space for the inference procedure in terms of variable domains corresponding to the micro-labels can be derived as follows. Given a micro-label $y_i$ and a set of
meso-label predictions covering it, the domain of $y_i$ comprises the values predicted for that micro-label as part of each of the meso-label predictions. If the meso-labels are trigrams, as assumed so far, a domain can contain as many as three different values, or as little as one value, in case of complete disagreement or full agreement respectively. However, the approach is not restricted to trigram meso-labels. Any meso-label covering the given micro-label can contribute to its domain. Note that the way domains are composed is exactly the same as for multi-label classification in Chapter 3. What has changed is the coverage of micro-labels; deriving the micro-label domains from those meso-labels proceeds in the same way.

Deriving constraints from the meso-label predictions again proceeds according to the description of Chapter 3. For each predicted meso-label a constraint is added that requires the predicted values of the micro-labels covered to be reflected in the output label sequence.

### 4.2.3 Solving the CSP

The output space of sequence labelling tasks is exponential in the length of the input sequence. In constraint satisfaction inference, effort is spent to run the constraint solver in a solution space that is smaller than the complete output space of the original problem. 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 our sequence labelling approach has to search this space.

This issue is not unique to constraint satisfaction inference, though. It is faced by any inference-based sequence labelling approach, and thus solutions for inference for sequence labelling already exist. For reasons of efficiency, exhaustive search seems to be excluded. A popular approach is to use the Viterbi algorithm. Under the Markov assumption, it is guaranteed to find the optimal solution. On the downside, the Markov assumption makes that the dependencies that are modelled are restricted. As a less-restrictive alternative to the Viterbi algorithm, approximate search algorithms have been employed, such as beam search (Ratanaparkhi, 1996), or simulated annealing (Finkel et al., 2005). Viterbi is by far the most popular inference algorithm.

It could be employed as the basis of the constraint solver as well. However, the underlying Markov assumption of the algorithm interferes with our objective of not restricting the types of dependencies that can be modelled. Unrestricted dependencies only leave exhaustive or approximate search as possibilities. For the experiments in this chapter, we choose exhaustive search. Even though, in the worst case, this leads to exponential-time inference, we assume the search space resulting from the CSP formulation to be sufficiently small to make exhaustive search feasible.
4.3 Sequence labelling tasks

In natural language processing, there is 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 sequence labelling tasks using the encoding scheme proposed by Ramshaw and Marcus (1995). According to this encoding, symbols assigned to input tokens signal whether the token is at the beginning of a segment (B), inside a segment (I), or outside a segment (O). Optional segment type labels are simply appended to this symbol. We will speak about a BIO encoding if segments always start with a B symbol, and about an IOB encoding if B only starts segments that are directly preceded by another segment of the same type.

To illustrate the applicability of constraint satisfaction inference for as wide a range of tasks as possible, we selected four representative processing tasks that can all be approached as sequence labelling tasks. Two of those tasks, letter-phoneme conversion and morphological analysis, operate on the word level, i.e. the input is a sequence of letters, together forming a single word. The other two tasks, syntactic chunking and named-entity recognition, operate on the sentence level, where the input is a sequence of words, corresponding to one single sentence. In addition, the four tasks include segmentation tasks—morphological analysis, syntactic chunking, and named-entity recognition—and one pure labelling task, letter-phoneme conversion. The four tasks, together with the feature representations used in the experiments, are presented briefly below.

4.3.1 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 4.2 shows an example of a sentence and the base phrases that are to be recognised as part of the syntactic chunking task.

The syntactic chunking task as performed in this chapter aims to find all chunks present in the sentence, whatever their type. In contrast, a popular subclass of this full syntactic chunking—NP chunking—only aims at finding noun phrases, ignoring all other chunk types. Several published results on syntactic
4.3. Sequence labelling tasks

chunking (e.g. Sha and Pereira, 2003; Sutton et al., 2004; Daumé III, 2006) only report on NP chunking experiments. It is important to keep this difference in mind, since scores on NP chunking and on the full chunking tasks cannot be compared directly. The former tend to be slightly higher, although the difference is rather small.

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 BIO notation. Besides the words and the chunks for each sentence, part-of-speech tags for those words have been obtained by running the part-of-speech tagger by Brill (1994) on the data. Using these predicted part-of-speech 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 correspond 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 on in this chapter.

The features we 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, part-of-speech tag, and a symbol encoding certain orthographical features of the word. In addition, in a window of three tokens centred around the focus token, we include conjunctions of pairs of consecutive words, and the same for their part-of-speech tags. Finally, again in a three-token window, conjunctions of word forms and their part-of-speech tag are encoded as features. Figure 4.3 illustrates these features in the context of an example sentence.

4.3.2 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.
Mr. Meador had been executive vice president of Balcor.

**Figure 4.3** Feature representation for the word “had” in the above sentence as used with the syntactic chunking task.

**Figure 4.4** Example sentence from the named-entity recognition task.

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 suitable 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 we use the English data. 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 IOB notation. The named entities to be recognised have been divided in four classes: persons, 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, we used the “testa” subset for development purposes, and the “testb” subset for the final evaluation.

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
4.3. Sequence labelling tasks

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 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. 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 we added features signalling the presence of the word in entity-specific gazetteer lists. 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. Figure 4.5 shows an example feature vector for the given sentence.

### Figure 4.5

Feature representation for the word “minister” in the above sentence as used with the named-entity recognition task.

<table>
<thead>
<tr>
<th>Word-2</th>
<th>Word-1</th>
<th>Word</th>
<th>Word+1</th>
<th>Word+2</th>
</tr>
</thead>
<tbody>
<tr>
<td>word = proposes</td>
<td>word = foreign</td>
<td>word = minister</td>
<td>word = for</td>
<td>word = U.N.</td>
</tr>
<tr>
<td>tag = VBZ</td>
<td>tag = JJ</td>
<td>tag = NN</td>
<td>tag = IN</td>
<td>tag = NNP</td>
</tr>
<tr>
<td>orth = a+</td>
<td>orth = a+</td>
<td>orth = a+</td>
<td>orth = a+</td>
<td>orth = A.A.</td>
</tr>
<tr>
<td>word/tag = foreign/JJ</td>
<td>word/tag = minister/NN</td>
<td>word/tag = for/IN</td>
<td>word/tag = for/IN</td>
<td>gazetteers = MISC/ORG</td>
</tr>
<tr>
<td>chunk = B-NP</td>
<td>chunk = I-NP</td>
<td>chunk = B-PP</td>
<td>gazetteers = none</td>
<td>gazetteers = none</td>
</tr>
<tr>
<td>gazetteers = none</td>
<td>gazetteers = none</td>
<td></td>
<td>prefix2 = mi</td>
<td>prefix2 = er</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>prefix3 = min</td>
<td>prefix3 = ter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>suffix2 = er</td>
<td>suffix3 = ter</td>
</tr>
</tbody>
</table>

### Word-1/Word

<table>
<thead>
<tr>
<th>words = foreign/minister</th>
<th>words = minister/for</th>
</tr>
</thead>
<tbody>
<tr>
<td>tags = JJ/N</td>
<td>tags = NN/IN</td>
</tr>
<tr>
<td>orth = a+/a+</td>
<td>orth = a+/a+</td>
</tr>
</tbody>
</table>

### 4.3.3 Letter-phoneme conversion

A text-to-speech system needs to perform various processing steps in order to turn a written text into an audible sound signal. One of those steps involves obtaining a phonetic transcription of the words to be pronounced. While many systems incorporate large pronunciation dictionaries that store the pronunciation for the most common words, unknown words will occasionally have to be pronounced. In the letter-phoneme conversion task, we will only be concerned with the latter
case, i.e. producing a phoneme sequence for unseen words. In the strictest sense, letter-phoneme conversion is not a pure sequence labelling task, in which there is a one-to-one correspondence of letters and phonemes. Instead, several letters may map to one phoneme, one letter may need to be pronounced with several phonemes, or letters should not be pronounced at all. As a result, letter-phoneme conversion is actually a sequence-to-sequence mapping task where input sequences and output sequences might not be equally long. Nevertheless, we will reformulate letter-phoneme conversion as a pure sequence labelling task, in which input and output sequences will have the same length. This reformulation is reasonable, since one-to-one correspondences between letters and phonemes are still most frequent.

Turning letter-phoneme conversion into a pure sequence labelling task requires an alignment procedure that matches letters with phonemes, and if necessary inserts special null phonemes in the output sequence, or merges phonemes to make sure both input and output sequence have the same length. The former may be necessary to reflect that a letter is not pronounced, or several letters map to a single phoneme, but only one of those can actually be aligned with it. The latter is required to represent two phonemes that map to single letter—e.g. two phonemes [ks] mapped to the letter “x” are replaced by a double phoneme [X]. The actual alignment has been performed by a search method based on expectation-maximisation (Dempster et al., 1977).

Using the above procedure, 65,467 words and aligned phoneme sequences have been obtained from the English part of the CELEX lexical database (Baayen et al., 1993). 80% of those words have been used for training, the remaining 20% have been evenly divided over development and test sets. Before this split, the order of the words has been shuffled randomly to prevent clustering of similar words as a consequence of the original, alphabetic order.

The features used for the letter-phoneme conversion task are rather basic. They include the letters in a seven-letter window centred around the focus letter, and conjunctions of consecutive letters of widths two and three overlapping with the focus letter.

4.3.4 Morphological analysis

Morphemes are the primitive building blocks of lexical meaning. Any word is composed of one or more such morphemes, and its meaning relates to the meaning of its morphemes. Morphological analysis is concerned with finding the morphemes a given input word is built from. It is a word-level segmentation task in which the segments correspond to morphemes. In addition to segmenting the words, the analysis should also label the morphemes, for example indicating whether it is a stem, an inflectional suffix, or some other type of affix. As an example, the morphological analysis of the word *booking* is \([\text{book}]_{\text{stem}}[\text{ing}]_{\text{inflection}}\). The potential uses of such a segmentation are numerous in natural language processing. For example, syntactic parsers may extract agreement features of words, a morpho-
logical analysis of a word may help recovering its lemma, and a spell checker can break apart compound words on the basis of morphological structure.

The data for the morphological analysis experiments have been obtained from the English part of CELEX, which contains morphological analyses of 65,558 words. From those words, 80% are used as training data; development and test sets have been composed of 10% of the words each. As with the letter-phoneme conversion data, the order of the words has been shuffled randomly before this split to counteract the effects of the original, alphabetic order. To perform the morphological segmentation task with the sequence labelling method of this chapter, it is encoded using BIO notation, such that each letter of the input word is assigned a symbol. Four different segment types are distinguished: a stem, an inflection, a stress-affecting affix, or a stress-neutral affix. The feature set used for this task is the same as for letter-phoneme conversion.

4.4 Experimental setup

4.4.1 Evaluation

As diverse as the various applications for sequence labelling are, so are the methods for evaluating performance on sequence labelling tasks. Often, the specific target domain dictates one of those methods as the most suited for the application. However, in this chapter we do claim there is one learning method that can be used for performing all the diverse tasks that can be formulated as sequence labelling problems. Therefore, it seems only natural that not only task-specific performance measures are interesting to look at, but also metrics that purely consider the quality of the predicted label sequence without taking into account task-specific loss functions. When evaluating constraint satisfaction inference for the four benchmark tasks, we will use both types of measures.

Arguably the purest way to evaluate sequence labelling performance is to measure the proportion of complete label sequences that are predicted correctly. **Sequence accuracy**, as this metric will henceforth be referred to, is the inverse of 0/1 loss on the sequence level. As such, it is an extremely strict measure, which ignores the fact that even partly incorrect label sequences may still be usable in many applications. Nevertheless, sequence accuracies will be reported, since they are insightful for the pure sequence labelling performance of the learning approach. Moreover, for letter-phoneme conversion, sequence accuracy is in fact more relevant than the proportion of correct individual labels, since too many deviations from the correct phoneme sequence lead to a distorted speech signal.

While sequence accuracy is the strictest performance metric for label sequences, the most forgiving performance criterion simply counts the proportion of individual micro-labels predicted correctly. This metric, henceforth **token accuracy**, corresponds to the inverse of 0/1 loss on the token level, or equivalently, the hamming loss on the sequence level. As an advantage of token accuracy, label
sequences that are almost yet not completely predicted correctly still contribute proportionally to the performance score. For some tasks—such as part-of-speech 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. Similar lines of reasoning, to a greater or lesser extent, go for the three other tasks dealt with in this chapter.

Four out of three tasks are in fact 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. Obviously, sequence accuracy does benefit from correctly identified segments, albeit in a rather strict way; token accuracy does not necessarily credit segmentation quality. The most common metrics that do specifically measure segmentation, and optionally labelling, quality are precision, recall, and $F_\beta=1$. Precision corresponds to the proportion of predicted segments that are correct, where we will consider a segment 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.

### 4.4.2 Constraint prediction

The choice for $n$-gram meso-label constraints still leaves the parameter $n$ to tune, or at the very least, to fix 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, we choose not to do this, but instead decide on an $n$-gram length that will be used for all 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 meso-label to predict simply matches the macro-label. It has already been discussed in Chapter 3 why this is 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
4.4. Experimental setup

<table>
<thead>
<tr>
<th></th>
<th>n-gram</th>
<th>Token</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td></td>
<td>96.64</td>
<td>79.24</td>
</tr>
<tr>
<td>Oracle\textsuperscript{3-gram}</td>
<td>91.25</td>
<td>98.64</td>
<td>91.26</td>
</tr>
<tr>
<td>Saboteur\textsuperscript{3-gram}</td>
<td>91.25</td>
<td>92.89</td>
<td>62.04</td>
</tr>
<tr>
<td>CSI\textsuperscript{3-gram}</td>
<td>91.25</td>
<td>97.15</td>
<td>83.85</td>
</tr>
<tr>
<td>Oracle\textsuperscript{5-gram}</td>
<td>81.15</td>
<td>99.13</td>
<td>94.36</td>
</tr>
<tr>
<td>Saboteur\textsuperscript{5-gram}</td>
<td>81.15</td>
<td>83.69</td>
<td>34.71</td>
</tr>
<tr>
<td>CSI\textsuperscript{5-gram}</td>
<td>81.15</td>
<td>94.64</td>
<td>71.92</td>
</tr>
</tbody>
</table>

Table 4.1 shows the outcomes of these experiments, evaluated on the development set. The n-gram column lists the prediction accuracies of the constraint-predicting base classifiers. Unsurprisingly, the higher the n-gram length, the more difficult it is to predict the constraints correctly. This does not mean though, that constraint satisfaction inference will necessarily perform worse. In fact, the oracle scores suggest that longer n-grams potentially allow for better performance, even though the accuracy of the predicted n-grams decrease with increasing n. The explanation for this is that longer n-grams often result in more candidate labels for each token position, and consequently a larger solution space for the inference procedure. This larger solution space contains candidate solutions not part of

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. With respect to the latter, note that constraint satisfaction inference based on unigram constraints only is equivalent to micro-label prediction, which has been shown in Chapter 3 to be suboptimal for structured prediction. To illustrate the effect of the value of n, we run two experiments with constraint satisfaction inference for letter-phoneme conversion based on trigram constraints and 5-gram constraints, respectively. In addition, we also report scores for a unigram predicting baseline classifier.
the smaller solution space constructed from the trigram predictions, and if the correct solution is among those, the oracle inference procedure knows how to find it. Unfortunately, the larger solution space resulting from the 5-gram predictions contains many weaker solutions as well. Indeed, we can see that the saboteur scores for 5-gram predictions are much lower than for trigram predictions. It will depend on the quality of the predicted constraints whether the larger solution space of the inference based on 5-gram predictions is a strength or a weakness.

Looking at the performance scores for real constraint satisfaction inference on top of the predicted constraints, it can be seen that constraint satisfaction inference does indeed break down when based on 5-gram constraints. Apparently, the solution space is too large and the predicted constraints too noisy to successfully identify the correct solution. The result is that the performance is even worse than that of the local baseline system.

In contrast, if constraint satisfaction inference is run on top of trigram constraints, the performance is substantially better than that of the baseline. This shows that, even though the theoretical optimum for trigram constraint satisfaction inference is worse than that of 5-gram constraint satisfaction inference, the quality of the predicted constraints is the decisive factor in how closely the theoretical optimum can actually be approached.

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, we will choose trigram constraints as the basis for all further experiments in this chapter. It is expected that the effect of longer $n$-gram constraints as illustrated for letter-phoneme conversion, will also be observed with most other realistic sequence labelling tasks. Another 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.

**Deriving additional constraints**

Although trigram constraints enable more accurate structured predictions when compared to a unigram baseline, they are still a rather strict way to estimate compatibility with base classifier predictions. A candidate output sequence only receives credit when all three labels covered by the trigram constraint match. If two out of three labels match, no credit is received at all. However, modelling such partial credit assignment might in fact be beneficial, since subparts of predicted trigrams may be expected to be more accurate. For this reason, extra constraints are derived from the predicted trigrams. The goal of those is to model partial credit assignment as discussed before. We propose two complementary methods for deriving extra constraints.
Table 4.2  Performance on the syntactic chunking task. The trigram column lists the accuracy of the trigram-predicting base classifier. The remaining columns correspond to performance measures on the label sequences.

<table>
<thead>
<tr>
<th></th>
<th>Trigram</th>
<th>Token</th>
<th>Sequence</th>
<th>Prec</th>
<th>Rec</th>
<th>$F_{\beta=1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>94.81</td>
<td>44.33</td>
<td>90.60</td>
<td>92.62</td>
<td>91.60</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>88.21</td>
<td>97.70</td>
<td>69.88</td>
<td>95.95</td>
<td>96.41</td>
<td>96.18</td>
</tr>
<tr>
<td>Saboteur</td>
<td>88.21</td>
<td>90.34</td>
<td>27.58</td>
<td>84.02</td>
<td>85.95</td>
<td>84.98</td>
</tr>
<tr>
<td>CSI</td>
<td>88.21</td>
<td>96.03</td>
<td>58.00</td>
<td>93.81</td>
<td>93.80</td>
<td>93.80</td>
</tr>
<tr>
<td>CRF</td>
<td>95.94</td>
<td>59.29</td>
<td>93.84</td>
<td>93.70</td>
<td>93.77</td>
<td></td>
</tr>
<tr>
<td>Structured perceptron</td>
<td>95.85</td>
<td>57.80</td>
<td>93.47</td>
<td>93.55</td>
<td>93.51</td>
<td></td>
</tr>
</tbody>
</table>

1. In addition to the predicted trigram, constraints are also created from the two bigrams and three unigrams covered by the trigram. The weight for such an $n$-gram constraint (where $n \in \{1,2\}$) corresponds to the sum of the classification confidences for all trigrams in the predicted class distribution that also contain the same $n$-gram.

2. Constraints are not only derived from the highest scoring trigram in the class distribution, but from the top-3 predicted trigrams instead. Note that this is only done for the constraints. Micro-label domains are still constructed from the highest scoring predictions only.

The extra constraints have a small positive effect: token accuracy increases to 97.21, and the sequence accuracy becomes 84.17. It is important to emphasise that adding all these new constraints does not alter the size of the solution space, and therefore does not affect the efficiency of constraint satisfaction inference. Even the second method, which takes into account more trigram predictions, only results in a larger number of constraints, but does not change the micro-label domains. Because of this, we extract these additional constraints from the trigram predictions for all tasks performed in the experiments.

4.5 Results

The constraint satisfaction approach introduced in the previous sections can be applied to any linguistic processing task that can be formulated as a sequence labelling problem. To verify this, we ran experiments with constraint satisfaction inference for the four tasks introduced in Section 4.4. Since sequence labelling has received wide attention in recent years, and as a result many sequence labelling methods are already available, it is also interesting to see how constraint satisfaction inference compares with those techniques. To this end, a selection of existing
### Table 4.3 Performance on the named-entity recognition task. The trigram column lists the accuracy of the trigram-predicting base classifier. The remaining columns correspond to performance measures on the label sequences.

<table>
<thead>
<tr>
<th>Method</th>
<th>Trigram</th>
<th>Token</th>
<th>Sequence</th>
<th>Prec</th>
<th>Rec</th>
<th>$F_{\beta=1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>96.28</td>
<td>69.50</td>
<td>73.52</td>
<td>79.82</td>
<td>76.54</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>92.31</td>
<td>98.05</td>
<td>82.88</td>
<td>88.38</td>
<td>89.43</td>
<td>88.90</td>
</tr>
<tr>
<td>Saboteur</td>
<td>92.31</td>
<td>94.32</td>
<td>63.48</td>
<td>68.16</td>
<td>71.26</td>
<td>69.68</td>
</tr>
<tr>
<td>CSI</td>
<td>92.31</td>
<td>97.15</td>
<td>77.87</td>
<td>85.88</td>
<td>85.29</td>
<td>85.58</td>
</tr>
<tr>
<td>CRF</td>
<td>96.83</td>
<td>76.98</td>
<td>84.84</td>
<td>83.82</td>
<td>84.32</td>
<td></td>
</tr>
<tr>
<td>Structured perceptron</td>
<td>96.86</td>
<td>76.37</td>
<td>83.83</td>
<td>83.71</td>
<td>83.77</td>
<td></td>
</tr>
</tbody>
</table>

### Table 4.4 Performance on the letter-phoneme conversion task. The trigram column lists the accuracy of the trigram-predicting base classifier. The remaining columns correspond to performance measures on the label sequences.

<table>
<thead>
<tr>
<th>Method</th>
<th>Trigram</th>
<th>Token</th>
<th>Sequence</th>
<th>Prec</th>
<th>Rec</th>
<th>$F_{\beta=1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>94.92</td>
<td>73.53</td>
<td>79.41</td>
<td>85.79</td>
<td>82.48</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>91.59</td>
<td>98.05</td>
<td>89.06</td>
<td>92.37</td>
<td>94.35</td>
<td>93.35</td>
</tr>
<tr>
<td>Saboteur</td>
<td>91.59</td>
<td>90.99</td>
<td>64.70</td>
<td>73.96</td>
<td>78.71</td>
<td>76.26</td>
</tr>
<tr>
<td>CSI</td>
<td>91.59</td>
<td>96.59</td>
<td>85.32</td>
<td>90.37</td>
<td>91.32</td>
<td>90.85</td>
</tr>
<tr>
<td>CRF</td>
<td>94.97</td>
<td>79.85</td>
<td>87.01</td>
<td>86.61</td>
<td>86.81</td>
<td></td>
</tr>
<tr>
<td>Structured perceptron</td>
<td>94.60</td>
<td>78.60</td>
<td>86.13</td>
<td>85.85</td>
<td>85.99</td>
<td></td>
</tr>
</tbody>
</table>

### Table 4.5 Performance on the morphological analysis task. The trigram column lists the accuracy of the trigram-predicting base classifier. The remaining columns correspond to performance measures on the label sequences.

<table>
<thead>
<tr>
<th>Method</th>
<th>Trigram</th>
<th>Token</th>
<th>Sequence</th>
<th>Prec</th>
<th>Rec</th>
<th>$F_{\beta=1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>94.92</td>
<td>73.53</td>
<td>79.41</td>
<td>85.79</td>
<td>82.48</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>91.59</td>
<td>98.05</td>
<td>89.06</td>
<td>92.37</td>
<td>94.35</td>
<td>93.35</td>
</tr>
<tr>
<td>Saboteur</td>
<td>91.59</td>
<td>90.99</td>
<td>64.70</td>
<td>73.96</td>
<td>78.71</td>
<td>76.26</td>
</tr>
<tr>
<td>CSI</td>
<td>91.59</td>
<td>96.59</td>
<td>85.32</td>
<td>90.37</td>
<td>91.32</td>
<td>90.85</td>
</tr>
<tr>
<td>CRF</td>
<td>94.97</td>
<td>79.85</td>
<td>87.01</td>
<td>86.61</td>
<td>86.81</td>
<td></td>
</tr>
<tr>
<td>Structured perceptron</td>
<td>94.60</td>
<td>78.60</td>
<td>86.13</td>
<td>85.85</td>
<td>85.99</td>
<td></td>
</tr>
</tbody>
</table>
4.5. Results

sequence labelling methods have been trained and tested on the same data sets, and using the same features as used with constraint satisfaction inference. This setup is arguably the most objective one to directly compare the performance of the various different methods. Nevertheless, it may also cause certain techniques to perform suboptimally, because different types of feature sets are necessary to attain optimal performance. As an alternative means of evaluating the performance of constraint satisfaction inference, we will also compare the results with other published results on the same data sets. Specifically, this is possible for the syntactic chunking and named-entity recognition tasks, since the data splits for those have been standardised.

The experimental results are spread across four tables. Table 4.2 shows the scores for syntactic chunking, Table 4.3 for named-entity recognition, Table 4.4 for letter-phoneme conversion, and Table 4.5 for morphological analysis. The columns correspond to several evaluation metrics, introduced in Section 4.4.1. Trigram accuracy only applies to constraint satisfaction inference; it is the classification accuracy of the trigram-predicting base classifier. The other columns report token and sequence accuracies, and for all tasks but letter-phoneme conversion, precision, recall, and $F_{\beta=1}$ computed on segments. In addition to the performance of constraint satisfaction inference and the alternative sequence labelling techniques, each table also lists two other types of information. First of all, the first rows report the performance of a simple baseline system. For this system, a memory-based learner is trained to predict each output label separately without taking into account any information about the rest of the output label sequence. In other words, the baseline scores reflect the performance of a typical non-structured prediction technique. Finally, the Oracle and Saboteur rows in each table correspond to the inference procedures introduced in Chapter 3 for computing the upper and lower bounds on performance given the predicted trigrams.

Let us first focus on the performance of constraint satisfaction inference, and come back on the comparison with alternative sequence labelling techniques later. When comparing with the baseline approach, it can be seen that constraint satisfaction inference shows superior performance on all four tasks. With respect to this, it is important to note how similar the underlying classifiers are. Both are memory-based classifiers and have been trained on the same data with exactly the same features. In other words, the preconditions for both approaches have been the same. However, on the output side, the two approaches diverge; one simply predicts single output labels and ignores the rest of the output sequence, the other predicts trigrams of output labels and then runs constraint satisfaction inference to optimise global output structure. The results emphasise once more how important this final step is for structured prediction.

In order to allow for constraint satisfaction inference, the base classifier first has to perform the relatively difficult task of predicting three output labels simultaneously. As expected, the accuracies on predicted trigrams are worse than those on the unigram predictions of the baseline system. Nevertheless, considering that
the task is more difficult than only predicting unigrams, the results show that the predictions can actually be made quite accurately. For all tasks but syntactic chunking, the accuracies on predicted trigrams are above 90%; for chunking the accuracy is 88.2%. However, the accuracies as such are not important. What is important is what solution space they give rise to and how well they can guide the inference procedure in finding the best solution in that space. For the former, it is insightful to consider the Oracle and Saboteur scores. From those, it can be concluded that the solution spaces for all four tasks contain both label sequences whose quality is well above that of the baseline predictions, and sequences that are considerably worse. It all depends on the quality of the constraints, what end of this spectrum the actual performance ends up in. Here, again the findings are uniform over all four tasks. The performance of constraint satisfaction inference is always closer to the upper bound than it is to the lower bound. This confirms that indeed the predicted constraints help guide the inference procedure in selecting a good solution.

It is also important to point at the role of the constraint weighting. For all tasks, approximately 10% of all predicted constraints are incorrect, or at least partly so. Given that there is overlap among predicted constraints, there will be conflicts between correct and incorrect constraints. The inference procedure faces the task of deciding which constraints to prefer, and the constraint weighting, based on classification confidence, apparently successfully assists it in doing so.

4.5.1 Comparison to alternative techniques

We have shown that constraint satisfaction inference outperforms a baseline approach that ignores global output structure. As a more critical touchstone, we also compared constraint satisfaction inference with two implementations of structured linear models, one of the dominant approaches to sequence labelling. Conditional random fields (CRF) implement the structured linear model with maximum-entropy parameter estimation (Lafferty et al., 2001); the structured perceptron uses the perceptron update rule to train its model (Collins, 2002), and performs smoothing by averaging the parameters obtained at the end of each training iteration. The findings with respect to these two methods are twofold.

First, on the two sentence-based tasks, syntactic chunking and named-entity recognition, constraint satisfaction inference performs comparably with the other techniques. As will be illustrated later in this section, the scores for syntactic chunking are among the highest reported for the case where training is restricted to only using the training data used for the experiments in this chapter. It may therefore be speculated that the convergence point for the task based on this training set has been reached, and that further improvements should be sought in more clever ways to treat unknown words. Nevertheless, the experiments do illustrate that good structured prediction is a necessity for syntactic chunking, and in that respect, constraint satisfaction inference compares favourably with state-of-the-art approaches.
4.5. Results

<table>
<thead>
<tr>
<th></th>
<th>Prec</th>
<th>Rec</th>
<th>$F_{\beta=1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ando and Zhang (2005)</td>
<td>94.57</td>
<td>94.20</td>
<td>94.39</td>
</tr>
<tr>
<td>Zhang et al. (2002)</td>
<td>94.28</td>
<td>94.07</td>
<td>94.17</td>
</tr>
<tr>
<td>Kudo and Matsumoto (2001)</td>
<td>93.89</td>
<td>93.92</td>
<td>93.91</td>
</tr>
<tr>
<td><strong>CSI</strong></td>
<td><strong>93.81</strong></td>
<td><strong>93.80</strong></td>
<td><strong>93.80</strong></td>
</tr>
<tr>
<td>Carreras (2005)</td>
<td>94.20</td>
<td>93.38</td>
<td>93.79</td>
</tr>
<tr>
<td>Kudo and Matsumoto (2000)</td>
<td>93.45</td>
<td>93.51</td>
<td>93.48</td>
</tr>
</tbody>
</table>

Table 4.6  Comparison of the performance of constraint satisfaction inference on syntactic chunking with other published results.

Similarly, for named-entity recognition, constraint satisfaction inference successfully passes the comparison with the other techniques—it is even slightly better than the best-performing of those two, conditional random fields. Unlike for syntactic chunking, better scores for this task have been reported. However, the improvement still possible lies mainly in more powerful features. For the experiments reported on, all techniques used the same set of features. In this setting, constraint satisfaction scored among the best.

On the other two tasks, letter-phoneme conversion and morphological analysis, a different picture emerges. Constraint satisfaction inference vastly outperforms the other structured prediction techniques. In fact, the latter are even worse than the baseline approach. The large difference appears to be related to properties of the underlying learning algorithms. Both the maximum-entropy training procedure of conditional random fields and the perceptron update rule used for training structured perceptrons are greedy learning algorithms, which learn models that are abstractions of the phenomena that occur in the training data, as opposed to the memory-based learning method used for the baseline system and for our constraint satisfaction inference approach, which is a lazy learning algorithm that stores every training example in its instance base. A disadvantage of greedy learning algorithms is that they abstract away from exceptional training examples, assuming these to be noise. Yet, such exceptional cases turn out to be valid training examples in many language learning tasks. This is especially true for the English letter-phoneme conversion and morphological analysis tasks, in which one has to deal with the English spelling rules, notorious for having many exceptions. Indeed, Daelemans et al. (1999) have found ignoring exceptional training examples in, among others, letter-phoneme conversion to be a suboptimal strategy for this task. A memory-based learning approach, which does not forget exceptions, is at an advantage in this situation.
Chapter 4. Sequence labelling

Table 4.7 Comparison of the performance of constraint satisfaction inference on named-entity recognition with other published results.

<table>
<thead>
<tr>
<th></th>
<th>Prec</th>
<th>Rec</th>
<th>$F_{\beta=1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ando and Zhang (2005)</td>
<td>-</td>
<td>-</td>
<td>89.31</td>
</tr>
<tr>
<td>Florian et al. (2003)</td>
<td>88.99</td>
<td>88.54</td>
<td>88.76</td>
</tr>
<tr>
<td>Chieu and Ng (2003)</td>
<td>88.12</td>
<td>88.51</td>
<td>88.31</td>
</tr>
<tr>
<td>Klein et al. (2003)</td>
<td>86.12</td>
<td>86.49</td>
<td>86.31</td>
</tr>
<tr>
<td>CSI</td>
<td><strong>85.88</strong></td>
<td><strong>85.29</strong></td>
<td><strong>85.58</strong></td>
</tr>
<tr>
<td>Zhang and Johnson (2003)</td>
<td>86.13</td>
<td>84.88</td>
<td>85.50</td>
</tr>
<tr>
<td>Carreras et al. (2003)</td>
<td>84.05</td>
<td>85.96</td>
<td>85.00</td>
</tr>
</tbody>
</table>

4.5.2 Comparison to other published results

In the experiments reported upon, constraint satisfaction inference has been compared with state-of-the-art sequence labelling techniques in a setup where the feature sets were the same for all techniques. Although this is by all means the most objective approach to such a comparison, it is certainly true that some methods might actually perform better with different types of features. This is, for example, a likely explanation for the large performance differences observed for letter-phoneme conversion and morphological analysis. 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. We will briefly discuss the top-performing systems for both tasks.

First of all, the top-performing systems for syntactic chunking are listed in Table 4.6. It is interesting to note that the CoNLL-2000 shared task on syntactic chunking took place before 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. True structured prediction approaches published since then easily outperform those scores. 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 general picture is slightly different. As can
be seen in Table 4.7, the top-performing systems in the CoNLL-2003 shared task are still among 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 all but a small minority of tokens actually belong to a single class makes that carefully crafted local classifiers can 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).

4.6 Analysis of base classifier impact

Looking at the constraint satisfaction inference procedure, the relation between the classifications made by a base classifier and the eventual output structure coming out of the inference is a rather indirect one. The predicted trigrams do not necessarily end up in the output sequence. Instead, constraints are derived from them that express a preference for the corresponding trigram to be copied to the output. However, such constraints may be ignored if other, conflicting constraints make for a higher-scoring solution to the constraint satisfaction problem. The logical question then follows, what exactly is good base classifier performance in the context of constraint satisfaction inference? More practically formulated, what should a base classifier be optimised for? Is it necessarily the case that better base classifier accuracy means better structured predictions? If the latter were true, that means one can safely optimise a base classifier in isolation, that is, without taking into account its future role as a constraint satisfaction inference base classifier. If however, constraint satisfaction inference is not necessarily served with the best possible base classifier accuracy, it might be needed to optimise the base classifier as part of the inference procedure, which is likely to be more expensive.

To provide some more insight in the relation between base classifier accuracy and output sequence quality, an experiment has been performed in which base classifiers of varying quality were used to drive the constraint satisfaction inference procedure. The sequence labelling task to be performed in this experiment was letter-phoneme conversion, where the development set was used for evaluation purposes. In order to obtain base classifiers with different classification accuracies, a variety of different classification parameter settings were used when applying the memory-based classifiers. This resulted in a large number of base classifiers with trigram accuracies ranging from 71.09 to 91.32.
Figure 4.6  Average number of candidate output sequences as a function of base classifier accuracy on the letter-phoneme conversion task
4.6. Analysis of base classifier impact

First of all, let us have a look at the effect of base classifier accuracy on the size of the solution space for the constraint satisfaction solver. This size can be computed as a function of the sizes of the micro-label domains derived from the trigram predictions. The plot in Figure 4.6 shows the average number of candidate sequences available to the inference procedure as a function of the prediction accuracy of the base classifier. It can be seen that this number drops with increasing classification accuracy. This finding may be unsurprising, but it is an important one nevertheless. First of all, the size of the output space may be expected to be correlated with the quality of the solutions found by the inference procedure. In other words, smaller solution spaces probably lead to better solutions. Another important consequence is that small solution spaces allow for constraint satisfaction solving by exhaustive search. This means that no restricting assumptions have to be made, and therefore constraints may apply to arbitrary structural properties.

Another effect of base classifier accuracy we are interested in is the relation between base classifier accuracy and output sequence accuracy. This relation is depicted in Figure 4.7 for three inference procedures: the Oracle and Saboteur...
Chapter 4. Sequence labelling

procedures, and constraint satisfaction inference. A number of conclusions can be drawn from those results. To start with the most important one, there appears to be a positive linear relation between base classifier accuracy and output sequence quality. As a consequence of this, better base classifier accuracy always means better structured prediction performance. The approximately linear relation applies to all three inference strategies. In other words, not only does the actual performance of constraint satisfaction inference improve, but also the worst-case performance and the theoretical optimum. The latter implies that, even though the size of the solution space decreases with increasing base classifier accuracy, this smaller solution space does not result in good solutions becoming unavailable to the inference procedure. On the contrary, the smaller solution spaces tend to contain better solutions. A second observation to be made from the plot is that as base classifier performance improves, the gap between the three inference strategies narrows steadily. In the limit, if the base classifier were perfect, all three inference strategies would make the same predictions; namely, the correct ones.

4.7 Conclusion

The sequence labelling experiments reported upon in this chapter provide us with the empirical means to relate the constraint satisfaction framework to the three research objectives formulated in Chapter 1. Here, we focus on the objective relating to inference without restricting assumptions. The other two objectives, efficient training and compatibility with arbitrary learning methods, are met implicitly by the design of the constraint satisfaction inference framework.

Sequence labelling yields an output space that, for structured prediction standards, is relatively modest: each element of the input sequence gives rise to a single micro-label. Nevertheless, because of the dependence on the length of the input sequence, the output space complexity is exponential in this length—recall that the multi-label classification task of the previous chapter had an output space exponential in the number of class labels, but constant with respect to the size of the input. Because of the exponential nature of the output, inference for sequence labelling by exhaustive search may only be feasible in the case of either short sequences, or few different labels.

The most popular alternative to exhaustive search is the Viterbi algorithm, as used for example by conditional random fields and structured perceptrons for sequence labelling, methods with which we have compared constraint satisfaction inference in the experiments of this chapter. The Viterbi algorithm provides a linear-time alternative to the exponential time complexity of exhaustive search, and is optimal under the Markov assumption. Unfortunately, while enabling efficient search, this assumption also severely limits the expressiveness of the prediction model: it does not allow modelling global dependencies in the output sequence. Indeed, optimal polynomial-time search is only possible under restrict-
4.7. Conclusion

ing assumptions such as the Markov assumption. Only exhaustive search allows for modelling of arbitrary dependencies.

With the approach to sequence labelling chosen in this chapter, we managed to perform inference by exhaustive search without harming run-time performance too much. This is made possible because inference is not performed in the full output space of the sequence labelling task, but in a reduced output space defined by the overlapping trigram predictions of a base classifier. In the worst-case, this still leaves the inference with $3^n$ sequences to consider, but as can be seen in Figure 4.6, accurate base classifier predictions tend to yield a much more favourable situation. In fact, inference for all four tasks performed in this chapter has been performed by exhaustive search, which did not cause a severe performance bottleneck. While we did not do so in the experiments, this allows for arbitrary additional constraints to be integrated into the inference procedure, a possibility that is incompatible with Viterbi search.

Obviously, restricting the search space of the inference procedure risks excluding too many good solutions, thereby putting a suboptimal upper bound on the sequence labelling performance. However, as illustrated by both the oracle scores and the true performance of the constraint satisfaction inference approach in this chapter, accurate base classifiers restrict the search space in such a way that state-of-the-art performance remains well within reach. It can therefore be concluded that constraint satisfaction inference for sequence labelling meets our objective of efficient inference without restricting assumptions.
Recognising syntactic structure in language is one of the most commonly used forms of abstraction in natural language processing. In the previous chapter, partial syntactic processing tasks such as part-of-speech tagging and shallow parsing have been treated. In this chapter we aim for more complete forms of syntactic processing, or full syntactic parsing. In syntactic parsing, a sentence is analysed according to the structure of some grammar formalism, resulting in a syntactic annotation of the words in the sentence. Mostly originating in traditional linguistics, a wide diversity of grammar formalisms exist, and many of those have also been subject to automatic processing. Two of the grammar formalisms that are found most frequently in natural language processing are constituent parsing and dependency parsing. The latter has seen an increasing interest in recent years and will also be the topic of this chapter.

Whatever the specific grammar formalism chosen, syntactic parsing is considered an important preprocessing step in many higher level processing tasks. Most semantic analysis tasks benefit from having access to the syntactic structure of their input. Punyakanok et al. (2008) show the importance of syntax for semantic role labelling. Zhang et al. (2006) demonstrate the improvement attainable by adding syntactic information to a relation extractor. As a last example of the importance of syntax, we mention syntax-based machine translation (e.g. Yamada and Knight, 2001; Gildea, 2003; Quirk et al., 2005), which uses syntax for mapping from source to target language. In addition to the examples mentioned here, many more applications for syntactic parsing can be found.

Dependency grammars define the syntactic structure of a sentence in terms of head-modifier dependency relations between pairs of words, where words modify other words with relations of a certain type. For example, the subject of a verb is expressed by a subject relation pointing from the head verb to the modifying word that is the subject of that verb. Every syntactic function within a sentence is expressed by such relations. In the formalism used in this chapter, each word is restricted to modify at most one other word, though a head word may have multiple modifiers. As a consequence of this restriction, the complete dependency
structure, or dependency graph, of a sentence has the form of a tree.

5.1 Existing work

Automatic syntactic parsing has long been the domain of rule-based approaches. Special-purpose parsing algorithms were equipped with manually written grammar rules to find the syntactic structure of a sentence. Building parsers this way requires a tedious manual grammar engineering step, which may complicate adapting them for e.g. different domains, or even languages. Due to the complexity of full syntactic parsing, successful machine learning approaches to the task have long remained rather sparse. However, the advance of structured prediction approaches does make full syntactic parsing an attainable goal, and as a result, many approaches to syntactic parsing in general, and dependency parsing in specific have arisen. This section provides an overview of two dominant approaches in data-driven dependency parsing: transition-based models and graph-based models.

5.1.1 Transition-based models

Popularised by Nivre et al. (2004), transition-based models for dependency parsing employ machine learning to turn non-deterministic parsing algorithms into deterministic versions. While processing the input sentence, the dependency graph is built incrementally, starting with an empty tree at the start, completing it after the last input token has been processed. For the case of dependency parsing, the general class of parsing algorithms that can be used in the context of transition-based models is described as maintaining two data structures. The first is a stack containing partially processed tokens. During parsing, the stack is manipulated as a result of actions taken by the parsing algorithm. The contents of the stack serve as a history of the parsing process for the algorithm to base its next decisions on. The second data structure used in the parsing algorithm is a sequence of remaining input tokens. At the start of the parsing process, this sequence comprises the complete input sentence. Then, the parsing algorithm moves over the sentence from left to right, processing each input token at a time, and removing it from the sequence of remaining tokens.

As an example of a specific parsing algorithm usable in the context of transition-based models, we briefly review the algorithm introduced by Nivre (2003). As explained above, the algorithm makes a single left-to-right pass over the input sentence. At each word, it makes a single deterministic decision, which for this algorithm is one of the following three options.

- **Left-Arc**, adds a dependency relation from the next input token to the token on top of the stack and pops the latter off the stack;
• **Right-Arc**, adds a dependency relation from the token on top of the stack to the next input token and pushes the latter onto the stack;

• **Reduce**, pops the token on top of the stack;

• **Shift**, pushes the next input token onto the stack.

An important restriction of this algorithm is that the dependency trees it produces can only be projective. For languages with non-projective relations, this may hinder performance. Note that this is a restriction of this specific parsing algorithm, not of transition-based models in general. Nivre (2007) suggests the Covington algorithm (Covington, 2001) for non-projective parsing in transition-based models. An alternative is to use graph transformations to convert non-projective dependency structures to projective ones (Nivre and Nilsson, 2005). Applying this conversion to the training data of the projective parser results in a projective parser of which the output can be converted to non-projective trees by applying a reverse-transformation.

The parsing algorithms used in transition-based models are similar, or in cases even the same as those used by rule-based systems. However, whereas grammar-based parsers explore every possible grammatical derivation of a sentence, transition-based models take a single non-deterministic decision at each choice point. This decision, which in case of the above algorithm is one of the four parsing actions mentioned, is taken by a function that bases itself on the history, reflected by the contents of the stack, and the remaining input tokens. State-of-the-art results are obtained when machine learning is employed to learn this decision function.

### 5.1.2 Graph-based models

The second family of popular learning approaches for dependency parsing is that of graph-based models. In essence, this method is about learning a global score function that evaluates the quality of every possible parse tree for the input sentence. The most common approaches within this framework use the structured linear models introduced in Chapter 2. As in all such models for structured outputs, the target structure, i.e. the dependency graph, is factorised, and local score functions are learned on the resulting factors. The specific factorisation that is chosen is mainly a trade-off between expressive feature representations and efficient inference.

First-order factorisations of a dependency graph compute local scores on all pairs of words. All head-modifier dependency relation candidates are scored, and subsequently the parse tree that optimises the scores of its dependency edges is determined. A major advantage of first-order factorisation is that it allows for efficient inference. Eisner (2000) introduces a dynamic programming parsing algorithm for first-order inference that is guaranteed to find the best projective tree for
Chapter 5. Dependency parsing

a sentence of $n$ tokens in $O(n^3)$ time. Moreover, even non-projective parsing—which has a considerably larger output space—can be performed efficiently in first-order models. McDonald et al. (2005) formulate first-order non-projective parsing as finding the maximum-spanning tree of the complete graph connecting all words, and propose to use the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967), which runs in $O(n^2)$ time for dense graphs (Tarjan, 1977).

Although first-order factorisations are attractive from an efficiency point of view, the restricted feature space available to the local scoring functions might be problematic. For this reason, several studies have experimented with second-order factorisations. Such factorisations make it possible to score larger windows on the output structure. McDonald and Pereira (2006) present results obtained with a second-order factorisation that scores pairs of adjacent dependency edges. Carreras (2007) experiments with a factorisation that allows for an even richer feature representation that also considers children of the modifier. Both studies report that the extra effort of second-order factorisation pays off in terms of parsing accuracy. Unfortunately, the increased expressiveness comes at the cost of a severe performance penalty during inference. McDonald and Pereira (2006) prove that second-order non-projective inference is NP-hard and is therefore unlikely to be solvable efficiently. Both McDonald and Pereira (2006) and Carreras (2007) do propose modifications to the Eisner algorithm for efficient second-order projective parsing. In addition, McDonald and Pereira (2006) suggest an approximate non-projective parsing strategy, which comes down to running a hill-climbing search after the second-order projective algorithm.

5.2 Constraint satisfaction approach

The application of constraint satisfaction inference to dependency parsing will result in an approach that can be considered a member of the family of graph-based models. As other such methods, the constraint solver will search the space of dependency trees guided by a global score function, as opposed to transition-based models, which build the parse tree incrementally. Compared to the most-popular graph-based method, structured linear models, the details of the score function are different, though, as are the exact boundaries of the search space of the inference—we discussed this difference in more general terms in Chapter 3. To describe our constraint satisfaction approach, we start with defining the solution space of the dependency parsing task. After that, we introduce three types of soft constraints that jointly give rise to the global score function. Finally, we discuss how best to solve the resulting constraint satisfaction problem.

5.2.1 Solution space

In order to define our output space, i.e. the space of all possible dependency trees having the same number of words as the input sentence, we start from
an imaginary complete directed graph $G$. Each word $w_i$ in the sentence has a corresponding node $i$ in this graph, which in addition includes another node, 0, for an artificial root token, from which the dependency tree is to originate. The directed edges in $G$ are potential head-modifier relations. As in other graph-based approaches, the goal is to find a subgraph $T$ of $G$, such that $T$ is a valid tree spanning every node of the graph and rooted in node 0. From this model, the micro-label representation of the output space can be derived.

Let $y = \{y_{00}, y_{01}, \ldots, y_{nn}\}$ be the connectivity matrix of the complete directed graph $G$, where $n$ is the number of tokens in the input sentence. Then, $y_{ij} = d$ means that word $j$ modifies word $i$ with a dependency relation $d$, or $d$ can have the special value $\texttt{norel}$, which denotes no relation exists between the two given words. Without restrictions, $y$ covers a superset of the actual output space for dependency parsing. Many joint micro-label assignments exist that do not represent a valid tree, and should therefore be excluded from the solution space of the inference. This restriction is the most important structural constraint on the output space, and is naturally integrated in the constraint satisfaction formulation.

5.2.2 Constraints

From a theoretical point of view, the tree constraint on the output structure is naturally integrated in constraint satisfaction inference as a knowledge-based hard constraint. For example, Prosser and Unsworth (2006) describe constraints for modelling trees in constraint satisfaction problems. As we will explain later, the approach described in this chapter will not actually solve the full CSP with this constraint. Instead, we will use a solving algorithm that only searches the subset of the output space that corresponds to valid trees, and thus, implicitly satisfies this constraint.

In addition to the abovementioned hard constraint that has to be satisfied by any candidate solution, we are going to define and predict additional, weighted soft constraints that help the constraint solver find the best parse tree for the input sentence. Unlike the approach to sequence labelling in the previous chapter, based on a single $n$-gram constraint type, we devise several structural constraints to tackle the parsing task, so that each constraint relates to a different aspect of the output structure. Note that there are no real guidelines or restrictions to what constraints are added to the inference procedure. The only requirement is that classifiers can be trained for predicting them. For the dependency parser presented in this chapter, three constraints will be used: a dependency constraint, a modifiers constraint, and a direction constraint. These constraints are illustrated graphically in Figure 5.1 and explained in more detail below.
Chapter 5. Dependency parsing

(a) Dependency constraint. Each ordered pair of tokens is classified for the existence of a dependency relation. Positive classifications are turned into a constraint; negative \texttt{norel} classifications (printed in grey) are not.

(b) Modifiers constraint. Each word is mapped to its subcategorisation frame. Only those predictions that suggest modifying arguments are turned into a constraint; empty frames (printed in grey) are not.

(c) Direction constraint. Each word is classified for the relative position of its head word. Each of the three possible outcomes—left, right, and root—is made a constraint.

Figure 5.1 The three kinds of constraints used for dependency parsing and the classification cases that predict them, illustrated in the context of the sentence “John likes Mary”. 
5.2. Constraint satisfaction approach

Dependency constraint

The dependency constraint is a straightforward constraint on the existence of a dependency relation of a certain type between two words in the sentence. It is illustrated in Figure 5.1(a), where an optimal classifier would give rise to three constraints, requiring a SBJ relation between “John” and “likes”, a VMOD relation between “Mary” and “likes”, and finally, a ROOT relation between “likes” and the artificial root token. The base classifier responsible for predicting this constraint operates on an input window that corresponds to a pair of words, one of which is the potential head, the other, the potential modifier. Predictions are made for every ordered pair of words in a sentence, and correspond to either a dependency relation type, if the two candidates are indeed related, or a negative label, norel, indicating there is no dependency relation between the two words. After classification, dependency constraints are only created for positive—that is, non-norel—predictions. Thus, only constraints that require a certain relation are created, we choose to create no constraints that require the absence of one.

Formally, the constraint derived from a predicted dependency relation $d$ simply requires the corresponding micro-label to have the predicted value, i.e.

$$\text{Constraint: } y_{ij} = d$$

Since the predictions made by this base classifier correspond directly to the micro-labels of the output space, it can also be employed as a domain-defining base classifier for the constraint satisfaction problem. In fact, it is the sole domain-defining classifier in our approach to dependency parsing; the two remaining base classifiers predict constraints only. The precise domain definition for each output variable $y_{ij}$ extracted from the dependency predictions will be the set comprising the negative label norel, and the prediction made by this classifier. If it is empty as well, a special symbol unknownrel is added as a domain value, so that the inference procedure will always be able to recover a fully connected dependency tree, even if the dependency predictor generated too few dependencies for this.

$$D_{ij} = \{\text{norel}\} \cup \{d\}$$

Modifiers constraint

The modifiers constraint models the modifying arguments of words. In the example sentence of Figure 5.1(b), the word “likes” has modifying relations of two different types, which will be modelled by two constraints, stating that the word “likes” is modified by relations of type SBJ and VMOD respectively. Essentially, this modifiers constraint corresponds to the syntactic notion of subcategorisation; a subcategorisation frame being a specification of the types of arguments a word in a given sense occurs with. Unlike the traditional practice, however, we do not extract modifiers constraints from a subcategorisation database, but predict them with a base classifier instead.
This base classifier operates on input windows of a single word only. The target value corresponds to the subcategorisation frame of the input word, that is, all relation types that modify the word. There are several ways to extract constraints from predicted subcategorisation frames. One possibility is to require the exact subcategorisation frame to match its predicted value. A disadvantage of this choice, however, is that prediction errors might make this constraint less effective, or even harmful. Since predicting the exact subcategorisation frame might be a difficult task, we opt for lower-level constraints. Instead of introducing one constraint that requires the complete subcategorisation frame to match, we introduce a constraint for each of the relation types in the predicted frame. For example, a predicted frame SBJ/VMOD results in two constraints being added to the inference, one requiring the word to be modified by a subject, another that expects a verbal modifier for the word. Unlike the previous option, it is possible to match one constraint only, while leaving the other unsatisfied.

**Constraint:** \( \exists j y_{ij} = d \)

Modifiers constraints are weighted by the total weight an argument type receives in the predicted class distribution. For example, a constraint involving the argument type SBJ extracted from the predicted class SBJ/VMOD is weighted with the sum of weights of all classes in the class distribution that include the argument type SBJ, and thus, not only with the weight of the class SBJ/VMOD.

An interesting observation that can be made is that predicting the subcategorisation frame of a verb is actually a multi-label classification task. In Chapter 3, we developed a constraint satisfaction approach to multi-label classification. This approach could also be used for this constraint. In the same chapter, it could also been seen that multi-label classification with a relatively small number of micro-labels can be performed quite reliably by a single macro-label predictor. To reduce the complexity of our parser, that is indeed the approach we choose here.

**Direction constraint**

This last constraint again operates on the word level, but unlike the previous constraint, considers the modifier role of the word. For each word in the sentence, a constraint is introduced that reflects the relative position of the head word it modifies; possible values being left, right, and root. The latter means the word does not modify any other word in the sentence, but instead, modifies the artificial root token. In the example sentence in Figure 5.1, this is the case for the word “likes”. Furthermore, “John” should be constrained to modify a word to its right, and “Mary” a word to its left.

As with the previous two constraints, the actual values are to be predicted by a base classifier. Unlike in those two cases, though, we extract constraints from the complete class distribution, rather than just from the single-best prediction. This means that for each word, between one and three soft constraints, weighted
by their corresponding score in the class distribution, are added to the constraint satisfaction problem. The form of those constraints is straightforward, we simply require the suggested relative position to be true in the parse tree. For example, the following constraint requires that word $w_i$ modifies a head word to its right.

**Constraint:** \( \exists j \neq \text{norel} \land i < j \)

### 5.2.3 Solving the CSP

With the constraints defined in the previous subsection, the dependency parsing task is cast as a weighted constraint satisfaction problem. As was done for sequence labelling in the previous chapter, a standard constraint satisfaction solving method could be used to find the optimal dependency tree. However, in comparison with sequence labelling, the output space of dependency parsing is considerably larger. For this reason, it might not be preferable to revert to exponential-time solving methods. Besides, the tree constraint on dependency structures yields an important restriction on the output space, and it is expected that solving algorithms specifically designed to operate in this restricted space will do better than general-purpose CSP solvers that consider the tree constraint as just one of the constraints to enforce.

The above considerations resulted in the choice for standard parsing algorithms to solve the WCSP. More specifically, we opted for the Eisner parsing algorithm (Eisner, 2000). As we explained in Section 5.1.2, the choice for this algorithm restricts the expressiveness of the underlying model. For constraint satisfaction inference, it implies that constraints may cover single words, or pairs of words only. The constraints suggested in the previous section adhere to that restriction, but if more global constraints are added to the inference, other solving algorithms will need to be used. As a more serious restriction, the parses produced by the solver will only be of the projective kind. This may cause a performance penalty for languages with a high degree of non-projective dependencies. Nevertheless, even for those languages, most of the dependency relations are still projective. Therefore, we will restrict ourselves to projective parsing for the experiments in this chapter. Other solving algorithms may be employed if non-projective parsing is desired.

### 5.3 Experimental setup

#### 5.3.1 Data

An extensive collection of data sets for the evaluation of data-driven dependency parsers has been provided as part of the 2006 and 2007 CoNLL shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007). Data sets with annotations for dependency structure were provided for, in total, 19 languages. An interesting property of those data sets is that they show a great deal of variety, some caused
Table 5.1  Overview of the ten languages comprising the 2007 CoNLL shared task multilingual track. The treebank column lists the treebanks from which the data sets were obtained; the exact details of the data preparation procedure are described by Nivre et al. (2007).

<table>
<thead>
<tr>
<th>Language</th>
<th>Treebank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>Hajić et al. (2004)</td>
</tr>
<tr>
<td>Basque</td>
<td>Aduriz et al. (2003)</td>
</tr>
<tr>
<td>Catalan</td>
<td>Martí et al. (2007)</td>
</tr>
<tr>
<td>Chinese</td>
<td>Chen et al. (2003)</td>
</tr>
<tr>
<td>Czech</td>
<td>Böhmová et al. (2003)</td>
</tr>
<tr>
<td>English</td>
<td>Marcus et al. (1993); Johansson and Nugues (2007)</td>
</tr>
<tr>
<td>Greek</td>
<td>Prokopidis et al. (2005)</td>
</tr>
<tr>
<td>Hungarian</td>
<td>Csendes et al. (2005)</td>
</tr>
<tr>
<td>Italian</td>
<td>Montemagni et al. (2003)</td>
</tr>
<tr>
<td>Turkish</td>
<td>Oflazer et al. (2003)</td>
</tr>
</tbody>
</table>

by properties of the languages themselves, others by treebank annotation choices. Some of the most evident differences in the former category include the average length of sentences, the proportion of non-projective relations, and the morphological richness of the language. The latter category accounts for differences in such properties as the number of different dependency types, whether word lemmas are provided, and the amount of morphological structure explicitly annotated. In spite of these differences, though, data-driven parsers have shown to be able to perform well on all languages, although large differences still exist in state-of-the-art performance among the languages. For the experiments presented in this chapter, we have opted for the ten data sets used in the 2007 CoNLL shared task; the ten languages, and references to the treebanks the data sets were extracted from are listed in Table 5.1.

Although the core learning techniques for data-driven dependency parsing have shown to be adaptable to a wide variety of languages and annotation schemes, there is still a feature engineering effort involved in tuning a parser for any given language. For the experiments in this chapter, we apply constraint satisfaction inference dependency parsing to all ten languages of the abovementioned collection. We will refrain from language-specific optimisations involving heavy feature engineering, but instead choose fixed feature sets to use for all languages.

5.3.2 Evaluation

All experiments reported upon in this chapter have been performed following the same procedure as prescribed by the CoNLL 2006 and 2007 shared tasks. More specifically, the same data partitions have been used for training and testing, and
5.3. Experimental setup

no external knowledge sources have been exploited for extracting richer features than those available in the provided treebanks. In addition, the same evaluation procedure is adopted, which computes three performance scores.

*Labelled attachment score* (LAS) computes the proportion of tokens that has been assigned both the correct head word and the correct dependency relation. *Unlabelled attachment score* (UAS) looks at correct head word assignments only and ignores the predicted dependency type. In contrast, *label accuracy* (LA) counts the proportion of tokens that have been assigned the correct dependency relation, while ignoring the predicted head word.

In computing these scores, all tokens are taken into account. This matches the procedure followed in the 2007 CoNLL shared task, whereas in 2006, punctuation tokens were not considered in the evaluation.

5.3.3 Constraint prediction

With the exception of the tree constraint, which is a hard constraint implicitly enforced by the solver, the constraints designed in Section 5.2.2 require a classifier that predicts the precise values to be enforced. In this subsection, we describe the feature design and additional considerations relevant to training those classifiers.

**Dependency constraint**

The classifier responsible for predicting the dependency constraints plays two roles in the parsing process. Like the other two base classifiers, it is constraint defining, and thus, its predictions are added to the inference procedure as constraints. In addition, and unlike the other classifiers, it is domain defining: the exact boundaries of the output space in which the solver will search for a solution are the result of the predictions made by this classifier. This makes the quality of the constraint predictions critical to the performance of the complete system. Error predictions will not only result in incorrect constraints being added to the inference—an issue that may still be overcome by other constraints—it may also restrict the output space in such a way that the correct output is simply not reachable by the solver. We will look into the implications of this in more detail, but first we give an overview of the features designed for the classification.

The feature set used for building the constraint predictor is depicted in Figure 5.2. It can be divided in two parts. The first part consists of features extracted from either the head word or the modifier. For both words, those features are the word forms and part-of-speech tags of the words themselves and of the two words directly surrounding them. The second part of the feature set includes features of the combination of the head word and modifier. Conjunctions are made of their part-of-speech tags and morphological feature sets. In addition, the distance, i.e. number of words, between the head and modifier, and the relative position of the head with respect to the modifier are extracted.
Chapter 5. Dependency parsing

Figure 5.2 Feature representation used by the dependency predictor, illustrated on the basis of the Italian example sentence “Vi sono migliaia di casi di questo genere.” and the candidate head-modifier pair “migliaia”, “questo”.

The class to be predicted corresponds to the label of the dependency relation between the candidate head word and modifier, or alternatively signals that there is no such relation. Note that for each pair of words, two instances are classified, where both words are the candidate head word once, while the other word is the candidate modifier. Processing a sentence this way leads to a number of classifications quadratic in the length of the sentence. Since each word can only modify one other word in the sentence, the majority of instances in the training data will in fact be negative instances, and all non-negative instances will be vastly outnumbered by them. This will typically have the negative side-effect that a classifier trained on these data will be inclined to classify instances as negative, unless the evidence is particularly strong in favour of a non-negative classification. As a result, precision will be high, but it will be so at the cost of recall, which will be low. In the implementation of the constraint predictor, we take several measures to counteract this effect. The first of those is that the maximum distance between a modifier and its head word is limited to a fixed number of words. This distance limit is chosen such that 99.0% of all dependencies in the training data fall within it.

Even within this distance limit, however, most of the instances in the training data will still be negative. Limiting the maximum distance will therefore mainly serve to speed up the parser, but the issue of low recall will not be resolved. To see why this is problematic, consider its effect on the solution space. Since the dependency predictor is also responsible for constructing the domains of the micro-labels, low recall means that the correct value for many variables will not
be a member of that variable's domain, and therefore can never be recovered by
the constraint solver. In the light of this observation, it is much more preferable
to have high recall at the cost of precision, than the other way around. The
constraint that any word modifies at most one head word may compensate for
over-predicting dependency relations—which is the consequence of low precision—
but missed dependencies as a result of low recall cannot be compensated for. With
this objective in mind, we will down-sample negative instances in the training
data.

To measure the effect of down-sampling on the eventual parsing performance,
we evaluated several configurations of the English dependency parser. For this,
the constraint predictor has been trained on training data subject to increasing
degrees of down-sampling. The original negative to positive ratio of the English
training data is 25 to 1; we start down-sampling at a ratio of 15 to 1, and then
increase the ratio gradually. Figure 5.3 shows the precision and recall of the
dependency predictor, and the labelled attachment score of the resulting parser,
both using dependency constraints only, and using all three constraint types.
The typical effect of down-sampling on the performance of the constraint predictor is evident from the precision and recall curves, which are decreasing and increasing, respectively. Recall increases steadily, up to 95.0% when the number of negative training instances equals the number of positive ones. At this down-sampling ratio precision has dropped to a mere 42.1%. What interest us more, though, is the effect on the labelled attachment score. Here, we see that parsing performance improves with increasing recall, even though precision drops severely. The parser based on dependency constraints only peaks at a down-sampling ratio of three negative instances for every positive one and degrades slightly with stronger down-sampling. Interestingly, the parser with all three constraint types continues to improve, even up to the point where only as many negative instances as positive ones remain. Therefore, in the remaining experiments, this is the down-sampling ratio we will use for all languages.

Modifiers constraint

As suggested in Section 5.2.2, predicting the modifiers constraint is effectively a multi-label classification task. As such, there is a variety of approaches that can be used to learn this classifier. Here, the straightforward approach of predicting all modifying relations for a word as a single class is chosen. Doing this, it may be expected that low-frequent dependency types suffer from low recall. Although a higher recall is certainly preferable, missing some relations is not critical. It will result in constraints not being added to the inference. At the same time, this will not penalise adding a relation of the corresponding type to the dependency graph anyway; it will simply not encourage it either. It would in fact be worse if recall were to improve at the cost of precision. This would lead to an over-prediction of constraints encouraging dependency relations of certain types. In this respect, the precision/recall trade-off of this classifier is the exact opposite of that of the dependency predictor.

The feature set for this classifier, illustrated in Figure 5.4, includes various representations of the focus word and its context. First, the word forms and part-of-speech tags of the word itself and a window of the two words preceding and the two words following it. In addition, conjunctions of part-of-speech tags in that same window of lengths two and three are added. Finally, there is a feature for the morphological feature set of the word in focus.

Direction constraint

Conceptually, the direction constraint is the easiest constraint to predict. The target value is one out of only three possible classes: left, right, or root. An example of the feature representation used with the direction predictor is shown in Figure 5.5. The features again form a representation of the focus word and its context. Word forms and part-of-speech tags are extracted from a window spanning from two words before to two words following the focus word. In addition,
5.4 Results

Although evaluating the performance of the full parser is interesting in its own right, it is important to realise that this performance is the result of complex interactions between the various constraints that are active in the parsing process, whether those constraints are predicted or fixed in advance, as is the tree constraint. For this reason, this overview of experimental results starts with an evaluation of the individual classification performances of the constraint predictors, and of their contribution to the performance of the parser.
Chapter 5. Dependency parsing

Table 5.2  Parsing performance based on the dependency constraint. Left: the precision and recall of the dependency predictor. Middle: the upper bound, oracle performance yielded by the given dependency predictions. Right: the performance of the constraint satisfaction inference parser based on the predicted dependency constraints.

<table>
<thead>
<tr>
<th>Language</th>
<th>Dependencies</th>
<th>Oracle</th>
<th>CSI\textsuperscript{dep}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>precision</td>
<td>recall</td>
<td>LAS</td>
</tr>
<tr>
<td>Arabic</td>
<td>38.22</td>
<td>83.17</td>
<td>82.53</td>
</tr>
<tr>
<td>Basque</td>
<td>39.46</td>
<td>80.18</td>
<td>79.52</td>
</tr>
<tr>
<td>Catalan</td>
<td>42.87</td>
<td>91.74</td>
<td>90.77</td>
</tr>
<tr>
<td>Chinese</td>
<td>55.49</td>
<td>87.67</td>
<td>87.39</td>
</tr>
<tr>
<td>Czech</td>
<td>39.52</td>
<td>85.67</td>
<td>84.91</td>
</tr>
<tr>
<td>English</td>
<td>42.62</td>
<td>95.08</td>
<td>94.32</td>
</tr>
<tr>
<td>Greek</td>
<td>35.65</td>
<td>84.24</td>
<td>83.22</td>
</tr>
<tr>
<td>Hungarian</td>
<td>31.70</td>
<td>87.34</td>
<td>87.09</td>
</tr>
<tr>
<td>Italian</td>
<td>44.75</td>
<td>89.49</td>
<td>88.78</td>
</tr>
<tr>
<td>Turkish</td>
<td>49.49</td>
<td>83.44</td>
<td>82.92</td>
</tr>
</tbody>
</table>

5.4.1 Dependency constraint

In Subsection 5.3.3, we already pointed at the critical role of the dependency constraint predictor in the parsing process. This is due to the fact that the dependency predictor is the domain-defining classifier, and as such is responsible for defining the solution space in which the constraint solver will search for the output structure. Because of this, it has been decided to optimise the predictor for recall, even if at the cost of precision. For all languages, negative examples in the training data have been down-sampled to a ratio of one negative example for every positive one. After that, base classifiers have been trained on the training data thus obtained. The classification results for the ten languages are listed on the left-hand side of Table 5.2.

For all languages, the recall is considerably higher than the precision, as was the intended result of the down-sampling. Recall should be as high as possible, since it imposes an upper bound on the performance of the parser. Given these results, therefore, it is insightful to see what upper bound on the parsing performance is implied by these scores. For this, the oracle inference has been run based on these classifier outputs. The resulting parsing scores are listed in the middle columns of Table 5.2. Considering these scores, there are a few issues to be clarified. First of all, one might expect the labelled attachment score to be equal to the recall of the dependency predictor. However, the dependency predictor only classifies word pairs at a fixed maximum distance of one another, and therefore is guaranteed to miss a small proportion of dependencies. This guaranteed loss is not reflected in the scores of the classifier, but the parsing scores do show...
5.4. Results

<table>
<thead>
<tr>
<th>Language</th>
<th>Precision</th>
<th>Recall</th>
<th>$\text{CSI}^{\text{dep},\text{mod}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>LAS</td>
</tr>
<tr>
<td>Arabic</td>
<td>77.49</td>
<td>66.25</td>
<td>71.58</td>
</tr>
<tr>
<td>Basque</td>
<td>81.96</td>
<td>70.12</td>
<td>68.37</td>
</tr>
<tr>
<td>Catalan</td>
<td>89.31</td>
<td>80.57</td>
<td>82.30</td>
</tr>
<tr>
<td>Chinese</td>
<td>87.36</td>
<td>75.77</td>
<td>80.49</td>
</tr>
<tr>
<td>Czech</td>
<td>77.62</td>
<td>66.69</td>
<td>72.12</td>
</tr>
<tr>
<td>English</td>
<td>91.34</td>
<td>88.21</td>
<td>82.69</td>
</tr>
<tr>
<td>Greek</td>
<td>77.65</td>
<td>68.23</td>
<td>70.50</td>
</tr>
<tr>
<td>Hungarian</td>
<td>69.43</td>
<td>60.88</td>
<td>70.41</td>
</tr>
<tr>
<td>Italian</td>
<td>87.37</td>
<td>80.14</td>
<td>79.04</td>
</tr>
<tr>
<td>Turkish</td>
<td>85.13</td>
<td>60.21</td>
<td>72.61</td>
</tr>
</tbody>
</table>

Table 5.3 Parsing performance after adding the modifiers constraint. Left: the precision and recall of the modifiers predictor. Right: the performance of the constraint satisfaction inference parser based on the predicted dependency and modifiers constraints.

it. Secondly, the unlabelled attachment scores of all languages are greater than the recall of the dependency predictor. Again, this might seem counter-intuitive. However, the reported recall scores are labelled recall scores only. The oracle inference is implemented in such a way that it still adds a dependency to the parse if the predicted label is incorrect, but the predicted head is correct. This behaviour does not affect labelled attachment score, but can be observed in the unlabelled attachment score.

The Oracle scores listed in Table 5.2 give the upper bound of parsers based on the given set of dependency predictions. Now, let us have a look at what scores are attained by constraint satisfaction inference when only the tree constraint and the predicted dependency constraints are active, and when the domains are constructed from the dependency predictions. The right-hand side of Table 5.2 shows these results. Unsurprisingly, the performance is worse than that of the oracle. The difference in labelled attachment score amounts to approximately 10% to 13%, except for Hungarian, where the difference is larger, probably because the imbalance between recall and precision of the base classifier predictions is too large. Nevertheless, constraint satisfaction inference manages to build parse trees from a set of conflicting constraints that are at least reasonably accurate. For further performance improvement, the remaining constraints will be added to the inference.

5.4.2 Modifiers constraint

For the modifiers constraint, classifiers have been trained on the unaltered training set; no training examples have been removed as was done for training the
dependency predictor. The precision and recall of this classifier for the ten languages are shown on the left-hand side of Table 5.3. Note that precision and recall are noticeably out of balance, in favour of precision. As argued in Section 5.3.3, this is the desired trade-off. Since the modifiers predictor is not domain-defining, there will be no change in the upper bound of the parsing performance, which will therefore remain unchanged from the scores reported in the previous subsection. The role of the modifiers predictions and the constraints derived from them is to guide the inference procedure to select better values from the domains constructed by the dependency predictor. This effect is illustrated by the scores reported on the right-hand side of Table 5.3, which shows the results obtained by enabling both the dependency and modifiers constraints in the inference.

Adding the modifiers soft constraint improves the performance for all languages but Turkish. It is important to emphasise that the search space of the inference is unmodified from the previous experiment. For nine of the languages, the extra soft constraint derived from the modifiers predictions thus helps the parser make better decisions given the same solution space. This does not seem to be the case for Turkish, though. Labelled and unlabelled attachment scores barely improve, and label accuracy even drops slightly. Most likely, the cause of this is to be found in the large difference between precision and recall, which makes that few modifier constraints are actually added. Those that are, apparently add little additional information to the dependency constraints.

### 5.4.3 Direction constraint

Classifiers for the direction constraint have also been trained on the unaltered training set. The accuracy scores for the direction predictions are listed on the left-hand side of Table 5.4. Interestingly, the direction constraint is predicted fairly accurately using only basic features. This is especially the case for Arabic and Catalan with an accuracy of approximately 95%, and for Turkish, which attains close-to-perfect performance. The latter may be explained by the observation that the vast majority of tokens in Turkish—approximately 83% in the training data—modify a head word located to their right in the sentence. For the other languages, direction predictions are rather accurate as well, with the weaker performances around 85%. However, it is more important whether adding the direction constraint to the inference improves parsing performance. First, we experiment with a parser that only uses the dependency and direction constraints. The performance scores of this parser can be found in the middle columns of Table 5.4. For most of the languages, there is improvement compared with the parser based on dependency constraints only. The magnitude of this improvement tends to be comparable or slightly larger than the improvement attained with the modifiers constraint. Two languages, Chinese and English, do not seem to benefit from the direction constraint, however. Adding the direction constraints to the dependency constraints does not improve the performance of the parser.
Table 5.4 Parsing performance after adding the direction constraint. Left: the accuracy of the direction predictor. Middle: the performance of the constraint satisfaction inference parser based on the predicted dependency and direction constraints. Right: the parsing performance based on the predicted dependency, modifiers, and direction constraints.

5.4.4 Overall parsing performance

In the previous two subsections, we have shown how adding modifiers and direction constraints, respectively, to a dependency parser based on dependency constraints improves parsing performance for most of the ten languages. This illustrates the positive effect that predicted soft constraints can have on the uncertainty resolution performed by the inference procedure. Considering this positive effect of the two types of constraints individually, it is even more interesting to look at the combination of all three predicted constraints. The parsing performances thus obtained are reported on the right-hand side of Table 5.4. Here, it can be seen that combining all three constraints results in the best performance for all ten languages. Interestingly, the score increase with respect to the setup using only dependency constraints is in many cases larger than the sum of score increases of the modifiers and direction constraints separately. The most striking example is seen with English. For this language, the direction constraint did not seem to have any added value when used in combination with the dependency constraint. However, if combined with both the dependency and modifiers constraint types, the increase in labelled attachment score is almost twice as much as the increase attained by the modifiers constraint.

To summarise, Figure 5.6 shows the contributions of the individual constraint types, as well as their combinations. It can be seen that, in general, each of the constraint types has a positive contribution to the performance of the parser, and that the combination of all three types leads to the best performance. In some
cases, a constraint type may not have the intended effect. Such is the case with the modifiers constraint for Turkish, and with the direction constraint for Chinese. On the other hand, including those constraints anyway does not negatively affect the parser severely. We therefore conclude that the additional soft constraints serve their purpose by helping the inference procedure make better decisions. Most importantly, they result in improved parsing performance without enlarging the search space of the parsing algorithm, and thus, without increasing the overhead of the prediction framework.

**Comparison to other published results**

The primary aim of the experiments described in this chapter is to illustrate the application of constraint satisfaction inference to the task of dependency parsing, so that we will be able to relate the architecture of the resulting parser to the research objectives formulated in Chapter 1. Nevertheless, an important secondary aim is to compare the performance of a dependency parser based on constraint satisfaction inference with the performance of parsers based on alternative structured prediction frameworks.

Since we performed our experiments with standardised data sets, other results
for systems performing under the same experimental conditions are available with which we can compare. Here, we compare the performance of the constraint satisfaction inference parser with the performance of other participants in the 2007 CoNLL shared task. Table 5.5 lists the scores of the CSI parser alongside the averaged scores over all shared task participants, and the score of the best-performing parser for each language. First, looking at the averaged scores, we can see that the CSI-based parser performs around or well above average for all languages. On the other hand, when considering the best-performing parsers for each language, it is evident that a considerable gap still separates the CSI-based parser from the top-performing systems in the 2007 shared task.

The question then arises whether this difference is due to a limitation of the constraint satisfaction inference framework, or whether this particular instantiation of the parser—in particular, the features used for training the base classifiers—is suboptimal. In other words, could the CSI-based parser be optimised to perform on par with state-of-the-art parsers? We hypothesise that, at least, the largest part of the difference is not due a deficiency of constraint satisfaction inference. Its potential weakness is that the search space of the inference procedure is restricted so strongly, that better solutions are simply out of reach. The Oracle scores listed in Table 5.2, though, are above state-of-the-art for every language. Therefore, inference may still find parse trees that compare favourably with those found by better-performing parsers. The relative weakness, then, has to be looked for in the means used for guiding the inference, which in this case, are the constraints, which directly result from base classifier predictions. Although some effort has been invested in optimising these classifiers, we did not, nor did we intend to, exhaust the possibilities for optimisation. Again, looking at the 2007 shared task, parsers sharing similar architectures are found both at the top and at the bottom of the ranking. What distinguishes them is mostly a better set of features; for example, features encoding more morphological information, features that deal better with word sparsity, and features encoding more contextual information.

To conclude this discussion, we argue that the performance of the dependency parser developed in this chapter is sufficient to serve our purposes. First of all, we demonstrated how constraint satisfaction inference is employed to build a dependency parser. Its above-average performance suggests that the structural characteristics of the dependency parsing task are captured sufficiently by the constraint satisfaction approach. In addition, we have discussed that better performance, through optimisation, may realistically be expected. Secondly, the parser allows us to analyse its behaviour with respect to the research objectives of our study, which we will do in the next section.
(a) Labelled attachment scores

<table>
<thead>
<tr>
<th>Language</th>
<th>CSI</th>
<th>Avg</th>
<th>Best</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>73.01</td>
<td>68.34</td>
<td>76.52</td>
<td>(Hall et al., 2007)</td>
</tr>
<tr>
<td>Basque</td>
<td>69.83</td>
<td>68.07</td>
<td>76.94</td>
<td>(Hall et al., 2007)</td>
</tr>
<tr>
<td>Catalan</td>
<td>83.27</td>
<td>79.85</td>
<td>88.70</td>
<td>(Hall et al., 2007)</td>
</tr>
<tr>
<td>Chinese</td>
<td>80.64</td>
<td>76.59</td>
<td>84.69</td>
<td>(Sagae and Tsujii, 2007)</td>
</tr>
<tr>
<td>Czech</td>
<td>73.52</td>
<td>70.12</td>
<td>80.19</td>
<td>(Nakagawa, 2007)</td>
</tr>
<tr>
<td>English</td>
<td>83.83</td>
<td>80.95</td>
<td>89.61</td>
<td>(Carreras, 2007)</td>
</tr>
<tr>
<td>Greek</td>
<td>72.34</td>
<td>70.22</td>
<td>76.31</td>
<td>(Nakagawa, 2007)</td>
</tr>
<tr>
<td>Hungarian</td>
<td>72.36</td>
<td>71.49</td>
<td>80.27</td>
<td>(Hall et al., 2007)</td>
</tr>
<tr>
<td>Italian</td>
<td>79.98</td>
<td>78.06</td>
<td>84.40</td>
<td>(Hall et al., 2007)</td>
</tr>
<tr>
<td>Turkish</td>
<td>73.12</td>
<td>73.19</td>
<td>79.81</td>
<td>(Titov and Henderson, 2007)</td>
</tr>
</tbody>
</table>

(b) Unlabelled attachment scores

<table>
<thead>
<tr>
<th>Language</th>
<th>CSI</th>
<th>Avg</th>
<th>Best</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>82.36</td>
<td>78.84</td>
<td>86.09</td>
<td>(Nakagawa, 2007)</td>
</tr>
<tr>
<td>Basque</td>
<td>76.59</td>
<td>75.18</td>
<td>82.84</td>
<td>(Hall et al., 2007)</td>
</tr>
<tr>
<td>Catalan</td>
<td>87.52</td>
<td>87.98</td>
<td>93.40</td>
<td>(Titov and Henderson, 2007)</td>
</tr>
<tr>
<td>Chinese</td>
<td>85.68</td>
<td>81.98</td>
<td>88.94</td>
<td>(Sagae and Tsujii, 2007)</td>
</tr>
<tr>
<td>Czech</td>
<td>80.00</td>
<td>77.56</td>
<td>86.28</td>
<td>(Nakagawa, 2007)</td>
</tr>
<tr>
<td>English</td>
<td>84.87</td>
<td>82.67</td>
<td>90.63</td>
<td>(Carreras, 2007)</td>
</tr>
<tr>
<td>Greek</td>
<td>78.75</td>
<td>77.78</td>
<td>84.08</td>
<td>(Nakagawa, 2007)</td>
</tr>
<tr>
<td>Hungarian</td>
<td>76.27</td>
<td>76.34</td>
<td>83.55</td>
<td>(Hall et al., 2007)</td>
</tr>
<tr>
<td>Italian</td>
<td>83.89</td>
<td>82.45</td>
<td>87.91</td>
<td>(Nakagawa, 2007)</td>
</tr>
<tr>
<td>Turkish</td>
<td>79.79</td>
<td>80.33</td>
<td>86.22</td>
<td>(Titov and Henderson, 2007)</td>
</tr>
</tbody>
</table>

Table 5.5  Dependency parsing performance on the ten languages of the 2007 CoNLL shared task. The CSI columns report the scores obtained with the constraint satisfaction inference parser; the Avg and Best columns list the average score over all competing parsers and the score of the best parser—identified by its reference—for that language in the shared task competition.
5.5 Conclusion

In this chapter, we applied the constraint satisfaction inference framework to the task of dependency parsing. To relate our findings to the research objectives formulated in Chapter 1, we focus on the objective of efficient inference, since the other two objectives are implicitly satisfied by the constraint satisfaction framework in general, as we have explained in Chapter 3.

The importance of efficient inference increases with the magnitude of the output space of a task. For dependency parsing, this magnitude is by all means considerable. To see this, it is useful to distinguish two separate, yet interrelated subtasks of dependency parsing: the first aims to find the tree structure that connects the words of the input sentence; the second, labels the edges between words with a grammatical category. Cayley’s formula (Cayley, 1889) asserts that the number of spanning trees over \( n \) nodes, and thus the number of dependency trees covering \( n \) tokens, is \( n^{n-2} \). Since labelling an unlabelled tree, given \( L \) possible relation types, allows for \( L^{n-1} \) different labellings, the output space complexity of joint labelled dependency parsing is in the order \( O(n^nL^n) \) — obviously, a space that is impossible to search exhaustively.

For this reason, many of the graph-based approaches to dependency parsing perform the two subtasks sequentially, first identifying the tree structure, then labelling its edges. Unlabelled parsing allows for efficient dynamic programming solutions, and labelling the predicted trees comes down to conventional classification. Such two-stage approaches thus succeed in reducing the complexity of labelled dependency parsing; however, the decision to separate tree prediction and tree labelling results in a pipeline architecture (cf. Chapter 2). The disadvantages of this architecture discussed for the general case apply to the specific case of dependency parsing as well. Information only predicted during labelling might have been useful for the tree structure prediction, yet since the two stages are separated this information cannot be exchanged. The approach chosen in this chapter aims to do joint tree structure prediction and labelling, which results in an output space comprising not only every possible tree covering the input sentence, but every possible labelling of those trees as well.

The component of the CSI-based dependency parser that is responsible for the search space of the inference procedure is the domain-defining base classifier, i.e. the dependency predictor. For an ordered pair of two words, at most a single labelled edge is considered, the one predicted by the base classifier. If the base classifier makes a negative prediction for a pair of words, no labelled edge is even considered, though in this case, the domain will contain an unknownrel edge to enable the inference procedure always to find a dependency tree that fully spans the input sentence. As a result, for every ordered pair of words, only one labelled edge is ever considered by the inference procedure, and thus, dynamic programming parsing algorithms originally only suited for unlabelled parsing can be used for the joint task.

As seen in the previous chapters, the central role played by the domain-
defining base classifier makes the quality of its predictions critical with respect to the upper-bound performance of the inference procedure. On the one hand, the domain-defining base classifier should reduce uncertainty as much as possible—as this makes inference cheap. On the other hand, the search space should not be so restricted that the correct solution is no longer in it.

In this chapter, this trade-off has been illustrated by varying the down-sampling ratio of the training data for the dependency predictor (cf. Figure 5.3). If trained on the original training distribution, dependency predictions have high precision, but the search space yielded by them is often too restricted. Increasing the down-sampling ratio of the training data improves recall, and thereby improves the upper-bound performance of the parser. At the same time, though, the uncertainty to be resolved by the inference grows and the precision of the dependency constraints—the primary means of resolving uncertainty in CSI—drops. To compensate for this, additional constraints are introduced. The tree constraint, necessary for dependency parsing, already manages to compensate a great deal for the reduced precision of the dependency predictions. Nevertheless, if only the tree constraint and the dependency constraint are used, the performance of the parser peaks at a down-sampling ratio of three negative instances for each positive one. Stronger down-sampling degrades the labelled attachment score of the parser. Apparently, the uncertainty is too high to be resolved by the available constraints.

By adding two additional constraints, the modifiers and direction constraints, we managed to improve the performance of the parser, even at a down-sampling ratio that leaves the same number of positive and negative instances in the training data. This confirms that predicted soft constraints can make a positive contribution to uncertainty resolution in structured prediction, and leaves us with the perspective of further improvements by introducing even more types of constraints.
Chapter 6

Machine translation

Translating from one natural language to another can be seen as the ultimate natural language processing task. Performing it successfully requires a thorough understanding of many subtleties of both languages. For example, to translate a word correctly from Dutch to English, simply consulting a translation dictionary will not suffice. Instead, one first needs to determine the sense of the Dutch word in its current context. Only then can a suitable English translation be found. Essentially, such a word translation problem comes down to word-sense disambiguation, a task that is already difficult if performed in isolation. However, for translation it does not stop there. Often, it is not possible to translate a sentence from one language to another by mapping words to translated words. One word may require several words to translate it. Several words may be translated by only one word. Also, the grammatical structure of the two languages is probably different. As a result, translating the words in the original sentence and simply concatenating their translations to form the translated sentence will probably lead to a sentence that seems awkward at best, but might also be unintelligible. To resolve this, a translator also needs knowledge about the grammatical structure of the target language.

Thus, in order to correctly translate a word, one needs to combine information relating to the word itself, its context in the original sentence, and its translation’s context in the translated sentence. This complex interaction between dependencies on several levels—resulting in the situation where everything seems to depend on everything else—makes that automatic translation, or machine translation, is best dealt with as a structured prediction problem. As such, it is yet another problem to which we can apply our constraint satisfaction approach to structured prediction. Looking at machine translation as just another structured prediction task, it is—in general—no different from sequence labelling or dependency parsing. At the same time though, it is obvious that true state-of-the-art machine translation performance can only be attained through many task-specific optimisations. Those optimisations are outside the scope of this chapter, in which we present a proof-of-concept approach to machine translation, based on our con-
6.1 Data-driven machine translation

When machine translation first emerged as a sub-field of computational linguistics, in the 1950s, rule-based methods were the method of choice. As in most other fields, data-driven alternatives to rule-based approaches are taking over this position. Compared with rule-based systems, the main advantage of data-driven methods is the relative ease with which they can be adapted to new language pairs or new domains. With rule-based methods, such adaptation requires extensive rewriting of rule sets, or possibly even starting from scratch. For data-driven translation approaches, most of what is needed to adapt a system is to provide it with new training data.

For most work on data-driven machine translation, those training data have the form of parallel corpora, i.e. collections of documents that are available in several different languages. In some cases, such texts are readily available, and require no additional translation effort. This is the case, for example, for parliamentary proceedings that, for legal reasons, have to be available in several different languages, such as the proceedings of the European (Koehn, 2005) or Canadian (Germann, 2001) Parliaments. Other useful sources of readily available parallel texts are movie subtitles (Tiedemann, 2007), or software manuals (Tiedemann and Nygaard, 2004). Whatever their source, parallel texts can be seen as examples of how to translate from one language to another, in the same way that structural annotations are examples of how to perform more abstract linguistic processing tasks. However, whereas those annotations are often explicitly created for the sole purpose of applying automatic learning techniques on them, the documents in parallel corpora actually aim only at communicating the same message in all languages. For this reason, given two translations of the same document, it is not always evident how the different parts of the documents relate to one another. Nevertheless, in order to successfully learn to translate from those texts, this relation has to be made explicit. Doing so is referred to as alignment, which involves linking the segments of two texts that are translations of one another. Such alignment can be performed on several levels, for example, paragraphs, sentences, or words.

Without going into much detail here, we simply mention that methods exist
6.2. Existing work

Learning from word-aligned sentence pairs is the most common setup for data-driven machine translation. Given this common starting point, though, a wide variety of methods have been developed to actually learn how to perform the translation task. In this section, we give a survey of three such methods: statistical machine translation, example-based machine translation, and discriminative machine translation.

6.2.1 Statistical machine translation

Statistical machine translation (SMT) is by far the most popular approach to machine translation in recent years. Essentially, the technique assumes that if the conditional probability $P(e|f)$ of an English sentence $e$ given the foreign input $f$ can be computed, then theoretically at least, one can look for the English sentence for which this probability is highest, and select that sentence as the translation. Moving away from theory towards practical implementation issues, there are two main difficulties to be overcome in order to make statistical machine translation work. One of those is how to actually find the English sentence that has the

---

1Throughout this entire chapter, we will use the term source to refer to the language translated from, and target to refer to the language translated to. This is different from the terminology often used in the (statistical) machine translation literature, in which the use of the words source and target is exactly opposite to ours.
highest posterior probability for any given foreign sentence. The other is how to estimate the probability distribution.

In the statistical machine translation literature, the process of finding the translation with highest probability is often referred to as decoding; however, essentially it is the same as what is generally referred to as inference in general structured prediction. In contrast with inference for most structured prediction problems, efficient optimal decoding for machine translation is simply too expensive. Since there are infinitely many possible English sentences, enumerating them all is obviously not a realistic option. As an alternative to optimal decoding, heuristic search algorithms are employed that attempt to distinguish promising translation directions and only focus on those. The most common choice of algorithm is a stack search—or A* search—combined with a beam-search strategy that prunes low-probability hypotheses (Wang and Waibel, 1997; Koehn, 2004).

The second issue that has to be solved is how to model the conditional probability distribution $P(e|f)$. Due to sparsity issues, directly modelling this distribution is infeasible. To circumvent such sparsity issues, a factorisation of the probability distribution is required that expresses the probability in terms of probability distributions that can be estimated from data more reliably. In case of statistical machine translation, this factorisation is based on the noisy channel model (Shannon, 1948; Weaver, 1955), which uses Bayes rule to rewrite the original probability as the product of two other probabilities.

$$P(e|f) = P(f|e)P(e)$$

The first of these, $P(f|e)$ is the translation model and is estimated from an aligned parallel corpus (Brown et al., 1993). Since it is concerned with assessing the quality of word translation, the translation model is responsible for the faithfulness of the translation. The fluency of the translation is expressed by the language model $P(e)$. Estimating this distribution requires a monolingual English corpus only, and is typically done on the basis of $n$-gram models, although other model types have been used as well, such as those based on syntactic structure (Sawaf et al., 2000; Charniak et al., 2003).

The above equation describes the basis of statistical machine translation. Most modern statistical machine translation systems integrate many more pieces of information to improve the quality of the translation. For example, a distortion probability is used to prevent nearby words in the foreign sentence ending up far apart in the English translation. As another extension, translation models estimating the translation quality of entire phrases can improve both the faithfulness and the fluency of the translation. As a final example, word penalties may prevent the translated sentences from becoming too long or too short.

While all these extra distributions can potentially contribute to improved translation performance, the fact that conditional dependencies may exist between them makes it non-trivial to integrate them. Log-linear models are the most common formalism for doing so (Berger et al., 1996; Papieni et al., 1998).
6.2. Existing work

In such models, a probability function is rewritten as a weighed sum of logarithms of feature functions, where feature functions often correspond to probability distributions, though that is not required.

\[ P(e|f) \propto \sum_{m=1}^{M} \lambda_m f_m(e, f) \]

Numerical optimisation algorithms have been proposed to find good parameters \( \lambda_m \) automatically (Och and Ney, 2002). It has been found that directly optimising for some relevant evaluation metric, such as BLEU (Papineni et al., 2002), results in better performance than simply optimising for likelihood.

6.2.2 Example-based machine translation

Unlike statistical machine translation, which is defined by the noisy-channel equations, example-based machine translation (EBMT) is better regarded as a loose family of approaches. The common ground shared by all of those could be formulated as the notion that knowledge is better extracted implicitly from a large database of stored example translations on a case-by-case basis, than extracted beforehand and stored explicitly. The underlying assumption is that similar sentences have similar translations, which suggests that unseen sentences can be translated by analogy, looking at the translations of example sentences that are similar to the sentence to be translated. Originally proposed by Nagao (1984), this intuition has formed the basis for a rather diverse family of approaches that are grouped under the EBMT moniker.

Somers (1999) describes example-based approaches as consisting of three steps: matching, alignment, and recombination. Matching is concerned with finding source-language examples similar to the sentence to be translated. An important issue here is, when exactly two sentences are to be considered similar. Without question, an exact match is optimally similar, but finding exact matches must be considered a fortunate coincidence, rather than an assumption to base a translation system on. As an alternative to searching for exact matches, example-based approaches look for partial matches by decomposing sentences and collecting example sentences containing matching parts. As an example, Veale and Way (1997) perform such decomposition according to certain marker words, which signal pseudo-syntactic segment boundaries. Alternatively, Sato and Nagao (1990) match on syntactic dependency relations between words.

When similar sentences have been retrieved, the remaining two steps, alignment and recombination, are responsible for generating the actual translation given one or a few example sentence pairs. If found by partial matching, an example sentence contains some constituents that are also present in the source-language sentence to translate. Translating by analogy involves using the translation of those matching parts, and substituting the translation of dissimilar parts.
Doing so requires that the system knows what parts of the target-language sentence are translations of parts of the source-language sentence. Determining this is done in the alignment step. Finally, the recombination step merges the partial translations found by the matching and alignment steps into a good target-language sentence. Intuitively, recombination may be no more than template filling. However, combining partial translations retrieved from different places may involve having to repair incompatibilities, or *boundary friction*, between the different parts. This may, for example, be the case for grammatical agreement. Somers (1999) notes that this part is among the most difficult, and least investigated parts of example-based machine translation. Among the solutions explored to resolve boundary friction are target-side language models (Groves and Way, 2005), and recombination strategies based on overlapping translation fragments (Brown et al., 2003).

### 6.2.3 Discriminative machine translation

Although SMT is by far the most popular approach to machine translation, it is not without drawbacks. These drawbacks are mainly due to limitations of the noisy-channel model, which is at the basis of most SMT approaches. For computing the likelihood of a candidate translation, the conditional probability function is factorised as a product of several probability models. This factorisation, however, is only valid if all factors are conditionally independent. As one of the implications of this requirement, arbitrary overlapping features, which have been shown to be most useful in many other areas of natural language processing, are not allowed. Because of this, other areas of natural language processing have adopted discriminative alternatives to the generative noisy-channel model. Most notably this is the case for information extraction, where e.g. structured linear models have successfully replaced the noisy-channel-based hidden markov models for most purposes. In machine translation, discriminative training approaches for log-linear SMT models (Och and Ney, 2002) have been used to circumvent this restriction; however, they are only feasible for a small number of factors.

Recently, a number of fully discriminative structured prediction approaches for machine translation have been proposed. Unlike previously-mentioned approaches, these do not tune weights based on the output of a traditional SMT decoder, but actually integrate decoding and weight tuning, which allows for much larger numbers of features. As an example, Liang et al. (2006) reformulate phrase-based translation as a structured linear model. As explained in Chapter 2, such models find an output structure by maximising the product of a learned weight vector and a joint feature map $\psi(x, y)$ over both input and output. To allow for reordering of phrase translations in the target sentence, a hidden alignment structure $h$ is added to the model. Then, the feature map is defined over the input, output, and alignment structures, i.e. $\psi(x, y, h)$, and thus, the target function
optimised by the decoder to find the most-likely translation is the following.

\[ \hat{y} = \arg \max_{y, h} \langle w, \psi(x, y, h) \rangle \]

If only features for translation and language model probabilities are included in this vector, the model is rather similar to traditional phrase-based SMT. Significant performance improvements can be attained by adding additional features. For example, specific lexical features may override the translation or language model in case of additional contextual evidence. Also, part-of-speech features allow for better generalisation by adding the ability to back off to less specific representations of both the source and target sentence.

### 6.3 Constraint satisfaction approach

The constraint satisfaction approach presented in this chapter is formulated as an extension of statistical log-linear models. Recall from Section 6.2.1 that a typical log-linear model for machine translation combines a number of feature functions, each of which measures the quality of a candidate translation according to some aspect. Two feature functions tend to be part of any SMT system; a translation model, and a target language model, measuring the faithfulness and the fluency of the translation, respectively. Both of those are actually probability distributions, obtained by maximum-likelihood estimation from training data. The best translation is determined by maximising a weighted sum of those feature functions.

\[ \arg \max_y \lambda_{\text{TM}} \log P(x|y) + \lambda_{\text{LM}} \log P(y) \]

One of the problems with this traditional formulation is that the translation model ignores the context in which it is applied. This is the case for the source sentence context, as well as for the target sentence context. It is expected that the language model compensates for this. However, it is questionable whether this is a reasonable expectation. Since the language model does not take into account the source side at all, it can only resolve source-side ambiguities indirectly by looking at the translations of source words. Essentially, this means that the language model is not only used for attaining good fluency, but also in part for attaining good faithfulness, the latter of which, it might not be good enough for. Our extension will use constraint satisfaction to improve the translation model by having it take into account both source and target sentence contexts.

To do this, we replace the language model by a constraint model. The score assigned to a candidate translation by this model corresponds to the sum of weights of satisfied constraints according to a constraint satisfaction problem.
Chapter 6. Machine translation

The score formula is adapted as follows.

\[
\arg \max_y \quad \lambda_{CM} \quad f_{CM}(y) \\
+ \quad \lambda_{LM} \quad \log P(y) \\
+ \quad \lambda_{NM} \quad \sum_i [y_i = \text{NULL}] \log P(y_i = \text{NULL}|x_i) \\
+ \quad \lambda_{LP} \quad |y|
\]

(1) (2) (3) (4)

The constraint model feature function (1) scores the satisfied soft constraints, and thus corresponds to the same objective function as was optimised in previous chapters. Considering the difficulty of the translation task, we augment the objective function with a few more feature functions.\(^2\) The language model (2) is a standard back-off trigram language model, estimated using the SRILM toolkit (Stolcke, 2002). Two more feature functions are intended to compensate for the effect that n-gram language models tend to prefer shorter translations. The first, which we call the null model (3), multiplies the translation probability of those source-language words that are translated to NULL, i.e. words for which no corresponding word is generated in the target-language sentence. It is estimated using relative frequencies, and is in fact similar to the translation model of SMT systems, with the exception that it only applies to source-language words left untranslated. The final feature function, the length penalty (4), is fairly common in log-linear SMT systems. It simply counts the number of target-language words. Given a positive weight \(\lambda_{LP}\), it is in fact a length bonus rather than a penalty.

For the implementation of the constraint model, we proceed along the same lines as in the previous chapters by defining a weighted constraint satisfaction problem over a solution space of possible translations. Therefore, two questions need to be answered. First, how do we restrict the solution space? As in previous chapters we aim at excluding most candidate solutions before the inference even starts. Defining this solution space is done by introducing variables and populating their domains, and by formulating certain hard constraints that every valid translation has to satisfy. Secondly, what soft constraints are added to the constraint satisfaction problem? As mentioned previously, we want those constraints to improve the faithfulness of the translation by taking into account both source sentence context and target sentence context.

6.3.1 Solution space

Strictly speaking, the definition of machine translation provides us with only one guarantee with respect to the output, namely that it is a sentence in the target language. There is no way even to predict how many target-language words are

\(^2\)The log-linear formulation of the objective function can be shown to be equivalent to a weighted constraint satisfaction problem, but we choose to follow this formulation, since it eases comparison with SMT systems.
needed to translate the input sentence. One word in the translated sentence may actually cover several words in the original sentence. The reverse is possible as well, that is, one word in the source sentence that requires several words in the target sentence. Because of language differences like these, an approach that is guaranteed to be optimal has to search an infinite output space, which obviously is not a viable option. Efficient approaches will have to make strong assumptions about the parts of the true output space that are actually worth exploring. Below we present a model that is sufficiently restricted to allow for efficient inference, while remaining expressive enough to attain good translation performance.

To represent a solution space, we start by distinguishing two sub-problems that have to be solved as part of the translation task. First, source-language words have to be translated to the correct target-language word. Secondly, the translated words may need to be reordered—possibly new words have to be inserted as well—to make the translation a natural sentence according to the target language. By distinguishing these two sub-problems, we do not suggest that they be solved separately. On the contrary, treating machine translation as a structured prediction task means that all sub-problems are solved simultaneously. However, for illustrative purposes, we treat them separately.

Representing word translations

For modelling the translation of words in the source sentence, we adopt the assumption that each word in the source sentence is translated to exactly one word in the target sentence. This is in fact the same assumption as made by the IBM alignment models (Brown et al., 1993). In our constraint satisfaction framework it is naturally represented by introducing one variable for each source word. During inference, the target-language words that are part of the domain of a variable will be considered as possible translations of the corresponding source word. If domains are constructed simply by listing all translations for the given source word that are found in the training corpus, the solution space of our model would be rather similar to that of word-based SMT systems. However, in constraint satisfaction inference, the domain values are obtained from base classifier predictions. As a result, the solution space is likely to be considerably smaller.

A few additional issues need to be dealt with. First of all, spurious words in the source sentence should not be translated to a target-language word. This is resolved by translating the word to a special NULL symbol instead. By definition, this NULL symbol will be part of the domain of all variables. As a result, any source-language word may be left untranslated in the target translation; there is no need for a base classifier to suggest this. A second issue is the fact that several source-language words might be translated by only one target word. In this case, all corresponding variables are assigned that same word. The fact that those matching words are actually a single token in the target sentence is dealt with in the target sentence realisation.
Chapter 6. Machine translation

Representing target sentence realisation

Ignoring the issue of choosing the correct target-language translation for an input word, target sentence realisation involves three differences between the source language and the target language that have to be represented.

- **Word order differences.** Depending on how close the two languages are, this may or may not be a frequent issue, but for any pair of languages it is bound to occur at least sometimes. To illustrate this, in Figure 6.1, the Dutch words “betekende Maastricht” are translated in reversed order as “Maastricht was”.

- **Zero-fertility words.** Some words in the target sentence may not align with any word in the source sentence. When generating the translated sentence, such words somehow need to be inserted based on target language properties. For example, this is the case for the English word “all” in Figure 6.1, which is not aligned with any Dutch word, but still required to translate “vooral” correctly as “above all”.

- **Multi-fertility words.** As already hinted at earlier, sometimes several source words are translated by a single target word. This would be the case if the direction of translation in Figure 6.1 was reversed. Then, the two English words “above all” would be translated by the single Dutch word “vooral”.

**Word order differences**  To cope with arbitrary word reorderings in the translation, the inference procedure essentially has to consider every permutation of translated source words. For a compact representation of the search space yielded as a result of this, consider a complete directed graph, in which the words of the source sentence are represented by vertices, and one additional vertex $v_0$ corresponds to the start and end of the sentence. A directed arc from vertex $v_i$ to vertex $v_j$ means the translation of the word corresponding to $v_j$ directly follows the translation of the word represented by $v_i$. In addition, a directed arc from $v_0$ to $v_i$, or from $v_i$ to $v_0$ means that the translation of the word corresponding to $v_i$ is the first or last word of the sentence respectively. Note that this representation does not yet account for groups of source words that are translated to a single target word; the representation will be extended for that below. Now, the space of all candidate translations corresponds to all cycles that start and end at $v_0$. Such a cycle is not required to visit every vertex in the graph, i.e. it does not have to be a hamiltonian cycle. Cycles that do not visit a certain vertex $v_i$ correspond to translation candidates in which the source word represented by $v_i$ is not translated. In that case, the translation variable corresponding to this source word should have the value NULL, which can easily be enforced by a hard constraint on the output space.

Given this graph representation of the candidate translation space, it can be reformulated for the constraint satisfaction framework by introducing a set of
6.3. Constraint satisfaction approach

Figure 6.2 Visualisation of the connectivity matrix of the path corresponding to the correct translation of the Dutch sentence in Figure 6.1. The “•” in the row labelled with “Maar” and the column labelled with “vooral” denotes that the translation of the latter follows that of the former in the English target sentence. The word “all” in the row of “vooral” and the column of “Maastricht” is an English zero-fertility word separating the translations of the two Dutch words in the English target sentence.

\[(n + 1) \times (n + 1)\] variables, where \(n\) is the length of the source-language sentence, that correspond to the adjacency matrix of the graph just introduced. The domains of all those variables comprise two values, signalling whether or not the corresponding arc is included in the candidate translation cycle. Appropriate constraints have to be added to the constraint satisfaction problem to ensure that every candidate considered is indeed a cycle of the graph. Informally, this is the case if for every \(i \in \{1, 2, \ldots, n\}\), either the \(i\)th row and column do not contain any positive value at all, or they both contain exactly one positive value. Moreover, the 0th row and column should contain exactly one positive value. To illustrate all of the above, Figure 6.2 shows a filled matrix representing the correct translation order for the example sentence in Figure 6.1.

Zero-fertility words A common approach to zero-fertility word insertion is to keep a list of frequent zero-fertility words and attempting to insert words from this list at arbitrary positions in the translation. It is easy to see that this leads to a substantial expansion of the output space, which was already large to start with. As an alternative to common practice, we choose to attempt insertion of zero-fertility words only if there is evidence that doing so would make sense in the context of the current sentence. This evidence is to be provided by base classifier predictions. More specifically, those predictions will be used to collect
zero-fertility words that are candidates for insertion between the translations of two source words. Further details on this collection process are postponed until later in this chapter. For now, it is important to note that we restrict our representation, so that at most one zero-fertility word can separate two fertile words.

Modelling this is possible by a straightforward extension of the adjacency matrix introduced before. In addition to the two values that signify whether or not two translated words are adjacent in the target sentence, a matrix element can also be assigned a word. Such an assignment encodes the case where two translated words are separated by the zero-fertility word stored in the matrix. In Figure 6.2, the matrix element that connects the translations of “vooral” and “Maastricht” has the value “all”, which denotes that the translation of the latter follows that of the former, separated by the word “all”. The constraints that ensure that only cycles of the order graph are considered as candidate solutions can also be extended easily to this new setting. The zero-fertility word values can simply be treated as positive values.

**Multi-fertility words** The English word “all” in Figure 6.1 has to be inserted in the English translation to translate the Dutch word “vooral” correctly as “above all”. This example illustrates that translation is not simply a matter of uniquely mapping source words to target words. In the translation direction discussed so far, i.e. Dutch to English, the single word “vooral” should actually be translated by two English words. The other way around, when translating from English to Dutch, the two English words “above all” should be translated to a single Dutch word. To account for such many-to-one mappings we introduce one final value that can be assigned to order matrix entries. This value signals overlap between two source words, i.e. they are both mapped to the same target word.

### 6.3.2 Constraints

The constraints we devise for machine translation are similar to those used for sequence labelling in Chapter 4. In both types of task, sequential output structures are generated, and therefore it may be expected that constraints that worked well for sequence labelling will be useful for machine translation as well. In our approach to sequence labelling, constraints cover up to three consecutive output labels. As it happens, for sequence labelling, those labels correspond to three consecutive input tokens; however, the inference procedure does not assume that to be the case. Once predicted, constraints that are part of the inference procedure apply to the structure of the output only, they do not make any additional assumptions about the relation between input and output. For that reason, three consecutive elements of the output sequence may actually correspond to arbitrary elements of the input sequence. The flexibility resulting from this loose relation between the structure of the input and that of the output is especially useful for
6.3. Constraint satisfaction approach

Figure 6.3 The Dutch example sentence from Figure 6.1 and the English trigrams that are to be predicted for the words in the sentence. No training example is created for the word “de”, because it is aligned with the NULL token.

<table>
<thead>
<tr>
<th>Maar</th>
<th>But</th>
<th>above</th>
</tr>
</thead>
<tbody>
<tr>
<td>vooral</td>
<td>But</td>
<td>above</td>
</tr>
<tr>
<td>betekende</td>
<td>Maastricht</td>
<td>was</td>
</tr>
<tr>
<td>Maastricht</td>
<td>all</td>
<td>Maastricht</td>
</tr>
<tr>
<td>de</td>
<td>was</td>
<td>a</td>
</tr>
<tr>
<td>doorbraak</td>
<td>a</td>
<td>breakthrough</td>
</tr>
<tr>
<td>van</td>
<td>breakthrough</td>
<td>for</td>
</tr>
<tr>
<td>de</td>
<td>for</td>
<td>EMU</td>
</tr>
<tr>
<td>EMU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

machine translation, where it is not uncommon that the order of target-language words is different from that of the source words they translate.

Similar to the n-gram constraints for sequence labelling, we choose the constraints for machine translation to cover up to three consecutive target-language words. Also following the approach for sequence labelling, such constraints are created by predicting a trigram of target-language words for each word in the source sentence. Figure 6.3 illustrates this process for the sentence pair in Figure 6.1. The middle word of the predicted trigrams is the hypothesised translation of the source word in focus. The left and right parts are the words surrounding the translation in the target sentence, and, once again, are not necessarily the translations of the words surrounding the focus word in the input sentence. In fact, they may even be zero-fertility words. Note that no training example is created for the Dutch word “de” in Figure 6.1. Nevertheless, when translating a sentence, trigrams are predicted for all words in the source-language sentence—whether a word is aligned with NULL is unknown for new sentences.

Given a predicted trigram, two types of constraints are extracted from it. First, a trigram constraint covering the translated word and the two words surrounding it in the target-language sentence. Secondly, two bigram constraints defined on the translated word and either one of the two surrounding words.

6.3.3 Solving the CSP

The solution space of the constraint satisfaction problem defined in this section has immense proportions. To illustrate this, even if a base classifier perfectly predicts the correct translations of all source words, which is already overly optimistic, the inference procedure still has to consider every possible permutation of those
translated words as a candidate translation. An important implication of this is that solving the constraint satisfaction problem by exhaustive search, as was done for sequence labelling, is infeasible for all but the smallest of sentences. Unfortunately, no further restrictions or assumptions can be made that would restrict the solution space sufficiently to allow for exhaustive solving. Rather, enormous output spaces are inherent to machine translation and therefore cannot be avoided. As a consequence, approximate solving is the only option.

With this in mind, we choose the greedy decoding algorithm of Germann (2003) as the basis for the constraint solver. The algorithm starts with a complete candidate translation; for example, one where all source words are mapped to their most likely translations and added to the target sentence in original order. Subsequently, a hill-climbing search is started in which simple transformations of the current translation are attempted and the one leading to the highest score increase is actually applied. New transformations are tried until no further improvement can be attained. The following transformations are considered.

- **Change** the translation of a source-language word. If the target word currently aligned with it has a fertility greater than one, a new target word is inserted in the translation at the position maximising the translation score; otherwise, the current translation is changed, while its position is left unchanged. Among the translation candidates tried is also **NULL**, which results in the word being removed from the candidate translation.

- **Insert** a zero-fertility word. According to our model, zero-fertility words are only inserted in between two fertile words.

- **Erase** a zero-fertility word.

- **Join** two target-language words. Essentially, this operation comes down to removing one of the words from the translation and aligning with the remaining word all words previously aligned with the word that was removed.

- **Swap** two non-overlapping segments of the target sentence. For reasons of efficiency, both the maximum length of a segment and the maximum distance between the two segments is restricted.

Although the algorithm has been proposed in the context of statistical machine translation, it can more generally be seen as optimising an arbitrary objective function defined over candidate translations. By replacing the noisy-channel equations optimised originally by a credit function based on constraint weights, the algorithm can be employed for solving our constraint satisfaction problem.
6.4 Experimental setup

6.4.1 Data

For training the components of the translation system, as well as for evaluating its performance, a parallel corpus is required. For this purpose, we use four different corpora: EuroParl, JRC-Acquis, EMEA, and OpenSubtitles. They cover a diverse range of different genres, from legal texts to both formal, and informal speech. From each of the four corpora, we prepare data sets for the translation direction Dutch to English.

EuroParl The EuroParl corpus (Koehn, 2005) is a multi-lingual parallel corpus extracted from the proceedings of the European Parliament. Following European law, these are available in all official languages of the European Union; 11 of those are collected in the corpus. The complete Dutch-English parallel subcorpus consists of 1,313,111 bilingual sentence pairs.

JRC-Acquis The JRC-Acquis corpus (Steinberger et al., 2006) comprises a large collection of legislative texts extracted from the Acquis Communautaire, the total body of European Union law. It is available in 22 languages. The Dutch-English parallel subcorpus provides 1,235,878 bilingual sentence pairs.

EMEA The EMEA data set is composed of medical texts made available by the European Medicines Agency, and like the above two corpora, translated in all official European languages. It is one of the corpora included in the OPUS parallel corpus (Tiedemann and Nygaard, 2004). The parallel texts for Dutch and English cover 751,602 sentence pairs.

OpenSubtitles The OpenSubtitles data set is another part of the OPUS parallel corpus. The OpenSubtitles part of this corpus provides aligned movie subtitles in various different languages. For the language pair Dutch-English, it comprises 288,160 sentence pairs.

In the experiments, the above parallel corpora have been used as four different benchmarks. No texts have been mixed, but rather in four experiments, the translation system has been trained and tested on texts of a single corpus. For this evaluation, as well as for tuning the system, from each of the four corpora, two sets of 1,000 sentences each have been selected for testing and development purposes respectively; the remainder is used for training. This training data has subsequently been aligned at the word level using GIZA++ (Och and Ney, 2000).

6.4.2 Evaluation

Various evaluation methods for machine translation have been proposed. There is no question that having human judges decide on the quality of the translations is
the most reliable method; however, automatic evaluation may be preferable, since it is quick and cheap. Fortunately, formal metrics are available that have been shown to correlate fairly well with human judgement. Of those metrics, BLEU (Papineni et al., 2002) has seen the broadest adoption. The metric computes a score that is based on the amount of overlapping n-grams between the proposed translations and one or more reference translations for each sentence. In our experiments, translations are evaluated with BLEU against a single reference translation for each sentence. Overlap between the produced translation and the reference translation is computed on the basis of n-grams of size 1 to 4, and case information is ignored.

6.4.3 Constraint prediction

For predicting the soft constraints of our translation model, we need to map each word of the source sentence to a trigram of target words. The middle word of that trigram is the translation of the source word in focus; the left and right words are the two target words surrounding it. We stress once more that those are not assumed to be the translations of the two source words surrounding the source word in focus. Essentially, the classification task at hand is rather similar to word-sense disambiguation, since the most difficult step in finding the translation of a word is deciding on the correct sense of the word. Classification approaches to word-sense disambiguation often use broad sets of features that describe the context of the word in focus (Lee and Ng, 2002; Martinez et al., 2002; Agirre and Stevenson, 2006). This stands in sharp contrast with common practice in statistical machine translation. There, translation models tend to ignore context altogether. Instead, translation models simply return maximum-likelihood distributions of the possible translations, and have the target language model decide on the correct translation.

Several recent studies (Carpuat and Wu, 2007; Chan et al., 2007; Giménez and Márquez, 2007; Stroppa et al., 2007) have experimented with classification-based alternatives to traditional translation models that take into account contextual information of the word in the source sentence, similarly to the way word-sense disambiguation is performed. Our constraint predictor is similar to the classifiers used in these studies in the sense that contextual information is used to improve the suggested translations. The feature set used in our classifier is much simpler, though. In specific, the features used correspond to a word window of length three centred on the focus word. As a consequence of the small number of features and the large number of classes, it will often be the case that the classifier finds several classes that have the same score for an input sentence. In our memory-based learning implementation, such ties are normally resolved by choosing the class that has highest prior probability among all classes that have the same maximum score. However, classes that are assigned the same score by a base classifier are the perfect example of uncertainty that cannot be resolved locally, and thus should be delegated to the inference procedure. Therefore, for the experiments in this
chapter, we disable tie-breaking in the base classifier, and extract domain values and constraints from all classes that have the maximum score.

The target-word trigrams predicted by the base classifier are used to add constraints to the inference, as well as to compose the domains of the variables. Constraints are derived from the predicted trigrams in the same way as is done for sequence labelling, that is, the predicted trigram itself is turned into a constraint, but also the two bigrams covered by the predicted trigram.

Whereas the prediction of constraints proceeds in the same way as for sequence labelling, the composition of the domains of the variables is necessarily different. When predicting trigrams for sequence labelling, it is known what variables the left and right parts of that trigram may be assigned to; the variables corresponding to the left and right neighbours of the input token in focus, respectively. However, this is only the case because the output labels retain the same order as the input tokens they correspond to. For machine translation, this is not necessarily true. For every predicted trigram, only the destination variable for the middle part of it can be identified, since that is the hypothesised translation of the word in focus. Therefore, the domains of the translation variables will be composed of the middle parts of the trigram predictions for the corresponding source word. This procedure is illustrated in Figure 6.4, where in the absence of tie-breaking in the classifier, two trigrams have been predicted for each source-language word. Since for all words, both trigrams suggest a unique translation, the domains of the three words, \( D_1 \), \( D_2 \), and \( D_3 \), each contain two candidate translations, as well as the symbol NULL, which is always included as a possible translation.

The variable domains for the order variables always contain at least the two values that signal that the two corresponding words do or do not follow one another in the translated sentence. In Figure 6.4, these variables correspond to the edges of the graph depicted at the bottom of the figure. The two symbols \textsc{FOLLOWS} and \textsc{NONE} are included in all domains. Furthermore, the model also allows for an overlap value or a zero-fertility word as value for order variables. As for the former, the overlap value is only added to the domain of the order variable \( y_{ij} \) if words \( i \) and \( j \) can be translated to the same target word, or more formally, if their domains overlap, \( D_i \cap D_j \neq \emptyset \). As an illustration, \( D_2 \) and \( D_3 \) both contain the word “impossible”, and therefore, \( D_{23} \) and \( D_{32} \) contain the symbol \textsc{OVERLAPS}.

Potential zero-fertility words are also added to a domain only if base classifier predictions provide sufficient evidence for that. Specifically, the zero-fertility words included in the domain of the order variable \( y_{ij} \) are those words that appear both as the right part of the trigram predicted for word \( i \), and as the left part of the trigram for word \( j \). In the example, the words “That is” predicted for “Dat” overlap with the words “is impossible”, predicted for both “kan” and “niet”. For this reason, “is” is made a potential zero-fertility word if the translation of either “kan” or “niet” were to follow that of “Dat”. Similarly, “can” is a potential zero-fertility word between the translations of “Dat” and “niet”, since “It can” has been predicted for the former, and “can’t” for the latter source-language word.
Figure 6.4 Visualisation of the search space resulting from a set of base classifier predictions. Top: the correct alignment of a Dutch-English example sentence pair. Middle: trigram predictions for the words in the Dutch sentence. Bottom: the complete graph connecting all Dutch words. The nodes and edges of this graph correspond to micro-labels and are labelled with a value out of the corresponding domains. Any valid translation is a directed cycle in this graph that starts and ends in the BEGIN/END node.
6.5 Results

To evaluate our constraint satisfaction approach to machine translation, we trained and tested the system using the four Dutch to English data sets described in Section 6.4.1. In addition, we also trained and tested an existing statistical machine translation system on the same data sets. More specifically, we implemented a word-based SMT system based on the ISI ReWrite decoder, which uses the greedy decoding algorithm of Germann (2003). Comparing with this system is especially interesting since the decoding algorithm is the same as the one used in our constraint satisfaction system. Therefore, the differences that are observed are most likely to be attributed to the modelling choices underlying the two systems. First, the constraint satisfaction system uses a richer objective function based on the constraint model that replaces the translation model. Secondly, constraint satisfaction inference searches a smaller solution space than the ReWrite system, which does not restrict its solution space in advance.

Unquestionably, the primary evaluation measure for machine translation should be the quality of its translations. However important translation quality may be though, given the enormous output space of the translation task, translation speed should be a matter of concern as well. Since reducing the cost of inference is one of our objectives in this study, we will specifically take into account this factor. We start our discussion of the experimental results with an overview of the translation scores. After that, we focus on the speed with which translations are generated.

Table 6.1 shows the BLEU scores for both systems on each of the four data sets. Remarkably, the conclusions to be drawn are different for each of the four data sets we tested on. The two systems are closest in performance on the EuroParl data, though with a BLEU score of 21.1, constraint satisfaction inference outperforms ReWrite, which attains 19.8. More substantial differences are observed with the remaining three data sets. On EMEA, ReWrite outperforms constraint satisfaction inference, with a score of 39.5 for the former, and 28.0 for the latter. On JRC-Acquis and OpenSubtitles, constraint satisfaction inference outperforms ReWrite by quite large margins. Constraint satisfaction inference scores 51.3 and 19.3, versus 45.0 and 8.5 respectively, for ReWrite.

The relative performance differences are rather diverse among the four data sets. This may be attributed to the underlying search algorithm, a greedy hill-climbing search, which is known to risk ending up in suboptimal local optima. In particular, this explanation more than likely accounts for the exceptionally low score of ReWrite on the OpenSubtitles data, but this shortcoming of hill-climbing search may be expected to have an effect on the performance of both systems on other data sets as well. Having said this, constraint satisfaction inference seems to deal with this circumstance better than ReWrite. On the one hand, the richer objective function used by constraint satisfaction inference, based on the predicted constraint model, may account for the better performance of constraint satisfaction inference. On the other hand, though, the smaller solution space searched by constraint satisfaction inference may also be expected to have fewer
Chapter 6. Machine translation

<table>
<thead>
<tr>
<th></th>
<th>EuroParl</th>
<th>JRC-Acquis</th>
<th>EMEA</th>
<th>OpenSubtitles</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSI</td>
<td>21.06</td>
<td>51.33</td>
<td>28.04</td>
<td>19.26</td>
</tr>
<tr>
<td>ReWrite</td>
<td>19.81</td>
<td>44.97</td>
<td>39.51</td>
<td>8.51</td>
</tr>
</tbody>
</table>

Table 6.1  BLEU scores for constraint satisfaction inference and the ReWrite SMT system on the four Dutch to English translation tasks.

local optima.

The smaller solution space searched by constraint satisfaction inference plays a crucial role in attaining good translation speed as well, the second aspect that we evaluated in the experiments. Table 6.2 shows the time both systems spend on translating the 1,000 test sentences of each of the four data sets. On EuroParl and JRC-Acquis, the decoding times for constraint satisfaction inference are considerably shorter than for ReWrite. This is most apparent with the JRC-Acquis data, for which ReWrite needs almost 26 hours, whereas constraint satisfaction inference is finished within 3 hours. With average sentence lengths around 25 tokens for both data sets, it is only to be expected that decoding is a costly operation. Yet, constraint satisfaction inference manages to restrict its decoding time on these two data sets much better than ReWrite.

In the EMEA and OpenSubtitles test sets, the average sentence lengths are only 9 tokens and 7 tokens respectively. This explains why both systems only need minutes rather than hours for decoding these. ReWrite is the faster of the two. It spends only 3 minutes on translating the OpenSubtitles test set, which suggests that it tends to reach local optima of its objective function rather quickly. Considering the low BLEU score attained by ReWrite on the OpenSubtitles task, though, these local optima are suboptimal. On the EMEA data set, ReWrite is also faster than constraint satisfaction inference, yet here, it attains a better BLEU score. Apparently, the hill-climbing algorithm only needs a few transformation operations to reach good translations. Constraint satisfaction inference takes longer, but attains a lower score. Its objective function causes it to perform more transformation operations than it should.

6.6 Conclusion

One of the research objectives in this dissertation is to propose a method that manages to restrict the costs of inference in structured prediction, while not excessively limiting the expressiveness of the model. With respect to this objective,

3It should be pointed out, though, that our decoder was implemented in Python, whereas the ReWrite decoder is programmed in C++. The latter is known to be potentially dozens or even hundreds of times faster, which suggests that decoding on these two data sets is also performed the fastest by constraint satisfaction inference.
one can hardly think of a better touchstone than machine translation, an area that deals with huge output spaces that are costly to search, and at the same time depends strongly on the quality of the inference, since global structural dependencies play an important role in machine translation. Because of the latter, it is to be preferred that the model is restricted as little as possible, so that global dependencies of arbitrary types may be modelled. Because of the immensity of the output space, having to take measures to restrict the cost of search is unavoidable.

The decoding algorithm chosen for the experiments in this chapter is an important ingredient for achieving the above objective. Since the algorithm is a local hill-climbing method, at any moment where the objective function evaluates a hypothesis, there is a complete, rather than a partial translation as would be the case in A* or Viterbi search. As a result, the objective function can take into account arbitrary structural dependencies. The possibilities for such dependencies are virtually unlimited. In this chapter, we experimented with only a single type of constraint, which models trigrams of target-language words. We expect that large improvements can be achieved by introducing additional constraints. For example, constraints that model phrase-based translations, word reordering in the target sentence, or even explicit syntactic structure of the target sentence.

A potential weakness caused by using a greedy search method is the risk of ending up with suboptimal solutions as a result of local optima in the search space. Although there is nothing that can really be done about this, one can make sure that the search space in which the decoder operates already has a certain minimum quality. Constraint satisfaction inference uses a domain-defining base classifier to define the exact solution space searched by the decoder. As the most important benefit of this, all candidate translations that are part of the solution space have been predicted and filtered based on the context of the source-language word in the input sentence. The intended effect of this is that candidate translations that are completely irrelevant for the current sentence will not be considered by the decoder, and thus local optima based on such translations are made impossible.

As another positive effect of using base classifiers to define the output space, we have witnessed that the solution space of the decoder is much smaller than the full output space resulting from considering every possible translation of the source words. Because of this, decoding will in general take less time. In the experiments reported in this chapter, we found evidence supporting this claim.

Table 6.2 Decoding speed (h:m:s) for constraint satisfaction inference and the ReWrite SMT system on the four Dutch to English translation tasks.

<table>
<thead>
<tr>
<th></th>
<th>EuroParl</th>
<th>JRC-Acquis</th>
<th>EMEA</th>
<th>OpenSubtitles</th>
</tr>
</thead>
</table>

In conclusion, one can hardly think of a better touchstone than machine translation, an area that deals with huge output spaces that are costly to search, and at the same time depends strongly on the quality of the inference, since global structural dependencies play an important role in machine translation. Because of the latter, it is to be preferred that the model is restricted as little as possible, so that global dependencies of arbitrary types may be modelled. Because of the immensity of the output space, having to take measures to restrict the cost of search is unavoidable.
decoding algorithm as the constraint satisfaction inference system, turned out to be much slower than our system. Given that the same algorithm is used in both systems, this can only be caused by the much larger output space considered by the ReWrite system. The faster translation speed of constraint satisfaction inference does not cause a degradation in translation quality though. On three out of the four data sets, constraint satisfaction inference outperforms the ReWrite decoder.
Chapter 7

Conclusions

Many processing tasks in natural language processing involve mappings between complex structures. We started this dissertation by introducing the area of structured prediction. This emerging subarea of machine learning is based on the assumption that learning approaches to complex mappings better deal with this in an integrated fashion, rather than dividing the problem, and solving the pieces separately. After having introduced the area, we also pinpointed several issues related to a structured approach to complex mappings. Existing structured prediction methods suggest solutions for some of those, but other issues remain. Moreover, the proposed solutions are often based on restricting assumptions that limit the types of dependencies that can be modelled. While making such assumptions is inevitable, different assumptions may lead to considerably different results. To gain a better understanding of the role of these issues in structured prediction, we proposed to develop a novel framework for structured prediction.

7.1 Discussion of the research objectives

The most important aim of this study was to explore whether a structured prediction framework based on design decisions different from those of existing frameworks could circumvent some of the issues with those frameworks. To do so, we formulated three research objectives. Below, we restate these objectives, and explain in what way constraint satisfaction inference manages to meet them.

Efficient inference without restricting assumptions

A typical structured prediction task has an output space that is exponential in the number of micro-labels. Searching this space may turn into a severe performance bottleneck if implemented naively. This is already the case for sequence labelling, as treated in Chapter 4, where the output space contains all possible label assignments to each input token. For all but the shortest sequences, searching this space exhaustively is infeasible. An even larger output space is encountered
Chapter 7. Conclusions

in dependency parsing, the topic of Chapter 5. Again, exhaustively searching all possible dependency trees for a sentence is not an option. While these two output spaces are large, the output space of the machine translation task of Chapter 6 is effectively infinitely large; every target-language sentence is a potential translation. There is no approach to any of the tasks mentioned that searches the output space exhaustively. Each technique attempts to restrict the cost of searching the output space, but the strategy for doing so is different for each technique.

Given that the aim is to restrict the cost of searching the output space, there is still a wide variety of ways to proceed. However, it could be argued that there are two, not necessarily mutually-exclusive, strategies for attaining efficient inference. One possibility is to make assumptions about certain properties of the output structure that restrict the part of the true output space to be searched, and may even enable optimal search in this space. The most relevant example of this for structured prediction is the Markov assumption, which allows a dynamic programming solution for searching the output space, e.g. the Viterbi algorithm for sequence labelling, and the Eisner and Chu-Liu-Edmonds algorithms for dependency parsing. Assumptions that allow for such dynamic programming solutions are most desirable from an efficiency point of view, and in fact, the Markov assumption has empirically been shown to be appropriate for many natural language processing tasks. Nevertheless, the Markov assumption could be considered an artificial one, and it is not unlikely that, at some point, it may hinder further performance improvements. For example, Finkel et al. (2005) show that modelling long-distance dependencies significantly improves the performance on two information extraction tasks. Modelling such dependencies violates the Markov assumption, though, and therefore the Viterbi algorithm cannot be used for inference; Finkel et al. chose simulated annealing, an approximate search algorithm, instead.

As an alternative to assumptions to restrict the part of the output space to be searched, some approaches use approximate search algorithms for inference. Approximate algorithms are arguably the only option for machine translation. The most popular approach to decoding for machine translation is based on A* search with a beam to cut off unlikely solutions (Koehn, 2004). Another example of approximate decoding is the greedy decoder of Germann (2003). Greedy search is essentially also the search algorithm behind Searn and the dependency parser of Nivre et al. (2004); there, the base classifier decides which action to take next. In the case of Searn, the uncertainty that is associated with those classifications is already taken into account at training time. The result is fast inference. Moreover, because of the absence of any restricting assumptions, classifications can be based on the entire partial structure generated so far. This is far more flexible than approaches based on a Markov assumption. On the downside, approximate search algorithms are slightly problematic from a theoretical stance. One cannot always be sure how close the approximate solution will be to the truly optimal solution. Moreover, if approximate inference is used instead of exact inference during training, learning quality may suffer (Kulesza and Pereira, 2008).
7.1. Discussion of the research objectives

The need for search, or inference, in structured predictions originates in having to deal with uncertainty in local classifications. Searn assumes that all uncertainty has been resolved during training, and therefore, does not need any inference at all. Structured linear models are almost at the other extreme; every possible output structure that adheres to the assumptions is considered. This makes the inference either expensive, or restricted. It is, however, questionable whether it is necessary to search the complete output space. With structured linear models, the base model is solely used as a scoring function, not as a local classifier that could already make definite decisions about the micro-label assignment at a given position. Instead, every possible assignment is considered, even if a local model can already estimate that only one or a few labels are worth considering. If we look at the performance of the baseline systems in Chapter 4, we can see that, even though they perform substantially worse than structured alternatives, they are still right most of the time. In fact, decades of machine learning research has invested substantial effort in attaining just that: classifiers that are right most of the time. The question then is, should their predictions really be doubted as much as structured linear models do?

In constraint satisfaction inference, the strategy is not to doubt local classifications at all. The domains of inference variables are only composed of values extracted from classifications made by a base classifier. Consequently, if a certain value for a micro-label is not predicted by a base classifier, it is simply not considered during inference. Furthermore, base classifications are turned into soft constraints on the output, which is once more motivated by the assumption that local classifiers are right most of the time. Still, it is important to model the uncertainty of local models with respect to the global output structure in some way—this is what makes structured prediction different from conventional multi-class classification. In constraint satisfaction inference, we do not model this uncertainty by considering other classes than the ones predicted by the base classifiers. Instead, we have base classifiers predict potentially conflicting subparts of the output structure. This is the case when predicting overlapping trigrams for sequence labelling. It also happens when making dependency predictions for dependency parsing based on down-sampled training data, or when disabling the tie-breaking mechanism in predicting target-language trigrams in machine translation. Then, uncertainty is modelled by potentially conflicting classifications that are each right most of the time. Thus, if they do not agree, that is evidence for uncertainty.

Figures 4.6 and 4.7 in Chapter 4 illustrate the effect of modelling uncertainty this way. The better the base classifier is, the smaller the uncertainty, and as a result, the cheaper searching the output space is. In the case of sequence labelling, the output space for all tasks was even so small that exhaustive search was possible. The result of this is that arbitrary dependencies among parts of the output structure can be modelled. Also, for machine translation, because only candidate translations actually predicted by the base classifier were considered, decoding speed was considerably higher than that of the otherwise similar decoder.

**Efficient training**

Our survey of existing approaches to structured prediction in Chapter 2 shows that many of those techniques have training procedures that are more expensive than for conventional multi-class classification. Structured linear models require an inference step as part of the parameter training. Searn trains a conventional multi-class base classifier, but does so several times, each time training on the output of the previous classifier. It is easy to see that such extended training procedures cause a severe efficiency penalty. Of course, in the end, classification speed is more important than training efficiency, but if the micro-label space is rather large as well, as was the case for the morphological analysis and letter-phoneme conversion tasks in Chapter 4, training times can get impractically long, which hinders the possibility for empirical optimisation.

Training for constraint satisfaction inference requires nothing more than training the multi-class base classifiers; inference is only needed for prediction. This makes training vastly more efficient than for structured linear models and for Searn. Still, the exact cost of training depends on the number of base classifiers, and the number of instances they extract from the training data. Nevertheless, the advantage remains that however many base classifiers are to be trained, they are all trained independently of one another. Thus, the extra overhead associated with adding a base classifier corresponds to the multi-class training cost for that classifier.

In terms of training time, constraint satisfaction inference compares advantageously with structured linear models and Searn, which require extended training procedures. A relevant question then is, why extended training procedures are required for those methods, and why constraint satisfaction inference can do without? In structured linear models and Searn, there is a clear need for this extra training effort. Both make their predictions based on features of the, possibly partial, output structure, as well as of the input. Without an extended training procedure, classifiers would be trained on gold-standard information about those features. As a result, the trained models would suffer from a label bias. It is not surprising that features of the output are so effective for structured prediction, yet constraint satisfaction inference seems to do rather well without such features. It could be argued that the meso-label predictions, which are turned into constraints, encode similar information as is encoded by output features; however, the fact that this information is part of the predicted class, rather than of the input features pays off in terms of more efficient training.

**Compatibility with existing learning techniques**

All experiments reported in this dissertation have been performed with memory-based classifiers. The framework itself does not require that, though. In fact,
only two requirements are imposed upon a base classifier. It should be able to
make multi-class predictions, and it should be able to express its confidence in
those predictions. Most general-purpose classification methods adhere to these
requirements. Therefore, in the experiments in the previous chapters, we could
easily have replaced the memory-based classifiers by, say a support vector machine
or a decision tree.

Given the implementation details of constraint satisfaction inference it is quite
obvious that almost any type of classifier is usable for implementing the constraint
prediction. However, this flexibility is not present in all structured prediction ap-
proaches. Most notably, structured linear models require linear models; adapting
the model for other types of classifiers would be non-trivial at least. Structured
prediction based on output kernels only applies to regression models. Further-
more, given the high dimension of the output space of those models, the technique
should be able to scale to such dimensions, and preferably learn sparse models, so
that large amounts of training data can be used. Other existing methods are more
flexible, though, in the requirements on the base classifier. The requirements for
the Constraint Satisfaction with Classifiers framework are similar to ours, that
is, multi-class predictions with confidences. Searn is also independent of the type
of base classifier. It does not even expect confidence estimates. The absence of
strong requirements on the base classifier is an advantage with regard to future
extendability. If new types of learning techniques arise in the future, they are
immediately usable for structured prediction problems.

As another advantage of the loose requirements on the types of base classifiers,
constraint satisfaction inference enables an interesting form of classifier combina-
tion. The approach to dependency parsing described in Chapter 5 integrates
multiple base classifiers. All of those are memory-based, but they do not need
to be. As mentioned previously, all that is required of a base classifier is that
it can make multi-class predictions, and in some way estimate its confidence in
those. Thus, if we have two different types of base classifiers that both meet those
requirements, it should—at least in theory, we did not verify this empirically—be
possible to integrate predictions of both classifiers in the same inference pro-
cedure. Doing so is possible along two options: both types of classifier are employed
to predict all constraints, or some constraints are predicted by one type of classi-
fier, while the remaining are predicted by the other type. These two options are
not mutually exclusive. It could be decided to have one constraint be predicted
by one type of classifier, another constraint by another type, and yet another
constraint by both.

As for the case where two types of classifiers predict the exact same constraints,
a memory-based classifier and a support vector machine could both predict depend-
dency, modifiers, and direction constraints. Most likely, this leads to even more
conflicting results, but that is something that might turn out to be beneficial;
recall that some degree of conflict is even necessary for the inference procedure
to have any choice. This option for integrating multiple different classifiers is
somewhat related to classifier ensembles (Hansen and Salamon, 1990; Dietterich,
2002a), although the combination method is arguably more complex in constraint satisfaction inference.

With regard to this option, it is important to point at the difference between constraint-defining and domain-defining base classifiers. For the experiments performed in Chapter 5, the classifier predicting the dependency constraint was at the same time the classifier used for composing the inference variable domains, that is, the classifier was both constraint-defining and domain-defining. If, as we suggest here, we have two classifiers predict dependency constraints, we can opt for making both, or only one of them domain-defining. The former may have the advantage of resulting in a larger solution space that contains better solutions than the one spanned by the predictions of only one of the two classifiers—this will be apparent from a higher oracle performance. On the other hand, the solution space may also become too large, causing slow inference, and even worse, reduced parsing performance.

Instead of all classifiers predicting all constraints, we can also have some constraints predicted by one type of classifier, and some constraints by another type. For example, suppose it is found that support vector machines are actually superior in predicting dependency constraints, while they are outperformed by memory-based classifiers for predicting modifiers and direction constraints. Then, it might be worthwhile to have each constraint predicted by the type of classifier that has been found to be the best option for it.

For classifier combination to work, it may be necessary to calibrate the confidence estimates of the different classifier types. So far, the only real demand on the confidence estimates has been that higher confidence values suggest better predictions. The comparison of two values is purely relative, and the exact numbers are of lesser importance. Thus, if the same classifier is used, confidence values can safely be compared. Even when using several classifiers of the same type, this appears to be possible, as we observed in Chapter 5. However, it may not be the case for comparing confidence estimates of different types of classifiers. Then, some measure has to be taken to make those estimates more compatible. This could be as simple as assigning a linear weighting factor to each classifier and optimising the value of those on held-out training data. Alternatively, more complex scaling techniques could be applied, such as Platt scaling (Platt, 1999) or isotonic regression (Zadrozny and Elkan, 2001, 2002).

### 7.2 How generic is constraint satisfaction inference?

In Chapter 3 we presented constraint satisfaction inference as a generic framework for structured prediction, and in subsequent chapters we applied it to a diverse selection of structured prediction tasks. Tackling those tasks sometimes involved jumping into task-specific details. Since attaining state-of-the-art performance on any of the tasks was not our primary purpose, we did not exhaust the available
task-specific optimisation options, yet it is evident that in each of the application-oriented chapters, some design decisions had to be made. At the end of this dissertation, it is time to return to the higher-level view of structured prediction. Did we succeed in devising a framework that is truly generically applicable to any structured prediction task? What is the task-specific effort that still has to be invested for each individual task?

Constraint satisfaction inference is not a single algorithm that can be handed arbitrary input objects and then be instructed to produce a phoneme sequence, a dependency tree, or an English translation. The way that it should be seen is as a framework. Most importantly, it provides a formalism—in the form of the constraint satisfaction problem—that allows the output space to be defined in a formal way, and constraints, whether predicted or manually inserted, to be defined over it. The constraint satisfaction problem formulation of structured prediction is what makes all approaches presented in this dissertation a truly unified approach. Once a constraint satisfaction problem has been constructed, one could opt for employing a generic constraint satisfaction solver to find the output structure—this would mean that the same algorithm could be used for every task—or, with the no free lunch theorem (Wolpert and Macready, 1997) in mind, one could select the best solving algorithm for the task at hand, as we have done for dependency parsing and machine translation. In case optimal solving is possible, this difference may merely be one of efficiency, though in the approximate case it may also affect prediction quality.

Task-specific effort remains on the route that leads to constructing the constraint satisfaction problem. No specific procedure can explain the exact number of base classifiers one must employ, nor the types of constraints those classifiers must predict. Nevertheless, general guidelines can be given for what design choices may lead to a successful application; here, we describe the two most important ones. The first guideline specifically relates to domain-defining base classifiers.

Since these are responsible for constructing the micro-label domains, they are the key ingredients for modelling the uncertainty to be resolved during inference. A crucial observation is that if no domain-defining predictions are in conflict, there is no uncertainty to be resolved by the inference procedure, and the approach is essentially demoted to a simple combination of local classifiers. Therefore, good domain-defining base classifiers make overlapping predictions that are potentially conflicting. This is illustrated by Chapter 5, in which stronger down-sampling of the training data results in increasing overprediction of dependency constraints, and the performance of the dependency parser actually improves because of it. Overlapping and conflicting predictions are also at the basis of the constraint satisfaction approaches to sequence labelling and machine translation. In the former, overlapping trigram predictions may give rise to multiple candidate labels for every given micro-label; in the latter, disabling the tie-breaking mechanism of the base classifier serves a similar purpose.

Whereas the first guideline is mainly concerned with modelling uncertainty, the second guideline is about resolving it. Additional constraints may be added to
the inference procedure to help resolving the uncertainty yielded by the domain-defining base classifier. In Chapter 5, we showed how adding modifiers and direction constraints to the inference procedure for dependency parsing improved performance in comparison with a dependency parser based on dependency constraints only. The motivation for adding several different types of constraints, each covering interrelated aspects of the output structure, is that while individual classifiers may be incorrect, it is much less likely that an incorrect micro-label assignment satisfies all constraints that cover it. The number and types of constraints that one can add is virtually unlimited, and is typically determined based on task-specific considerations.

The final question then is, what does this task-specific effort say about the genericity of constraint satisfaction inference? As already said, constraint satisfaction inference is a framework, not an algorithm. The sheer diversity of the problems that can be seen as structured prediction tasks, however, arguably make it impossible to come up with such an algorithm. In fact, other generic approaches to structured prediction, e.g. structured linear models, Searn, output kernels, are all frameworks, rather than algorithms. None of them enable state-of-the-art performance out of the box, but all of them do pay a significant contribution to attaining such performance by exploiting the structural properties of output structures ignored by traditional machine learning techniques. Constraint satisfaction inference is a novel addition to this family of approaches, it is ready to be applied to new problems, and will hopefully contribute to an improved understanding of structured prediction.
References


References


Summary

Many natural language processing tasks are concerned with the mapping between input and output values that have complex structures. For example, in syntactic parsing, a sentence is mapped to a parse tree; in letter-phoneme conversion, a word is mapped to its phonemic transcription; and in machine translation, a sentence in one language is mapped to a sentence in another language. Even though machine learning approaches have been successfully employed for performing linguistic processing tasks for many years, performance is sometimes hindered by the fact that traditional machine learning approaches learn simple decision functions aimed at discriminating between small sets of target values, an output space that is much simpler than the complex output spaces of typical natural language processing tasks. Structured prediction, an emerging subfield of machine learning, specifically deals with learning in the types of complex output spaces that are prevalent in natural language processing, but also in for example, bioinformatics and computer vision.

In this dissertation, we present constraint satisfaction inference (CSI) as a new framework for structured prediction, which was designed with three objectives in mind. First, the approach should allow for efficient search in structured output spaces without severely restricting the structural dependencies that can be modelled. Secondly, training of structured prediction models should not be more expensive than training of traditional multi-class classification models. Thirdly, the structured prediction framework should be compatible with arbitrary existing learning techniques, such that it can be used with any learning algorithm that is able to learn multi-class classification tasks.

In Chapter 2, we define structured prediction as the joint prediction of multiple interrelated output values. This abstract definition covers all the popular structured prediction tasks in natural language processing such as syntactic parsing, letter-phoneme conversion and information extraction. After this definition, we provide an overview of existing work in structured prediction. All approaches to structured prediction somehow divide the structured output value into smaller output values, which can be treated as multi-class classification cases. Yet, the approaches differ with respect to how the uncertainty resulting from those local classifications is resolved. Prior to the introduction of structured prediction techniques, such uncertainty resolution came down to selecting the value deemed
most likely by the local classifier. In most structured prediction techniques, all local uncertainty is delegated to an inference procedure that takes into account structural dependencies between the output variables to resolve local uncertainty in a way that optimises the quality of the global output structure. While shown to be effective, such approaches have the disadvantage of making this inference step costly, since the latter essentially has to explore the complete output space of the structured prediction task in order to evaluate what local value assignments optimise global output quality. Some structured prediction techniques perform full inference not only for prediction, but as part of their training procedure as well, which makes training prediction models rather expensive. The only option to keep inference procedures efficient is to make assumptions about the dependencies that exist between output variables. Unfortunately, those assumptions restrict the types of structural information that can be taken into account during inference, and therefore, may harm the performance.

Chapter 3 presents the constraint satisfaction inference framework, which aims to resolve the abovementioned issues with existing techniques. Structured prediction in constraint satisfaction inference proceeds according to two steps. First, one or more base classifiers predict a weighted constraint satisfaction problem (WCSP) based on the input. Secondly, a constraint solver is employed to solve the WCSP and obtain the structured output value. Given the extensive research on constraint satisfaction and combinatorial optimisation, many algorithms exist to perform the latter step. The main contribution of this dissertation is in the way a weighted constraint satisfaction problem is formulated in order to map input instances to structured output values.

Central to the structured prediction approach in constraint satisfaction inference is a different way of modelling global uncertainty. Like most other structured prediction techniques, structured output values are decomposed into smaller classification cases. However, uncertainty in predictions made by base classifiers is always resolved locally. As an alternative means to communicate local uncertainty to the inference procedure, base classifiers are instructed to predict overlapping and potentially conflicting parts of the output structure. The uncertainty to be resolved by the inference procedure, then, corresponds to conflicts between such predictions. The solution space resulting from this method tends to be much smaller than the true output space of the task, and as a result, inference may be considerably cheaper. In some cases, even exhaustive search of this space is possible. Therefore, arbitrary structural dependencies may be taken into account during inference. Modelling such dependencies takes the form of soft constraints on the output space predicted by base classifiers, and enforced by the inference procedure.

To evaluate the potential of the constraint satisfaction inference framework, we apply it to three challenging natural language processing tasks that deal with structured output spaces: sequence labelling, dependency parsing, and machine translation.

Chapter 4 reports on experiments with four processing tasks that can be for-
mulated as sequence labelling problems: syntactic chunking, named-entity recognition, letter-phoneme conversion, and morphological analysis. When applied to these tasks, constraint satisfaction inference attains results comparable to other state-of-the-art structured prediction approaches. Importantly, because of the way constraint satisfaction inference defines the search space of the inference procedure, inference for all four tasks could be performed by exhaustive search, whereas many structured prediction approaches rely on the Viterbi algorithm, which severely restricts the type of structural dependencies that can be modelled.

Chapter 5 deals with dependency parsing. In this chapter, it is shown how multiple base classifiers, each predicting constraints on different aspects of the output structure, are combined by the inference procedure, and that parsing performance consistently improves with each type of constraint added. Experiments with ten different languages show that constraint satisfaction inference using only basic features attains above-average performance when compared to existing systems applied to the same data sets. Again, the alternative approach to uncertainty modelling pays off in terms of a smaller solution space for the inference procedure. Because of that, our approach is able to perform labelled dependency parsing in a single step, whereas comparable dependency parsers require two steps: one to find an unlabelled dependency tree, and another to label its edges.

Chapter 6 describes how constraint satisfaction inference can be applied to machine translation, the most challenging of the processing tasks tackled in this dissertation. Its output space is effectively infinitely large, since any sentence in the target language might be a potential translation of the input sentence. For this reason, it is essential to restrict the cost of inference, one of the main objectives of constraint satisfaction inference. In experiments with four different Dutch to English translation tasks, we compared constraint satisfaction inference with an existing statistical machine translation system, and found that constraint satisfaction inference succeeds in meeting this objective for machine translation as well. Our approach to machine translation is faster than the competing system, while at the same time, it produces better translations on three out of four translation tasks.

In Chapter 7, we conclude that constraint satisfaction inference meets all three of the research objectives formulated for this study. Given that throughout this dissertation, experiments with a variety of structured prediction tasks have consistently shown that good performance is attainable with constraint satisfaction inference, we argue that constraint satisfaction inference is a structured prediction framework that can compete with existing state-of-the-art approaches, and thus, that trade-offs different from those made in the design of existing structured prediction techniques can still lead to state-of-the-art performance. These findings may contribute to an improved understanding of the issues surrounding structured prediction.
Samenvatting

Veel taken in de natuurlijke taalverwerking draaien om het omzetten van invoer- naar uitvoerwaarden die complexe structuren hebben. Bijvoorbeeld, bij grammaticaal ontleiden wordt een zin omgezet naar een grammaticale boomstructuur; bij letter-foneemconversie wordt een woord omgezet naar zijn fonetische transcriptie; en bij automatisch vertalen wordt een zin in de ene taal omgezet naar een zin in een andere taal. Benaderingen gebaseerd op automatisch leren worden al jaren succesvol ingezet voor het uitvoeren van taalkundige verwerkingstaken. Niettemin worden de prestaties soms belemmerd door het feit dat traditionele benaderingen van automatisch leren eenvoudige beslisfuncties leren die gericht zijn op het onderscheiden van kleine aantallen doelwaarden. Een dergelijke uitvoerruimte is veel eenvoudiger dan de complexe uitvoerruimten van typische natuurlijke taalverwerkingstaken. Structured prediction, een opkomend deelgebied van het automatisch leren richt zich specifiek op het leren in het soort complexe uitvoerruimten dat alom aanwezig is in de natuurlijke taalverwerking, maar ook in bijvoorbeeld de bioinformatica en automatische beeldherkenning.

In dit proefschrift presenteren we constraint satisfaction inference (CSI) als een nieuwe methode voor structured prediction. Het is ontworpen met drie doelstellingen voor ogen. Ten eerste moet de aanpak het mogelijk maken om efficiënt te kunnen zoeken in gestructureerde uitvoerruimten zonder de structurele afhankelijkheden die gmodelleerd kunnen worden te beperken. Ten tweede mag het trainen van structured prediction modellen niet kostbaarder zijn dan het trainen van traditionele meerklaasse classificatiemodellen. Ten derde moet de structured prediction methode samen kunnen werken met willekeurige bestaande leertecnieken zodat het gebruikt kan worden in combinatie met ieder leeralgoritme dat meerklaasse classificatietaakken kan leren.

In Hoofdstuk 2 definiëren we structured prediction als het gelijktijdig voorspellen van verschillende aan elkaar gerelateerde uitvoerwaarden. Deze abstracte definitie dekt alle populaire structured prediction taken in de natuurlijke taalverwerking zoals grammaticaal ontleden, letter-foneemconversie en informatieextractie. Na deze definitie presenteren we een overzicht van bestaand werk op het gebied van structured prediction. Elke aanpak van structured prediction deelt de gestructureerde uitvoerwaarde op een of andere manier op in kleinere uitvoerwaarden, die behandeld kunnen worden als meerklaasse classificatiegevallen. De
aanpakken verschillen echter met betrekking tot hoe de onzekerheid die ontstaat bij die lokale classificaties wordt opgelost. Voordat structured prediction technieken geïntroduceerd werden, werd zulke onzekerheid opgelost door de waarde te kiezen die de lokale classifier het waarschijnlijkst acht. In de meeste structured prediction technieken wordt alle lokale onzekerheid doorgeslipt naar een inferentieprocedure die structurele afhankelijkheden betrekt in het oplossen van de lokale onzekerheid op een manier die de kwaliteit van de globale uitvoerstructuur optimaliseert. Hoewel zulke aanpakken effectief gebleken zijn, hebben ze het nadeel dat deze inferentiestap kostbaar is. Deze moet immers de volledige uitvoerruimte van de structured prediction taak onderzoeken om te bepalen welke lokale waardetoewijzing de kwaliteit van de globale uitvoer optimaliseert. Sommige structured prediction technieken gebruiken niet alleen inferentie voor het voorspellen, maar ook als onderdeel van hun trainingsprocedure, wat het trainen van voorspelmodellen erg kostbaar maakt. De enige manier om inferentieprocedures efficiënt te maken is het maken van aannamen over de afhankelijkheden die tussen uitvoervariabelen bestaan. Helaas beperken zulke aannamen de soorten structurele informatie die tijdens de inferentie gebruikt kan worden als gevolg waarvan de prestaties negatief beïnvloed kunnen worden.

Hoofdstuk 3 presenteert de constraint satisfaction inference methode, die bedoeld is om de hierboven genoemde problemen met bestaande technieken op te lossen. Structured prediction met constraint satisfaction inference gebeurt volgens twee stappen. Eerst voorspellen een of meer basisclassifiers op basis van de invoer een weighted constraint satisfaction probleem (WCSP). Vervolgens wordt een constraint satisfaction oplosalgoritme gebruikt om de WCSP op te lossen en zodoende de gestructureerde uitvoerwaarde te verkrijgen. Dankzij het vele onderzoek dat verricht is naar constraint satisfaction en combinatoriële optimalisatie zijn er vele algoritmen voorhanden om deze laatste stap uit te voeren. De voornaamste bijdrage van dit proefschrift is de manier waarop een weighted constraint satisfaction probleem geformuleerd wordt om invoerinstanties om te zetten naar gestructureerde uitvoerwaarden.

De belangrijkste eigenschap van constraint satisfaction inference is de alternatieve manier waarop globale onzekerheid gemodelleerd wordt. Net als andere technieken voor structured prediction worden gestructureerde uitvoerwaarden opgedeeld in kleinere classificatiegevallen. De onzekerheid bij voorspellingen van basisclassifiers wordt echter altijd lokaal opgelost. Als een alternatieve manier om de inferentieprocedure te informeren over lokale onzekerheid worden basisclassifiers ingezet om overlappende en mogelijk conflicterende delen van de uitvoerstructuur te voorspellen. De onzekerheid die de inferentieprocedure moet oplossen komt dan overeen met de conflicten tussen die voorspellingen. De oplossingsruimte die voortkomt uit deze methode is vaak veel kleiner dan de werkelijke uitvoerruimte van de taak en daarom kan inferentie veel minder kostbaar zijn. In sommige gevallen zal het zelfs mogelijk zijn om deze ruimte volledig te doorzoeken. Als gevolg hiervan kunnen willekeurige structurele afhankelijkheden in beschouwing genomen worden tijdens de inferentie. Het modelleren van zulke afhankelijkheden neemt
Samenvatting

161

de vorm aan van ‘zachte’ constraints op de uitvoerruimte die voorspeld worden door de basisclassifiers en vervolgens afgedwongen door de inferentieprocedure.

Om het potentieel van de constraint satisfaction inference methode te evalueren hebben we deze toegepast op drie uitdagende natuurlijke taalverwerkingsstaken: sequentielabelen, grammaticaal ontleden en automatisch vertalen.

Hoofdstuk 4 beschrijft experimenten met vier verwerkingstaken die geformuleerd kunnen worden als sequentielabelproblemen: oppervlakkig ontleden, opsporen van namen, letter-foneemconversie en morfologische analyse. Wanneer de constraint satisfaction inference methode toegepast wordt op deze taken behaalt zij resultaten die vergelijkbaar zijn met die van andere state-of-the-art technieken voor structured prediction. Door de manier waarop constraint satisfaction inference de zoekruimte van de inferentieprocedure vormgeeft, was het mogelijk om de inferentie voor alle vier de taken te doen op basis van volledig zoeken. Veel andere aanpakken voor structured prediction maken gebruik van het Viterbi-algoritme, dat de soorten structurele afhankelijkheden die gemodelleerd kunnen worden sterk beperkt.

Hoofdstuk 5 gaat over grammaticaal ontleden. In dit hoofdstuk wordt getoond hoe verschillende basisclassifiers, die elk constraints voorspellen op verschillende aspecten van de uitvoerstructuur, gecombineerd worden door de inferentieprocedure en dat de kwaliteit van de geproduceerde ontleidingen verbetert met ieder type constraint dat toegevoegd wordt. Experimenten met tien verschillende taken tonen aan dat constraint satisfaction inference, gebruik makend van slechts eenvoudige informatie, bovengemiddeld presteert in vergelijking met bestaande systemen die toegepast worden op dezelfde data. Ook hier resulteert de alternatieve manier om onzekerheid te modelleren in een kleinere oplossingsruimte voor de inferentieprocedure. Daarom kan onze aanpak in een enkele stap gelabelde grammaticale structuren produceren. Vergelijkbare grammaticale ontleiders hebben hiervoor twee stappen nodig: een om een ongelabelde grammaticale boomstructuur te vinden en een andere om de onderdelen van die structuur van een label te voorzien.

Hoofdstuk 6 beschrijft hoe constraint satisfaction inference toegepast kan worden om automatisch te vertalen, de uitdagendste verwerkingstaak die we in dit proefschrift behandelen. De uitvoerruimte van deze taak is in essentie onbeperkt groot, aangezien iedere zin in de doeltaal een vertaling zou kunnen zijn van de invoerzin. Daarom is het essentieel dat de kosten van inferentie beperkt worden, wat een van de hoofddoelen van constraint satisfaction inference is. In experimenten met vier verschillende Nederlands-Engels vertaaltaken hebben we constraint satisfaction inference vergeleken met een bestaand statistisch vertaalsysteem en vastgesteld dat constraint satisfaction inference ook deze doelstelling haalt. Onze aanpak is sneller dan het systeem waarmee we vergeleken zonder dat dit ten koste gaat van de kwaliteit: voor drie van de vier taken produceert ons systeem betere vertalingen.

In Hoofdstuk 7 concluderen we dat constraint satisfaction inference voldoet aan alle drie de doelen die we voor deze studie geformuleerd hebben. Aangezien
Samenvatting

de in dit proefschrift beschreven experimenten met een diversiteit aan structured prediction taken zonder uitzondering aangetoond hebben dat met constraint satisfaction inference goede resultaten behaald kunnen worden, betogen we dat constraint satisfaction inference een methode voor structured prediction is die kan wedijveren met bestaande state-of-the-art methoden en dus dat andere ontwerpkeuzen dan gemaakt voor bestaande technieken toch kunnen resulteren in state-of-the-art prestaties. Deze bevindingen kunnen bijdragen aan een beter begrip van de vraagstukken rondom structured prediction.
1. Pashiera Barkhuysen
   *Audiovisual prosody in interaction*
   Promotores: M.G.J. Swerts, E.J. Krahmer
   Tilburg, October 3, 2008

2. Ben Torben-Nielsen
   *Dendritic morphology: function shapes morphology*
   Promotores: H.J. van den Herik, E.O. Postma
   Co-promotor: K.P. Tuyls
   Tilburg, December 3, 2008

3. Hans Stol
   *A framework for evidence-based policy making*
   Promotor: H.J. van den Herik
   Tilburg, January 21, 2009

4. Jeroen Geertzen
   *Act recognition and prediction. Explorations in computational dialogue modelling*
   Promotor: H.C. Bunt
   Co-promotor: J.M.B. Terken
   Tilburg, February 11, 2009

5. Sander Canisius
   *Structured prediction for natural language processing: a constraint satisfaction approach*
   Promotores: A.P.J. van den Bosch, W.M.P. Daelemans
   Tilburg, February 13, 2009