Classifier Combination Systems and their Application in Human Language Technology

László Felföldi

Research Group on Artificial Intelligence

May 2008

A dissertation submitted for the degree of doctor of philosophy of the University of Szeged

University of Szeged
Doctoral School in Mathematics and Computer Science
Ph.D. Programme in Informatics

Preface

For years, the pattern recognition community has focused on developing optimal learning algorithms that can produce very accurate classifiers. However, experience has shown that it is often much easier to find several relatively good classifiers instead of one single very accurate predictor. The advantages of using classifier combinations instead of a single one are twofold: it helps reducing the computational requirements by using simpler models, and it can improve the classification performance.

Most Human Language Technology applications are based on pattern matching algorithms, thus improving the performance of the pattern classification has a positive effect on the quality of the overall application. Since combination strategies proved very useful in reducing classification errors, these techniques have become very popular tools in applications such as Speech Technology and Natural Language Processing.

The aim of this dissertation is basically to investigate suitable combination techniques for Human Language Technology applications. We propose a novel combiner algorithm based on the Analytical Hierarchy Process, and apply different ensembles in Speech Technology and Natural Language Processing in order to improve their performance.

The first of my acknowledgements goes to my supervisor, Prof. János Csirik for supporting my research and letting me work at an inspiring department, the Research Group on Artificial Intelligence. Secondly, I would like to thank András Kocsor for the fruitful discussions and for encouraging me to pursue my studies. My heartfelt thanks goes to all my colleagues in the Speech Technology Team, who offered an invaluable amount of help and assistance in getting the results presented in this dissertation. I would also like to express my gratitude to David P. Curley for scrutinizing and correcting this dissertation from a linguistic point of view.

László Felföldi, May 2008.

Notations used

M	number of classes
N	number of patterns in the database
R	number of classifiers
T	number of iterations (for Bagging a Boosting)
$\mathcal{C}_i$	the ith classifier
$\mathcal{I}$	the learning algorithm that creates classifiers $\mathcal{C}_i$
$\mathbf{x}$	a pattern
$\mathbf{x}^{(i)}$	feature vector of pattern $\mathbf{x}$, employed by the ith classifier $\mathcal{C}_i$
$\mathbf{x}_i$	the ith pattern of the database
ω_j	the jth class label
y_i	the label (or its numeric representation) of the ith pattern
w_i	the weight of the ith classifier in linear combinations

List of Figures

1.1 The general model of a pattern recognition system. 2
1.2 Parallel combination scheme 3
1.3 Cascading combination scheme 3
1.4 Hierarchical combination scheme 3
1.5 Information levels of a typical classifier 4

3.1 A simple AHP hierarchy. In practice, many criteria have one or more layers of subcriteria. These are not shown in this simplified diagram. Also, to avoid clutter in AHP diagrams, the lines between the alternatives and criteria are often omitted or reduced in number. 22

4.1 The three-state left-to-right phoneme HMM. 32
4.2 Modular structure of the OASIS Speech Lab 33

6.1 Vocal Tract Length and its frequency shifting. The graph drawn with solid and dashed line shows the spectrum of a vowel uttered by a man and a girl, respectively. 48
6.2 Examples of VTL warping functions. The figures show the mapping between the original (horizontal axis) and the warped (vertical axis) frequencies. 49

7.1 Classification error of *Bagging* and *Boosting* algorithm on the training and training datatsets (dashed: Bagging, solid: Boosting). 61

8.1 A parsing tree of a Hungarian sentence (with its English equivalent) from the Szeged Corpus "Short Business News" 65
8.2 A tree pattern learning example. 67

List of Tables

3.1 Classifier sets for the *speech* and *letter* databases. 27
3.2 Classifier sets for the *satimage* database. 28
3.3 Classification errors [%] on the Speech database (Error without combination: 12.92%) 28
3.4 Classification errors [%] on the Letter database (Error without combination: 13.78%) 28
3.5 Classification errors [%] on the Satimage database. (Error without combination: 12.05%) 29

5.1 Classification errors of the individual classifiers 43
5.2 Combination error obtained using the Product Decision Rule 44
5.3 Classification error of hybrid combinations using ANN, SVM, and kNN 45
5.4 Classification error of Bagging classifiers 45
5.5 Classification error of Boosting classifiers 46

6.1 Recognition accuracies of the standalone classifiers (in percent). The parameter estimation of LD-VTLN requires all pattern data in advance, thus this method cannot be utilized in real recognition systems, its performance can be regarded as a reference value for the other normalization techniques. 52
6.2 Recognition accuracy on the databases with combination (in %). The various bars in each triplet correspond to the databases, and bar-triplets represent the applied combiner. 53

7.1 Accuracy of the baseline tagger and the TBL tagger. 57
7.2 Results achieved by various Hungarian POS taggers. 57

8.1 Results achieved by combining RGLearn parsers 69

C.1 The relation between the thesis topics and the corresponding publications 87

Contents

Notations v

1 Multiple-Classifier Systems 1
- 1.1 Pattern Recognition . . . 1
- 1.2 Multiple-Classifier Systems . . . 2
  - 1.2.1 Combination Architectures . . . 2
  - 1.2.2 Types of Knowledge Sources . . . 4
  - 1.2.3 Generating effective Classifier Sets . . . 4
- 1.3 Static Combination Schemes . . . 5
  - 1.3.1 Product Rule . . . 6
  - 1.3.2 Sum Rule . . . 6
  - 1.3.3 Max, Min Rule . . . 7
  - 1.3.4 Median Rule . . . 7
  - 1.3.5 Voting Rule . . . 7
  - 1.3.6 Borda count . . . 8
- 1.4 Summary . . . 8

2 The Additive Combination Model 9
- 2.1 Linear Combinations . . . 9
- 2.2 Theoretical background . . . 10
- 2.3 Simple Adaptive Combinations . . . 13
- 2.4 Bagging . . . 14
- 2.5 Random Forests . . . 15
- 2.6 Boosting . . . 15
  - 2.6.1 Adaboost . . . 15
  - 2.6.2 Discrete Adaboost . . . 16
  - 2.6.3 Real Adaboost . . . 17
  - 2.6.4 Generalization error . . . 18
- 2.7 Summary . . . 19

3 Combinations and the Analytic Hierarchy Process 21
- 3.1 Analytic Hierarchy Process . . . 21

3.2 Mathematical model . . . 23
3.2.1 Combinations based on AHP . . . 25
3.3 Experiments . . . 26
3.3.1 Evaluation Domain . . . 26
3.3.2 Evaluation Method . . . 27
3.3.3 Results and Discussion . . . 28
3.4 Conclusions and Summary . . . 29

4 Speech Recognition 31
4.0.1 Phoneme Modeling . . . 31
4.1 The OASIS System . . . 33
4.1.1 The Modular Structure and The Script-based User Interface . . . 34
4.1.2 Auxiliary Modules . . . 34
4.1.3 Signal Processing and Feature Extraction Modules . . . 35
4.1.4 Evaluators . . . 36
4.1.5 The Matching Engine . . . 36
4.1.6 Language Models . . . 37
4.2 Summary . . . 38

5 Phoneme Classification 39
5.1 Evaluation domain . . . 39
5.1.1 Acoustic features . . . 40
5.1.2 Classifiers . . . 41
5.1.3 Results of the standalone classifiers . . . 43
5.1.4 Selecting Classifier Set . . . 43
5.1.5 Comparing combination rules . . . 43
5.1.6 Results using Bagging . . . 44
5.1.7 Results using Boosting . . . 44
5.2 Conclusions and Summary . . . 46

6 Vocal Tract Length Normalization 47
6.1 The Model used for VTLN . . . 48
6.1.1 VTLN parameter estimation . . . 49
6.1.2 Real-time VTLN . . . 50
6.2 Evaluating Domain . . . 50
6.2.1 Feature Sets and Classifier . . . 50
6.2.2 Warping parameter estimation . . . 51
6.3 Combination strategies . . . 51
6.3.1 Multi-Model classifier . . . 51
6.4 Classification Results . . . 52
6.5 Combination results . . . 53
6.6 Conclusions and Summary . . . 54

7 POS Tagging **55**
7.1 POS Tagging of Hungarian Texts . . . 55
7.2 The TBL tagger . . . 56
7.2.1 Baseline Tagger . . . 57
7.3 Combination strategies . . . 58
7.3.1 Related works . . . 58
7.3.2 Combination strategies of the TBL tagger . . . 58
7.3.3 Context-dependent Boosting . . . 58
7.4 Experimental results . . . 60
7.5 Summary . . . 60

8 NP Parsing **63**
8.1 Related Works . . . 63
8.2 Hungarian NP Parsing . . . 64
8.3 The Training Corpus . . . 64
8.4 Learning tree patterns . . . 66
8.4.1 Preprocessing of training data . . . 66
8.4.2 Generalization of tree patterns . . . 66
8.4.3 Specialization of tree patterns . . . 66
8.4.4 Generating Probalistic Grammar Rules . . . 67
8.4.5 Syntactic parsing . . . 68
8.5 Combination Strategies . . . 68
8.6 Experiments . . . 69
8.6.1 Results . . . 69
8.7 Summary . . . 70

9 Conclusions **71**

Appendix A Databases Used in the Dissertation **73**
A.1 The OASIS-Numbers Database . . . 73
A.2 The MTBA Hungarian Telephone Speech Database . . . 73
A.3 The BeMe-Children Database . . . 74
A.4 The Szeged Corpus . . . 75
A.4.1 Text of Szeged Corpus . . . 75
A.4.2 Annotation of the Szeged Corpus . . . 76

Appendix B An Example OASIS Script **79**

Appendices **73**

Appendix C Summary in English **83**
C.1 Key Points of the Thesis . . . 86

Appendix D Summary in Hungarian **89**
D.1 Az eredmények tézisszerű összefoglalása 92

Bibliography **95**

1

Multiple-Classifier Systems

In this chapter we will review the most common classifier combination techniques used by the researchers today. In the literature a fair number of combination strategies have been proposed [22][52][59], these schemes differing from each other in their architecture, the characteristics of the combiner, and the selection of the individual classifiers. The first section will provide a general overview of these schemes, concentrating on their architecture and data representation, and discuss some of the techniques for generating an effective set of classifier instances for building combinations. After discussing the basic issues of multiple classifier systems, we will overview some of the static combination rules applied in Human Language Technology like the "Majority Voting" and "Product" rule.

1.1 Pattern Recognition

Pattern recognition (or classification) is the field of applications that seeks to learn from a given set of examples how to classify new data into a finite set of categories that are called classes. The input of a common pattern recognition system is thus an entity called a pattern (like a segment of speech signal or an image) which we would like to associate with an output class. A pattern recognition system is ususally divided into three main parts, namely preprocessing, feature selection or extraction, and classification. A pattern is represented by a set of measurements that should contain relevant information about the structure of the object that we wish to classify. The measurements can potentially be collected from a large number of data sources, which would result in a high dimensionality vector of measurements. An overall view of the main stages of a pattern recognition system is shown in Figure 1.1. The training examples (or training patterns) are the known instances from which we wish to learn a model that can generalize to previously unseen data.

Given infinite training data, consistent classifiers approximate the Bayesian decision

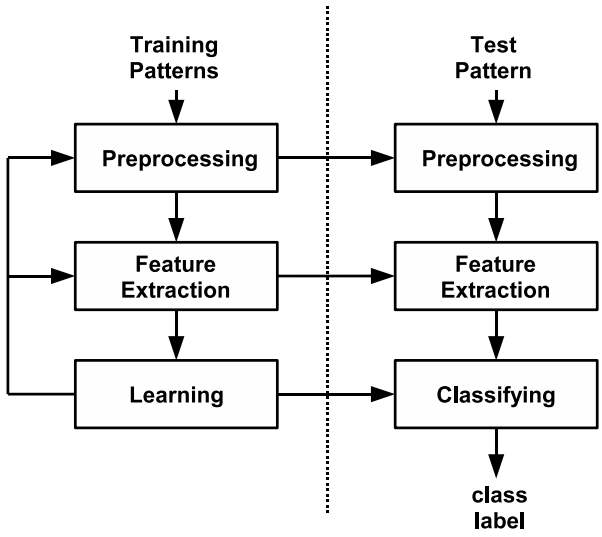

Figure 1.1: The general model of a pattern recognition system.

boundaries to arbitrary precision, therefore providing a similar generalization. However, often only a limited portion of the pattern space is available or observable. Given a finite and noisy data set, different classifiers typically provide different generalizations. It is thus necessary to train several classifiers when dealing with classification problems so as to ensure that a good model or parameter set is found.

1.2 Multiple-Classifier Systems

Given a set of independent inducers, the simplest way of building a classifier system is to select one with the best behaviour on a given testing database. During the classification just the output of the selected classifier is computed, and only this will affect the resulting decision. This selection is an "*early*" combination scheme widely used in Pattern Recognition.

However, selecting such a classifier is not necessarily the ideal choice since potentially valuable information may be wasted by discarding the results of the other classifiers. In order to avoid this kind of loss of information, the output of each available classifier should be examined for making the final decision.

1.2.1 Combination Architectures

To integrate the output of several classifiers into a final decision, various combination architectures are available. These architectures fall into the following three main groups:

- **Parallel:** Each of the inducers are invoked independently, and their results are then combined by a combiner. The majority of combination architectures in the literature belong to this category.
- **Cascading:** Individual classifiers are invoked in a linear sequence. The number of possible classes for a given pattern is gradually reduced as more classifiers

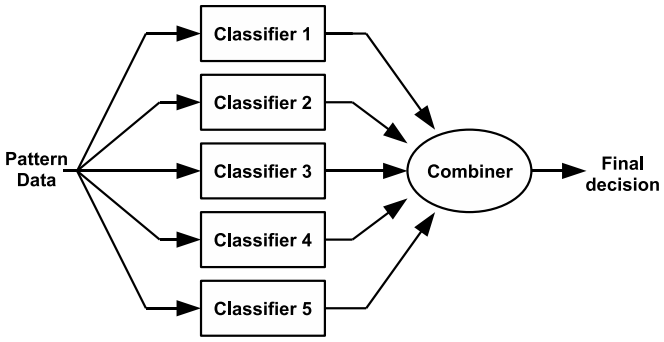

Figure 1.2: Parallel combination scheme

in the sequence are invoked. For the sake of efficiency, inaccurate but cheap classifiers are applied first, followed by more accurate and expensive inducers.

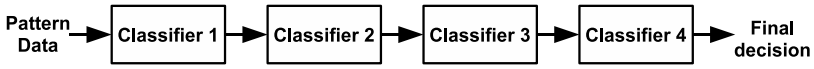

Figure 1.3: Cascading combination scheme

Cascading architectures, for instance, are commonly used in POS tagging [47] and NP parsing [102] applications.

- **Hierarchical:** Individual classifiers are combined into a structure similar to that of a decision tree classifier. The tree nodes, however, may now be associated with complex classifiers requiring a large number of features. The advantage of this architecture is its high efficiency and flexibility in exploiting the discriminant power of different types of features.

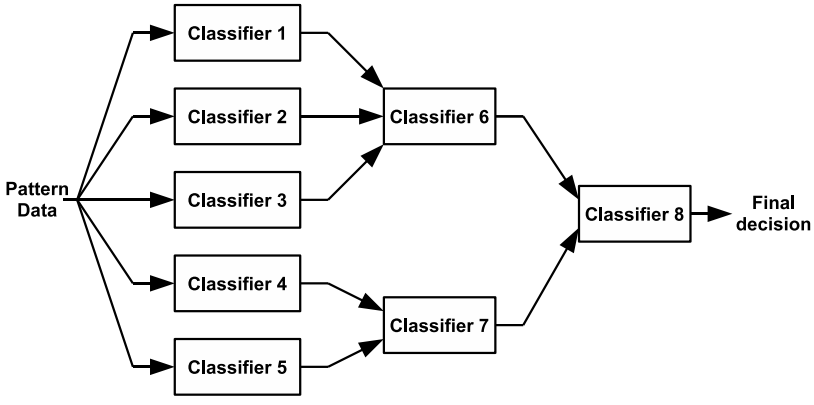

Figure 1.4: Hierarchical combination scheme

A number of classification schemes like Stacking [122] and dynamic selection use this type of architecture, and have became popular in Human Language Technology applications [110].

1.2.2 Types of Knowledge Sources

The goal of designing a combination scheme is to assign a class label for an input pattern using the information coming from the individual classifier instances. Each of the trained inducers has to be capable of providing the label that it prefers, but a number of machine learning methods can supply much more information than this. The combination strategies can incorporate the following types of classifiers based on the kind of information they provide:

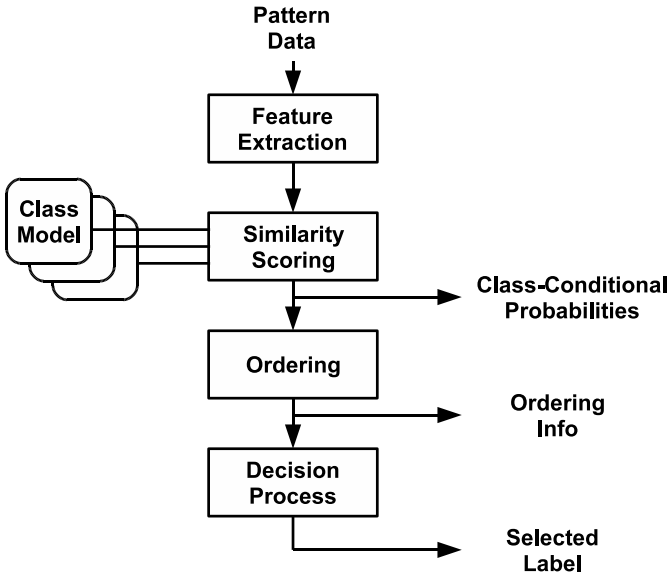

Figure 1.5: Information levels of a typical classifier

- **Abstract:** Just the assigned class label is provided. Combiners which only need this information as input are, for instance, the voting combiners like Bagging and Boosting.
- **Ranking:** Instead of merely providing the best class label associated with the given pattern, the list of labels is supplied, ranked in order of probability. This more general information type can be used as input for combiners like the Borda count rule.
- **Measurement or Confidence:** In the most general case, each of the a posteriori probabilities are provided. Combiners can aggregate these probabilities from different inducers and make a final decision. Examples of combiners which use the measurement information type are Prod, Sum, and Max Rules.

1.2.3 Generating effective Classifier Sets

A classifier combination is especially useful if the classifiers applied are largely independent. If this is not already guaranteed by the use of different learning sets or different learning methods, various re-sampling techniques like rotation and bootstrapping may be used to artificially create such differences.

- **Rotation:** The original learning set is divided into n disjoint subsets and uses different unions of $n-1$ subsets as training sets. This technique is commonly used in cross validation during error estimation.

- **Bootstrapping:** A bootstrap sample can be generated by sampling instances from the training set with replacement, using a specified probability distribution. Examples include Bagging and Boosting.

- **Generating noise:** The generated classification model of an unstable classifier strongly depends on existing errors in the training database. Adding artificial errors is a way of generating a set of more or less independent classifiers, making it possible to fulfil the requirements of a combiner.

1.3 Static Combination Schemes

The simplest non-trainable combiner is probably the most commonly used in the multiple classifier system community. Majority voting simply returns the class with the highest number of votes. There has been a huge amount of research on the theoretical aspects of majority voting for several decades, and, despite its simplicity it has proved to be quite efficient in most applications. Majority voting only takes into account the labels output by each individual classifier, but the natural way of making use of more information is to apply posterior probabilities for taking into account a confidence measure for each classifier. In the following we will concentrate on the combinations of classifiers that provide confidence level information.

Consider a pattern recognition problem where the pattern $\mathbf{x}$ is to be assigned to one of m possible classes $(\omega_1, \ldots, \omega_m)$. Let us assume that we have R classifiers, each representing the given pattern by a different feature vector. Next, denote this feature vector (employed by the ith classifier) by $\mathbf{x}^{(i)}$. In the feature space each class ω_k is modelled by the probability density function $p(\mathbf{x^{(i)}}|\omega_\mathbf{k})$ and its a priori probability of occurrence $P(\omega_k)$.

According to Bayesian theory, for given features $\mathbf{x}^i$, $i \in \{1, \ldots, R\}$ the pattern $\mathbf{x}$ should be assigned to class ω_j with the maximal value of the a posteriori probability such that

$$f(\mathbf{x}) = \omega_j, \quad j = \underset{k}{\operatorname{argmax}}\, P(\omega_k|\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(R)}). \tag{1.1}$$

Let us rewrite the a posteriori probability using Bayes' Theorem. Then We have

$$P(\omega_k|\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(R)}) = \frac{p(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(R)}|\omega_k)P(\omega_k)}{p(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(R)})}. \tag{1.2}$$

In the latter the unconditional joint probability density can be expressed in terms of the

conditional feature distributions, so that

$$p(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(R)}) = \sum_{j=0}^{m} p(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(R)}|\omega_j)P(\omega_j). \tag{1.3}$$

1.3.1 Product Rule

Let us assume that the probability distributions $p(\mathbf{x^{(1)}}, \ldots, \mathbf{x^{(1)}}|\omega_k)$ are conditionally statistically independent. Then

$$p(\mathbf{x^{(1)}}, \ldots, \mathbf{x^{(R)}}|\omega_k) = \prod_{i=0}^{R} p(\mathbf{x^{(i)}}|\omega_k), \tag{1.4}$$

and the decision rule

$$f(\mathbf{x}) = \omega_j, \quad j = \underset{k}{\operatorname{argmax}}\, P(\omega_k) \prod_i p(\mathbf{x^{(i)}}|\omega_k), \tag{1.5}$$

or, in terms of the a posteriori probabilities generated by the respective classifiers,

$$\underset{k}{\operatorname{argmax}}\, P^{1-R}(\omega_k) \prod_i p(\omega_k|\mathbf{x}^{(i)}). \tag{1.6}$$

1.3.2 Sum Rule

In some applications it may be appropriate to assume that the a posteriori probabilities computed by the respective classifiers will not dramatically deviate from those of the prior probabilities. This is a rather strong assumption but it may be readily satisfied when the available information is highly ambiguous due to the high level of noise. In such a situation we may assume that the a posteriori probability can be expressed in the form

$$P(\omega_k|\mathbf{x}^{(i)}) = P(\omega_k)(1 + \delta_{ki}), \tag{1.7}$$

where $\delta_{ki} \ll 1$. Substituting this for the a posteriori probabilities in (1.6), we find that

$$P^{1-R}(\omega_k) \prod_i P(\omega_k|\mathbf{x}^{(i)}) = P(\omega_k) \prod_i (1 + \delta_{ki}). \tag{1.8}$$

If we neglect terms of second and higher order, we can approximate the right-hand side and obtain the sum decision rule

$$f(\mathbf{x}) = \omega_j, \quad j = \underset{k}{\operatorname{argmax}} \left[(1 + R)P(\omega_k) + \sum_i p(\omega_k|\mathbf{x}^{(i)}) \right]. \tag{1.9}$$

1.3.3 Max, Min Rule

The decision rules (1.6) and (1.9) constitute the basic schemes for combining classifiers. Many commonly used combination strategies can be developed from these rules after noting that

$$\prod_i P(\omega_k|\mathbf{x}^{(i)}) \leq \min_i P(\omega_k|\mathbf{x}^{(i)}) \leq \sum_i P(\omega_k|\mathbf{x}^{(i)}) \leq \max_i P(\omega_k|\mathbf{x}^{(i)}). \tag{1.10}$$

This inequality suggests that the product and sum combination rules may be approximated by the max and min operators, where appropriate. These approximations lead to the following:

Max Rule:

$$f(\mathbf{x}) = \omega_j, \quad j = \underset{k}{\operatorname{argmax}} \left[(1+R)P(\omega_k) + R \max_i p(\omega_k|\mathbf{x}^{(i)}) \right], \tag{1.11}$$

Min Rule:

$$f(\mathbf{x}) = \omega_j, \quad j = \underset{k}{\operatorname{argmax}} \left[P^{1-R}(\omega_k) + R \max_i p(\omega_k|\mathbf{x}^{(i)}) \right]. \tag{1.12}$$

1.3.4 Median Rule

Note that using the *equal prior* assumption, the sum rule can be interpreted as computing the average a posteriori probability. It is well known that a robust estimate of the mean is the median, so it might be more appropriate to use it as the basis for the combining procedure. Adopting this leads to the following rule:

$$f(\mathbf{x}) = \omega_j, \quad j = \underset{k}{\operatorname{argmax}} \operatorname*{med}_i p(\omega_k|\mathbf{x}^{(i)}). \tag{1.13}$$

1.3.5 Voting Rule

Hardening a posteriori probabilities $P(\omega_k|\mathbf{x}^{(i)})$ will produce binary valued functions Δ_{ki} like

$$\Delta_{ki} = \begin{cases} 1 & if \ P(\omega_k|\mathbf{x}^{(i)}) = \max_j P(\omega_j|\mathbf{x}^{(i)}) \\ 0 & otherwise, \end{cases} \tag{1.14}$$

which results in combination decision outcomes rather than a combination of a posterori probabilities. Assuming that each a priori probability is equal, this leads to the following decision rule:

$$f(\mathbf{x}) = \omega_j, \quad j = \underset{k}{\operatorname{argmax}} \sum_i \Delta_{ki}. \tag{1.15}$$

Note that for each class ω_k, the sum on the right hand side of (1.15) simply counts the votes received for this hypothesis from each individual classifier.

1.3.6 Borda count

Instead of hardening a posteriori probabilities, it is possible to use modified probabilities ρ_{ki} based on ranking information.

$$\rho_{ki} = \frac{1}{C} \sum_{j:P(\omega_j|\mathbf{x}^{(i)}) \leq P(\omega_k|\mathbf{x}^{(i)})} 1 \quad , \tag{1.16}$$

where C is a normalization constant. This results in the following decision rule:

$$f(\mathbf{x}) = \omega_j, \quad j = \underset{k}{\operatorname{argmax}} \sum_i \rho_{ki}. \tag{1.17}$$

1.4 Summary

In this chapter we summarized the basic issues of multiple classifier systems, and derived some of the static combination rules applied in Human Language Technology like the "Majority Voting" and the "Product" rule. Although these techniques have become popular in multiple-classifier systems owing to their simplicity, they cannot be adapted to the special aspects of particular applications. For this reason, adaptive methods like additive combination schemes have become the focus of research. These techniques and their typical applications will be discussed in the next chapter.

2

The Additive Combination Model

In this chapter we will focus on linear combination schemes like *averaging* and boosting techniques. These techniques can be adapted to the context of a particular application, and they have a theoretical basis. Also, experimental studies have shown that linear classifier combinations can definitely improve the recognition accuracy. Tumer and Ghosh showed that combining networks using *single averaging* reduces the variance of the actual decision boundaries close to the optimum boundary [108]. Later Fumera and Roli extended the theoretical framework for the *weighted averaging* rule [92]. However, due to their highly restrictive assumptions, we cannot say anything with confidence about the performance of the combiner.

This chapter is organized as follows. In the first section we give a brief overview of linear combination schemes, including *averaging* techniques, and examine the theoretical background of these systems. In the next section we offer some strategies for setting the required coefficients. After, we describe the most commonly used algorithms for building effective ensemble systems like "Bagging" and "Boosting".

2.1 Linear Combinations

Combiners aggregate the outputs of different classifiers to make a final decision. This aggregation depends on the kind of information that the individual classifiers can supply. As discussed in the previous chapter the classification methods can be grouped into three main categories:

1. *Measurement or Confidence.* The classifier can model probability values for each class label. Let $f_i^j(\mathbf{x})$ denote the output of classifier $\mathcal{C}_j$ for class i and pattern $\mathbf{x}$.

The linear combination method can be described by the formula

$$\hat{f}_i(\mathbf{x}) = \sum_{j=1}^{N} w_j f_i^j(\mathbf{x}), \tag{2.1}$$

where $\hat{f}_i(\mathbf{x})$ is the combined confidence value, and w_j is the weighting factor of classifier $\mathcal{C}_j$. The final decision can be obtained by selecting the class with the greatest probability, in accordance with the Bayesian decision principle.

2. *Ranking*. The classifier can produce a list of class labels in the order of their probabilities. The combined position $\hat{g}_i$ is then computed via the formula

$$\hat{g}_i(\mathbf{x}) = \sum_{j=1}^{N} w_j g_i^j(\mathbf{x}), \tag{2.2}$$

where $g_i^j(\mathbf{x})$ is the position of the label ω_i in the classification result of pattern $\mathbf{x}$ obtained by the classifier $\mathcal{C}_j$. With the selection of a proper monotonic decreasing function u, and

$$f_i^j(\mathbf{x}) = u(g_i^j(\mathbf{x})), \tag{2.3}$$

the ranking-type combination can be reduced to the confidence-type scheme.

3. *Abstract.* The classifier supplies only the most probable class label. In this case the combination relies on the *majority voting* formula

$$\hat{l}(\mathbf{x}) = \arg\max_i \sum_{l_j(x)=i} w_j, \tag{2.4}$$

where $l_j(\mathbf{x})$ is the index of the class label calculated for pattern $\mathbf{x}$. Defining the choice of $f_i^j(\mathbf{x})$ as

$$f_i^j(\mathbf{x}) = \begin{cases} 1 & l_j(\mathbf{x}) = i \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

leads to the reformulation of voting as a confidence-type combination much like that for the ranking type.

As we showed earlier, the output of classifiers belonging to the *Ranking* or *Abstract* class can be transformed to class conditional probability values. Therefore, in the following section we shall deal only with the confidence-type combination, and expect the combiners to supply the class conditional probabilities.

2.2 Theoretical background

As mentioned previously, the output of the classifiers are expected to approximate the corresponding a posteriori probabilities if they are reasonably well trained. Thus

the decision boundaries obtained using this kind of classifier are close to Bayesian decision boundaries. Tumer and Ghosh developed a theoretical framework for analyzing the *averaging rule* of linear combinations [108]. Later Roli and Fumera extended this concept by examining the *weighted averaging* rule [92].

We shall focus on the classification performance near these decision boundaries. Consider the boundary between class i_1 and i_2. The output of the classifier is

$$f_i(\mathbf{x}) = p_i(\mathbf{x}) + \epsilon_i(\mathbf{x}), \tag{2.6}$$

where $p_i(\mathbf{x})$ is the real a posteriori probability of class i, and $\epsilon_i(\mathbf{x})$ is the estimation error. Let us assume that the class boundary $\mathbf{x}_b$ obtained from the approximated a posteriori probabilities

$$f_{i_1}(\mathbf{x}_b) = f_{i_2}(\mathbf{x}_b) \tag{2.7}$$

are close to the Bayesian boundaries $\mathbf{x}^*, i.e.$

$$p_{i_1}(\mathbf{x}^*) = p_{i_2}(\mathbf{x}^*), \tag{2.8}$$

for a boundary between class i_1 and i_2. Additionally assuming that the estimated errors $\epsilon_i(x)$ on different classes are independently and identically distributed (i.i.d.) variables with zero mean, Tumer and Ghosh showed that the expected error of classification can be expressed as:

$$E = E_{Bayes} + E_{add}, \tag{2.9}$$

where E_{Bayes} is the error of a classifier with the correct Bayesian boundary. The *added error* E_{add} can be expressed as:

$$E_{add} = \frac{s\sigma_b^2}{2}, \tag{2.10}$$

where σ_b^2 is the variance of b,

$$b = \frac{\epsilon_{i_1}(\mathbf{x}_b) - \epsilon_{i_2}(\mathbf{x}_b)}{s}, \tag{2.11}$$

and s is a constant term depending on the derivatives of the probability density functions at the optimal decision boundary.

Let us consider the effect of combining multiple classifiers. We shall deal only with linear combinations, so we have the following combined probabilities:

$$\hat{f}_i(\mathbf{x}) = \sum_{j=1}^{N} w_j f_i^j(\mathbf{x}), \tag{2.12}$$

where f_i^j denotes the output of the classifier $\mathcal{C}_j$ for the class i. Assuming normalized

weights, i.e.

$$\sum_{j=1}^{N} w_j = 1, \quad w_j \geq 0 \tag{2.13}$$

we have that

$$\hat{f}_i(\mathbf{x}) = p_i(\mathbf{x}) + \hat{\epsilon}_i(\mathbf{x}), \tag{2.14}$$

where

$$\hat{\epsilon}_i(\mathbf{x}) = \sum_{j=1}^{N} w_j \epsilon_i^j(\mathbf{x}) \tag{2.15}$$

Let us compute the variance of $\hat{b}$, where

$$\hat{b} = \frac{\hat{\epsilon}_{i_1} - \hat{\epsilon}_{i_2}}{s}. \tag{2.16}$$

Assuming that $\epsilon_i^j(\mathbf{x})$ are i.i.d. variables with zero mean and variance $\sigma_{\epsilon^j}^2$, the errors of different classifiers on the same class are correlated, while on different classes they are uncorrelated.

$$Cov(\epsilon_{i_1}^m(\mathbf{x}_b), \epsilon_{i_2}^n(\mathbf{x}_b)) = \begin{cases} \rho_{mn} & \text{when } i_1 = i_2 \\ 0 & \text{otherwise} \end{cases},$$

where ρ_{mn} denotes the covariance between the errors of classifier $\mathcal{C}_m$ and $\mathcal{C}_n$ for each class. Expanding the tag σ_ϵ^2 in Eq. (2.10), we find that

$$\begin{aligned} \hat{E}_{add} &= \frac{1}{s} \sum_{j=1}^{N} w_j^2 \sigma_{\epsilon^j}^2 \\ &+ \frac{1}{s} \sum_{m \neq n} w_m w_n \rho_{mn}. \end{aligned}$$

Expressed in terms of additional errors of the classifiers

$$\begin{aligned} \hat{E}_{add} &= \frac{1}{s} \sum_{j=1}^{N} w_j^2 E_{add}^j \\ &+ \frac{1}{s} \sum_{m \neq n} w_m w_n \rho_{mn}, \end{aligned}$$

where the term E_{add}^j denotes the added error of the classifier $\mathcal{C}_j$. In the case of uncorrelated estimation errors (i.e. when $\rho_{mn} = 0$), this equation reduces to

$$\hat{E}_{add} = \sum_{j=1}^{N} w_j^2 E_{add}^j \ , \tag{2.17}$$

which leads to the following optimal values for w:

$$w_j = c\frac{1}{E^j_{add}}, \tag{2.18}$$

where c is a normalization factor. With equally performing classifiers, that is when all the errors E^j_{add} have the same value

$$E^j_{add} = E_{add}, \quad j = 1, \ldots, N, \tag{2.19}$$

we obtain the *Simple Averaging* rule:

$$w_j = \frac{1}{N}, \tag{2.20}$$

which results in the following error value:

$$\hat{E}_{add} = \frac{1}{N}E_{add}. \tag{2.21}$$

This formula shows that, under the conditions mentioned, linear combinations reduce the error of the individual classifier. Taking into account correlated errors, however, does not lead to a simple general expression for optimal values of weights, and other methods are required to estimate the optimal parameters.

2.3 Simple Adaptive Combinations

To achieve the best combination performance the parameters of the combiner can be trained on a selected training data set. The form of linear combinations we deal with is quite simple, the trainable parameters being just the weights of classifiers. Thus the various linear combinations differ only in the values of these factors. In the following we give some examples of methods commonly employed, and in the next section we propose a new method for computing these parameters.

1. *Simple Averaging*. In this simplest case, the weights can be selected so they have the same value:
$$w_j = \frac{1}{N}. \tag{2.22}$$
As mentioned before, this selection is optimal when all the classifiers have a very similar performance, and all the assumptions mentioned in Section 2.2 hold.

2. *Weighted Averaging*. Experiments show [92] that *Simple Averaging* can be outperformed when selecting weights to be inversely proportional to the error rate of the corresponding classifier:
$$w_j = \frac{1}{E_j}, \tag{2.23}$$

where E_j is the error rate of the classifier $\mathcal{C}_j$, i.e. the ratio of the number of correctly classified patterns and total number of patterns on a selected data set for training the combiner. This rule (a more general version of the *Simple Averaging* rule) is employed in order to handle the combination of unequally performing classifiers.

3. *Hierarchical methods.* To calculate the weights w_j one can take those values that minimize some kind of distance between the computed and required conditional scores:

$$\min_{w} \sum_{x \in \mathcal{X}} \sum_{i=1}^{l} \mathcal{L}(\hat{f}_i(\mathbf{x}), r_i)), \tag{2.24}$$

where r_i is the requested conditional scores for class i, and $\mathcal{L}(\hat{f}_i(\mathbf{x}), r_i))$ is the loss function. Since machine learning algorithms can be regarded as optimization methods that minimize the expected value of a loss function on the training database, this optimization can be done by applying an appropriate machine learning algorithm.

2.4 Bagging

The Bagging (Bootstrap aggregating) algorithm[16] votes classifiers generated by different bootstrap samples (replicates). A bootstrap sample is generated by uniformly sampling m instances from the training set with replacement. T bootstrap samples $B_1, B_2, ..., B_T$ are generated and a classifier $\mathcal{C}_i$ is built from each bootstrap sample B_i. A final classifier C^* is built from $\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_T$ whose output is the class predicted most often by its sub-classifiers (majority voting).

Algorithm 1 Bagging algorithm

Require: Training Set S, Inducer $\mathcal{I}$
Ensure: Combined classifier $\mathcal{C}^*$
  for $i = 1 \dots T$ **do**
    $S' =$ bootstrap sample from S
    $\mathcal{C}_i = \mathcal{I}(\mathcal{S}')$
  end for
  $C^*(\mathbf{x}) = \underset{j}{\operatorname{argmax}} \sum_{i:\mathcal{C}_i(\mathbf{x})=\omega_j} 1$

For a given bootstrap sample, an instance in the training set will have a probability $1-(1-1/m)^m$ of being selected at least once from the m instances randomly selected from the training set. For large m, this is about 1-1/e = 63.2%. This perturbation causes different classifiers to be built if the inducer is unstable (e.g. ANNs, decision trees) and the performance may improve if the induced classifiers are uncorrelated. However, Bagging can slightly degrade the performance of stable algorithms (e.g. kNN) since effectively smaller training sets are used for training.

2.5 Random Forests

Breiman proposed Random Forests [15] as a variant of Bagging. A random forest is an ensemble creation method that uses tree classifiers like C4.5 algorithm as a base classifier. An ensemble of decision trees is built by generating independent identically distributed random vectors $\Theta_j, j = 1, ..., K$. One tree is grown from each vector. The difference with Bagging is that the vectors Θ_j can be built by sampling from feature sets, a sample set or by varying some parameters of the tree (e.g. the number of nodes). In /citeRodriguez, Rodriguez et al. proposed a variation of random forest called Rotation forests that simply adds a PCA pre-processing in order to decorrelate the training data. They showed that experimental improvements could be obtained on many datasets.

2.6 Boosting

The underlying idea of boosting is to combine simple rules iteratively to form an ensemble that will improve the performances of each single member. Boosting theory has its roots in Probably Approximately Correct (PAC) learning [112]. PAC gives a nice formalism for deciding how much training data we need in order to achieve a given probability of correct predictions on a given fraction of future test data. In [112], Valiant showed that simple rules, each performing only slightly better than random guessing, can be combined to form an arbitrarily good ensemble. The challenge of boosting is to find a PAC algorithm with arbitrarily high accuracy.

The first boosting algorithms were proposed by Schapire in 1990 [94] and Freund in 1995 [30]. However, several strong assumptions prevented the efficiently use of these algorithms in practical situations: They need prior knowledge of the accuracy of the weak hypotheses.

2.6.1 Adaboost

A first step towards more practical Boosting is the AdaBoost (Adaptive Boosting) algorithm [31]. Adaboost is adaptive in the sense that a new hypothesis is selected given the performances of the previous iterations. Unlike bagging, this allows the algorithm to focus on the hard examples. This adaptive strategy is managed by a weight distribution D over the training samples. A weight $D(i)$ is given to each training pattern $\mathbf{x}_i$. Examples with large weights will have more impact on choosing the weak hypothesis than those with low weights. Then after each round, the weight distribution is updated in such a way that the weight of misclassified examples is increased.

In the following we shall deal with only the case of binary classification, i.e. only two class labels $\{-1, +1\}$ are available. Let us consider, as usual, a training set $Z_N = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)$. We suppose that we have a base learning algorithm (or weak learner) which accepts as input a sequence of training samples Z_n along with a distribution D over the training samples. Given such an input, the weak learner

constructs a weak hypothesis $\mathcal{C}$. The predicted label y is given by $sign(\mathcal{C}(\mathbf{x}))$, while the confidence of the prediction is given by $\|\mathcal{C}(\mathbf{x})\|$. We shall also assume that the corresponding weighted training error is smaller than $1/2$. This means that the weak hypotheses have to be at least slightly better than random guessing with respect to the distribution D. The distribution D is first initialized uniformly over the training samples. Then it is iteratively updated in such a way that the likelihood of the objects beeing misclassified in the previous iteration is increased.

The loss function $\mathcal{L}$ used for updating the weights (step 4 in the main loop) is usually an exponential loss function, but other loss functions have been also proposed in the literature (Logitboost [32] or Arcing [14]).

Two critical choices in AdaBoost are how to choose the weak hypothesis $\mathcal{C}_t$ and what is the optimal w_t. Let us define the training error of the ensemble $\hat{\mathcal{C}}$:

$$\mathcal{L}_{1/0}(\hat{\mathcal{C}}) = \frac{1}{N}\sum_{i=1}^{n}\mathcal{L}_{1/0}(y_i, \mathcal{C}(\mathbf{x}_i)), \tag{2.25}$$

where $\mathcal{L}_{1/0}$ is the $1/0$ loss function

$$\mathcal{L}_{1/0}(y_i, \mathcal{C}(\mathbf{x}_i)) = \begin{cases} 1 & \text{if } y_i = \mathcal{C}(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases} \tag{2.26}$$

Shapire and Singer [95] give a bound on the training error of the combined hypothesis $\hat{\mathcal{C}}$:

$$\mathcal{L}_{1/0}(\hat{\mathcal{C}}) \leq \prod_{t=1}^{T} N_t, \tag{2.27}$$

where $N_t = \sum_i D_i^t \exp(-\omega_t y_i \mathcal{C}_t(\mathbf{x}_i))$.

According to this theorem, the training error can be minimized by greedily minimizing N_t on each round of boosting. To choose the optimal ω_t, we will consider several cases in the following sections.

2.6.2 Discrete Adaboost

Let us first consider the original version of AdaBoost, called Discrete AdaBoost, when the range of each weak hypothesis is restricted to the labels $\{-1, +1\}$. Then the optimal ω_t can be found by approximating N_t as follows:

$$N_t = \sum_i D_i^t \exp(-\omega_t y_i \mathcal{C}_t(\mathbf{x}_i)) \tag{2.28}$$

$$\leq \sum_i D_i^t \left(\frac{1 + y_i \mathcal{C}_t(\mathbf{x}_i)}{2} \exp(-\omega_t) + \frac{1 - y_i \mathcal{C}_t(\mathbf{x}_i)}{2} \exp(\omega_t) \right). \tag{2.29}$$

Algorithm 2 Boosting algorithm

Require: Training Set $Z_N = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)$, number of iterations T, Inducer $\mathcal{I}$

Ensure: Combined classifier C^*

$D_n^{(1)} = 1/N$ for all $n = 1, \ldots, N$.

for $t = 1 \ldots T$ **do**

$S' =$ bootstrap sample from Z_N with distribution D_n^t

$\mathcal{C}_t = \mathcal{I}(\mathcal{S}')$

$\epsilon_t = \frac{1}{n} \sum_{n=1}^{N} \mathcal{L}_{1/0}(y_n, \mathcal{C}(\mathbf{x}_n))$

if $\epsilon_t > 1/2$ **then** Exit

select optimal w_t

update weights $D_n^{(t+1)} = \frac{D_n^{(t)} \mathcal{L}(y_n, \mathcal{C}(\mathbf{x}_n))}{N_t}$,

where N_t is a normalization factor such that $\sum_{n=1}^{N} D_n^{(t+1)} = 1$.

end for

$\mathcal{C}^*(\mathbf{x}) = sign(\sum_{t=1}^{T} w_t \mathcal{C}_t(\mathbf{x}))$

The coefficient w_t that minimizes N_t can be found analytically:

$$w_t = \frac{1}{2} \log \left(\frac{1 + r_t}{1 - r_t} \right), \tag{2.30}$$

where $r_t = \sum_i D_i^t y_i \mathcal{C}_t(\mathbf{x}_i)$. With this choice of coefficient, the training error of the combined hypothesis $\hat{\mathcal{C}}$ is bounded by:

$$\mathcal{L}_{1/0}(\hat{\mathcal{C}}) \leq \prod_{t=1}^{T} \sqrt{1 - r_t^2}. \tag{2.31}$$

The quantity r_t is a natural measure of the correlation of the predictions of $\mathcal{C}_t$ and the labels y_i with respect to D_t. From equation 2.31 it follows that:

$$\mathcal{L}_{1/0}(\hat{\mathcal{C}}) \leq \exp \left(- \sum_{t=1}^{T} r_t^2 \right), \tag{2.32}$$

from which we infer that the condition $\sum_{t=1}^{T} r_t^2 \to \infty$ suffices to guarantee that the training error converges towards 0 when the number of iteration increases. Clearly this holds if $r_t \geq r_0 > 0$ for some positive constant r_0 (recall that the weighting training error of a weak hypothesis is less than 0.5).

2.6.3 Real Adaboost

A more general formulation of Discrete AdaBoost uses the confidence levels of each weak classifier instead of just binary outputs. It is called Real AdaBoost [95]. Unlike Discrete AdaBoost, the output space of the weak classifiers is not restricted to $\{-1; 1\}$, but can take values in R. More specifically, let us consider a partition of R into disjoint

blocks $X_1, X_2, ..., X_n$ for which $h(x') = h(\mathbf{x}) = c_j$ for all $(\mathbf{x}, \mathbf{x}') \in X_j \times X_j$. Let us further assume that the partitioning is given. The task is to find the optimal c_j , $j = 1, ..., N$. Let

$$W_j^+ = \sum_{i:\mathbf{x}_i \in X_j, y_i = +1} D_i \tag{2.33}$$

be the weighted fraction of positive samples falling into partition X_j, and

$$W_j^- = \sum_{i:\mathbf{x}_i \in X_j, y_i = -1} D_i \tag{2.34}$$

the equivalent for the negative class. Then the normalization factor N_t can be rewritten as:

$$N_t = \sum_j (W_j^+ \exp(-c_j) + W_j^- \exp(c_j)). \tag{2.35}$$

The expression that we want to minimize has the optimal c_j values

$$c_j = \frac{1}{2} \log \left(\frac{W_j^+}{W_j^-} \right) \tag{2.36}$$

This technique has proved to be very effective in many applications, outperforming Discrete AdaBoost by taking into account the confidence measures of the weak hypotheses. A good example of a weak learner that can be used using this partitioning technique is a simple decision tree. The leaves of the tree directly define the partition of the domain.

Note that in practice, W_j^+ or W_j^- is very small, which could produce an inconsistency in Equation 2.36. That is why we generally use a smoothed version of the coefficients:

$$c_j = \frac{1}{2} \log \left(\frac{W_j^+ + \epsilon}{W_j^- + \epsilon} \right), \tag{2.37}$$

where $\epsilon > 0$ is an appropriately small positive-valued constant.

2.6.4 Generalization error

So far have we only considered training error convergence to demonstrate the efficiency of AdaBoost, but minimizing the empirical risk is no guarantee of a good generalization. However, in practical situations, AdaBoost seems very unlikely to overfit and has one more interesting property: the test error continues decreasing with the number of iterations, even if the training error has reached zero. In order to explain this phenomenon, let us analyze AdaBoost from a margin perspective. Let us recall that the ensemble hypothesis has the form:

Without loss of generality, we can assume that $\sum_{j=1}^{N} \omega_j = 1$. Let us call $\mathcal{H}$ the space of the hypotheses h_t. Similar to margins in Support Vector Machines, we can define the margin of example $\mathbf{x}$ with label y to be $\rho = yf(\mathbf{x})$. The value of the margin

can be viewed as a confidence in the prediction of H. Shapire proved [95] that the generalization error is upper bounded with probability $1-\delta$ for all $\theta > 0$ and for all f by

$$P_{Z_n}(yf(\mathbf{x}) \leq \theta) + \mathcal{O}\left(\frac{1}{\sqrt{n}}\left(\frac{d\log^2(n/d)}{\theta^2} + \log(\frac{1}{\delta})\right)^{\frac{1}{2}}\right), \tag{2.38}$$

where P_{Z_N} denotes the probability with respect to choosing an example $(\mathbf{x}, y)$ uniformly from the training set Z_N and d can be regarded as the VC dimension of $\mathcal{H}$.

The bound stated in the equation does not depend on the number of classifiers in the ensemble, but remains quite loose in practical applications. However, it shows the tendency that larger margins lead to a better generalization. In fact, the generalization continues to improve even after the training error has reached zero because the margin is still increasing. Note that several studies have tried to interpret AdaBoost as a soft margin classifiers [74, 89]

2.7 Summary

In this chapter we outlined the main aspects of linear combination techniques, along with their theoretical background. We examined how to build simple adaptive combination strategies and described the popular methods called "Bagging" and "Boosting". These methods have proved effective in improving the overall classification performance, especially in the case of "weak" classifiers, but at a cost. To achieve high accuracies it may require hundreds or thousands of iterations, which limits their practical usefulness. In the next chapter we will introduce a simpler approach which works better than the simple additive models and is less CPU demanding than the sophisticated Boosting variants.

3

Combinations and the Analytic Hierarchy Process

Linear combination schemes, especially Boosting algorithms, are frequently used in machine learning applications due to their ability to improve the classification performance. The Adaboost algorithm and its variants create a sequence of classifier instances by training the same classifier algorithm on special bootstrap-samples of the training database. The classification performance of the original classifier method can be dramatically increased, especially in the cases of "weak" classifers, but for the final solution it requires hundreds or thousands of iterations. In the practice, however, most of the applications cannot provide the required huge amount of resources for applying such a great number of classifier instances.

In this chapter we will presents an effective combination solution for these kinds of applications. In the first section we will give a brief summary of the Multi-criteria Decision Making and the fundamentals of the Analytic Hierarch Process. In the second section we will investigate the mathematical properties of AHP, and a novel combination method will be proposed in the third section.

3.1 Analytic Hierarchy Process

Analytic Hierarchy Process allows decision makers to model a complex problem in a hierarchical structure which showing the relationships of the goal, objectives (criteria), sub-objectives, and alternatives (See Figure 3.1). Uncertainties and other influencing factors can also be included.

AHP allows for the application of data, experience, insight, and intuition in a logical and thorough way. It enables decision-makers to derive ratio scale priorities or weights as opposed to arbitrarily assigning them. In so doing, AHP not only supports decision-makers by enabling them to structure complexity and exercise judgment, but

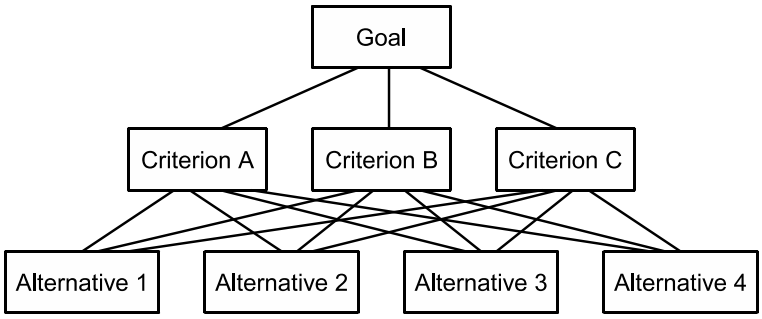

Figure 3.1: A simple AHP hierarchy. In practice, many criteria have one or more layers of subcriteria. These are not shown in this simplified diagram. Also, to avoid clutter in AHP diagrams, the lines between the alternatives and criteria are often omitted or reduced in number.

allows them to incorporate both objective and subjective considerations into the decision process. AHP is a compensatory decision methodology because alternatives that are deficient in one or more objectives can compensate by their performance in other objectives. It is composed of several previously existing but unrelated concepts and techniques such as hierarchical structuring of complexity, pairwise comparisons, redundant judgments, an eigenvector method for deriving weights, and consistency considerations. Although each of these concepts and techniques were useful in themselves, the synergistic combination of the concepts and techniques (along with some new developments) produced a process whose power is indeed far more than the sum of its parts.

Users of the AHP first decompose their decision problem into a hierarchy of more easily comprehended sub-problems, each of which can be analyzed independently. The elements of the hierarchy can relate to any aspect of the decision problem - tangible or intangible, carefully measured or roughly estimated, well- or poorly-understood-anything at all that applies to the decision at hand.

Once the hierarchy is built, the decision makers systematically evaluate its various elements, comparing them to one another in pairs. In making the comparisons, the decision makers can use concrete data about the elements, or they can use their judgments about the elements' relative meaning and importance. It is the essence of the AHP that human judgments, and not just the underlying information, can be used in performing the evaluations.

The AHP converts these evaluations to numerical values that can be processed and compared over the entire range of the problem. A numerical weight or priority is derived for each element of the hierarchy, allowing diverse and often incommensurable elements to be compared to one another in a rational and consistent way. This capability distinguishes the AHP from other decision making techniques.

In the final step of the process, numerical priorities are derived for each of the decision alternatives. Since these numbers represent the alternatives' relative ability to achieve the decision goal, they allow a straightforward consideration of the various courses of action.

The procedure can be summarized as:

1. The alternatives and the significant attributes are identified.

2. For each attribute, and each pair of alternatives, the decision makers specify their preference (for example, whether the location of alternative "A" is preferred to that of "B") in the form of a fraction between 1/9 and 9.

3. Decision makers similarly indicate the relative significance of the attributes. For example, if the alternatives are comparing potential real-estate purchases, the investors might say they prefer location over price and price over timing.

4. Each matrix of preferences is evaluated by using eigenvalues to check the consistency of the responses. This produces a "consistency coefficient" where a value of 1 means all preferences are internally consistent. This value would be lower, however, if a decision maker said X is preferred to Y, Y to Z but Z is preferred to X (such a position is internally inconsistent). It is this step that causes many users to believe that AHP is theoretically well founded.

5. A score is calculated for each alternative.

The two basic steps in the process are to model the problem as a hierarchy, then to establish priorities for its elements.

3.2 Mathematical model

To compute the importance of possible choices, pairwise comparison matrices are utilized for each criterion. The element a_{ij} of the comparison matrix A represents the relative importance of choice i against the choice j, implying that the element a_{ji} is the reciprocal of a_{ij}. Let the importance value v of choice y be expressed as a linear combination of the importance values for each applied criterion:

$$v(y) = \sum_{j=1}^{n} w_j v(y_j), \tag{3.1}$$

where w_j is the importance of choice y with respect to the criterion y_j. Using comparison matrices AHP propagates the importance values of each node from the topmost criteria towards the alternatives, and selects the alternative with the greatest importance value as its final decision.

Let us now focus on the computation of the weights w for a selected criterion. The elements of a given pairwise comparison matrix approximate the relative importance of the choices, thus

$$a_{ij} \approx \frac{w_i}{w_j}, \tag{3.2}$$

where the elements of the unknown vector w are the importance values. A matrix M

is called consistent if its components satisfy the following equalities:

$$m_{ij} = \frac{1}{m_{ji}}, \tag{3.3}$$

and

$$m_{ij} = m_{ik}m_{kj} \quad \forall\, i, j, k. \tag{3.4}$$

If A is not consistent, it is not possible to find a vector w that satisfies the equation

$$a_{ij} = \frac{w_i}{w_j}. \tag{3.5}$$

Now let us define the matrix of weight ratios by

$$W = \begin{pmatrix} \frac{w_1}{w_1} & \frac{w_1}{w_2} & \frac{w_1}{w_3} & \cdots & \frac{w_1}{w_n} \\ \frac{w_2}{w_1} & \frac{w_2}{w_2} & \frac{w_2}{w_3} & \cdots & \frac{w_2}{w_n} \\ \frac{w_3}{w_1} & \frac{w_3}{w_2} & \frac{w_3}{w_3} & \cdots & \frac{w_3}{w_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{w_n}{w_1} & \frac{w_n}{w_2} & \frac{w_n}{w_3} & \cdots & \frac{w_n}{w_n} \end{pmatrix}, \tag{3.6}$$

or, in matrix notation,

$$W = ww^T. \tag{3.7}$$

Note that eqs. (3.3) and (3.4) hold for the matrix W:

$$w_{ij} = \frac{w_i}{w_j} = \frac{w_i}{w_k}\frac{w_k}{w_j} = w_{ik}w_{kj}, \tag{3.8}$$

hence the matrix of weight ratios is consistent.

Because the rows of matrix W are linearly dependent, the rank of the matrix is 1, and there is only one nonzero eigenvalue. Knowing that the trace of a matrix is invariant under similarity transformations, the sum of diagonal elements is equal to the sum of eigenvalues, which implies that the nonzero eigenvalue λ_{max} equals the number of the rows:

$$\lambda_{max} = n. \tag{3.9}$$

It is straightforward to verify that the vector w is an eigenvector of matrix W corresponding to the maximum eigenvalue

$$\begin{aligned} (Ww)_i &= \sum_{j=1}^{n} W_{ij}w_j \\ &= \sum_{i=1}^{n} \frac{w_i}{w_j}w_j = \sum_{j=1}^{n} w_i \\ &= nw_i. \end{aligned}$$

The aim of AHP is to resolve the weight vector w from a pairwise comparison matrix A, where the elements of A correspond to the measured or estimated weight ratios. Following Saaty we shall assume that

$$a_{ij} > 0, \tag{3.10}$$

and

$$a_{ij} = \frac{1}{a_{ji}}. \tag{3.11}$$

From matrix theory it is known that a small perturbation of the coefficients implies a small perturbation of the eigenvalues. Hence we still expect to find an eigenvalue close to n, and select the elements of the corresponding eigenvector as weights. It can be proved that

$$\lambda_{max} \geq n,$$

and the matrix A is consistent if and only if $\lambda_{max} = n$. A way of measuring the consistency of the matrix A is by defining the *consistency index* (CI) as the negative average of the remaining eigenvalues:

$$CI = \frac{\sum_{\lambda<\lambda_{max}} \lambda}{n-1} = \frac{\lambda_{max} - n}{n-1} \tag{3.12}$$

3.2.1 Combinations based on AHP

As mentioned above, AHP provides the following solution for the problem of linear MCDM systems:

$$v(y) = \sum_{i=1}^{n} w_i v(y_i),$$

where the importance value of the choice is the linear combination of the importance values of the direct criteria. In linear classifier combinations the combined class conditional probabilities are computed as weighted sums of the probability values from each classifier, so

$$f_i(x) = \sum_{j=1}^{N} w_j f_i^j(x).$$

Noting the similarities between these two methods, it is clear that, by applying pairwise comparisons on classifiers performance, AHP provides a way of computing the weights of inducers in classifier combinations. Let us calculate the element a_{ij} of the comparison matrix as the quotient of classification performance on a selected test data set:

$$a_{ij} = \frac{\frac{1}{E_i}}{\frac{1}{E_j}} = \frac{1}{a_{ji}}, \tag{3.13}$$

where E_i is the classification error of classifier $\mathcal{C}_i$. If all the performance errors are measured on the same test data set, the comparison matrix A is consistent, and the elements of the eigenvector whose corresponding eigenvalue is N, that is

$$w_i = \frac{1}{E_i}, \tag{3.14}$$

are the same those as generated by *weighted averaging*. However, this method allows us to make pairwise comparisons of different inducers applied on different (e.g. randomly generated) test sets, taking advantage of the stabilizing effect of AHP. This leads to a more robust classification performance, especially in noisy environments, as shown in the experiments section.

3.3 Experiments

In this section we will describe our experiments for comparing the performance of *averaging* combiners and our AHP-based combiner.

3.3.1 Evaluation Domain

In the experiments three data sets were employed: a data-set used in our speech recognition system, and two other datasets (letter and satimage) originating from the statlog/UCI repository. (http://www.liacc.up.pt/ML/statlog)

1. *Speech data set*. The database is based on recorded samples taken from 160 children aged between 6 and 8. The ratio of girls and boys was 50% - 50%.The speech signals were recorded and stored at a sampling rate of 22050 Hz in 16-bit quality. Each speaker uttered all the Hungarian vowels, one after the other, separated by a short pause. Since we decided not to discriminate their long and short versions, we only worked with 9 vowels altogether. The recordings were divided into a train and a test set in a ratio of 50% - 50%. There are numerous methods for obtaining representative feature vectors from speech data, but their common property is that they are all extracted from 20-30 ms chunks or 'frames' of the signal in 5-10 ms time steps. The simplest possible feature set consists of the so-called bark-scaled filterbank log-energies (FBLE). This means that the signal is decomposed with a special filterbank and the energies in these filters are used to parameterize speech on a frame-by-frame basis. In our tests the filters were approximated via Fourier analysis with a triangular weighting. Altogether 24 filters were necessary to cover the frequency range from 0 to 11025 Hz. Although the resulting log-energy values are usually sent through a cosine transformation to obtain the well-known mel-frequency cepstral coefficients (MFCC), we abandoned it because, as we observed earlier, the learner we work with is not sensitive to feature correlations so a cosine transformation would bring no significant improvement.

2. *Letter Data Set*. The objective of the Data set is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 to 15. We typically trained on the first 16000 items and then used the resulting model to predict the letter category for the remaining 4000.

3. *Satimage Data Set*. One frame of Landsat MSS imagery consists of four digital images of the same scene in different spectral bands. Two of these are in the visible region (corresponding approximately to green and red regions of the visible spectrum) and two are in the (near) infra-red. Each pixel is a 8-bit binary word, with 0 corresponding to black and 255 to white. The spatial resolution of a pixel is about 80m x 80m. Each image contains 2340 x 3380 such pixels. The database is a (tiny) sub-area of a scene, consisting of 82 x 100 pixels. Each line of data corresponds to a 3x3 square neighborhood of pixels completely contained within the 82x100 sub-area. Each line contains the pixel values in the four spectral bands (converted to ASCII) of each of the 9 pixels in the 3x3 neighborhood and a number indicating the classification label of the central pixel. The number of possible class labels for each pixel is 7. We trained on the first 4435 patterns of the database and selected the remaining 2000 patterns for testing.

3.3.2 Evaluation Method

During the experiments we compared the performance of 6 different combiners applied on each of the 3 databases. For each of the databases we trained 3-layered neural networks with different structures, and selected 5 subsets of classifiers, denoted by $Set1$ to $Set5$.

In the case of the *speech* and *letter* databases we trained networks, setting the number of neurons in the hidden layer to 5, 10, 20, 40, 80, 160, and 320. Table 3.1 shows the construction of classifier sets. The columns refer to the number of hidden neurons, and the rows show which networks belong to the selected classifier sets.

	5	10	20	40	80	160	320
Set1	x	x	x	x	x	x	x
Set2		x	x	x	x	x	x
Set3			x	x	x	x	x
Set4				x	x	x	x
Set5				x	x	x	

Table 3.1: Classifier sets for the *speech* and *letter* databases.

The *satimage* database contains only 7 classes. In this case we trained networks

with hidden layer size set to 2, 5, 10, 20, 40, 80, and 160. The corresponding classifier selection is displayed in Table 3.2.

	2	5	10	20	40	80	160
Set1	x	x	x	x	x	x	x
Set2		x	x	x	x	x	x
Set3			x	x	x	x	x
Set4				x	x	x	x
Set5				x	x	x	

Table 3.2: Classifier sets for the *satimage* database.

The experiments compared the performance of linear combination schemes with different methods for acquiring the proper weights. We examined 2 schemes of *Averaging* : SA and WA (i.e. simple and weighted) averaging, and 4 schemes of *AHP*. To calculate the pairwise comparison matrices needed for the AHP method, we took the quotient of classification errors of the two competing networks on a random test set generated by bootstrapping (resampling the training data set with replacement) of the training set. Based on the size of the generated test set we had 4 AHP schemes, $AHP1$ to $AHP4$, setting the size of each to 50, 100, 200 and 400, respectively. With the WA combiner, the original training set was chosen for the calculation of the weights.

3.3.3 Results and Discussion

	Set1	Set2	Set3	Set4	Set5
SA	**8.52**	**9.26**	9.95	9.77	8.80
WA	8.66	9.21	9.91	9.81	8.75
AHP_1	8.66	8.94	**9.44**	10.05	8.80
AHP_2	8.56	9.12	9.72	10.19	8.80
AHP_3	8.61	9.21	9.49	9.58	**8.70**
AHP_4	8.61	9.31	9.55	**9.07**	**8.70**

Table 3.3: Classification errors [%] on the Speech database (Error without combination: 12.92%)

	Set1	Set2	Set3	Set4	Set5
SA	8.70	8.34	7.88	7.26	7.78
WA	7.56	7.64	7.64	7.06	7.68
AHP_1	6.84	7.26	7.04	**6.76**	**7.48**
AHP_2	6.74	**6.90**	6.98	6.82	7.56
AHP_3	**6.67**	6.94	6.96	6.80	7.58
AHP_4	6.78	7.00	**6.88**	6.82	7.54

Table 3.4: Classification errors [%] on the Letter database (Error without combination: 13.78%)

	Set1	Set2	Set3	Set4	Set5
SA	10.95	10.35	10.35	10.00	10.05
WA	10.50	10.35	10.45	9.90	10.00
AHP_1	10.05	9.95	10.05	9.60	**9.30**
AHP_2	10.00	9.70	**9.45**	9.50	9.50
AHP_3	**9.80**	**9.50**	9.50	**9.20**	9.45
AHP_4	9.90	9.55	9.75	9.30	9.50

Table 3.5: Classification errors [%] on the Satimage database. (Error without combination: 12.05%)

Tables 6.1, 6.2, and 3.5 show the results of our experiments. The columns represent the various classifier sets, while the rows show the classification errors measured using the selected combination of the corresponding classifier group.

As expected, However, in some cases SA performed better than WA and the AHP combiners, telling us that the strong assumptions of the method are not always satisfied [34].

The performance of AHP combiners depends on the size of the testing set. When the test sets selected were too small, the measured accuracy values did not characterize the goodness of the classifiers, and yielded poor combination results. Increasing the size, however, makes the consistency index (CI) tend to zero, producing weights that tend to the values calculated by *weighted averaging*. Determining the optimal size of the test set will probably require further study.

When considering the sensitivity for the selection of different classifier subsets, the AHP-based combiner has a behaviour similar to that of the WA method, hence the optimal classifier set can be selected by methods available for the *averaging* combiners [92].

3.4 Conclusions and Summary

In this chapter the Analytic Hierarchy Process was described in brief. Based on this technique the author designed and implemented a novel linear combination method, and then he compared its performance with those of other combiners. As shown in the experiments, AHP-based combinations proved an effective generalization of the *weighted averaging* rule; they outperformed the other *averaging* methods in almost every case. This dual aspect of simplicity and improved performance was the author's original intention for devising such a method.

4

Speech Recognition

Speech recognition is a pattern classification problem in which a continuously varying signal has to be mapped to a string of symbols (the phonetic transcription). Speech signals display so many variations that attempts to build knowledge-based speech recognizers have mostly been abandoned. Currently researchers tackle speech recognition only with statistical pattern recognition techniques. Here, however a number of special problems arise that have to be dealt with. The first one is the question of the recognition unit. The basis of the statistical approach is the assumption that we have a finite set of units (in other words, classes), the distribution of which is modelled statistically from a large set of training examples. During recognition an unknown input is classified as one of these units using some kind of similarity measure. Since the number of possible sentences or even words is potentially infinite, some sort of smaller recognition unit has to be chosen in a general speech recognition task. The most commonly used unit of this kind is the phoneme, hence the next chapters deal with the classification problem of phonemes.

The other special problem is that the length of the units may vary, that is utterances get warped along the time axis. The only known way of solving this is to perform a search in order to locate the most probable mapping between the signal and the possible transcriptions. Normally a depth-first search is applied (implemented by dynamic programming), but a breadth-first search with a good heuristic is a viable option as well.

4.0.1 Phoneme Modeling

Hidden Markov Models [87] synchronously handle both the problems mentioned above. Each phoneme in the speech signal is given as a series of observation vectors

$$O = o_1, ..., o_T, \tag{4.1}$$

and one has one model for each unit of recognition. These models eventually return a class-conditional likelihood $P(O|c)$, where c refers to these units. The models are composed of states, and for each state we model the probability that a given observation vector belongs to ("was omitted by") this state. Time warping is handled by state transition probabilities, that is the probability that a certain state follows the given state. The final "global" probability is obtained as the product of the proper omission and state-transition probabilities.

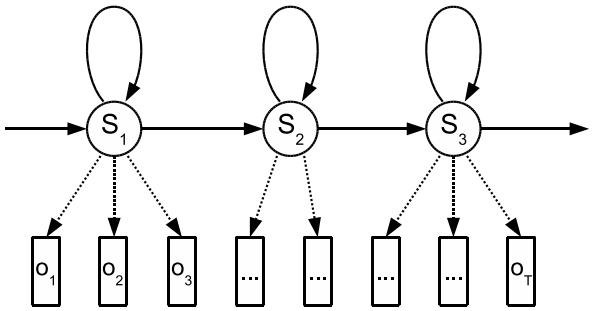

Figure 4.1: The three-state left-to-right phoneme HMM.

When applied to phoneme recognition, the most common state topology is the three-state left-to-right model (see Figure refHMM). We use three states s_1, s_2 and s_3 because the first and last parts of a phoneme are usually different from the middle due to coarticulation. This means that in a sense we do not really model phonemes but rather phoneme thirds.

Because the observation vectors usually have continuous values the state omission probabilities have to be modeled as multidimensional likelihoods. The usual procedure is to employ a mixture of weighted Gaussian distributions for all state s_j of the form

$$p(\rho|s_j) = \sum_{i=1}^{k} \alpha_i \mathcal{N}(\sigma, \mu_i, \Sigma_i), \tag{4.2}$$

where $\mathcal{N}(\sigma, \mu_i, \Sigma_i)$ denotes the multidimensional normal distribution with mean μ_i and covariance matrix Σ_i, k is the number of mixtures, and α_i are non-negative weighting factors which sum to 1.

A possible alternative to HMM are the Stochastic Segmental Models. The more sophisticated segmental techniques fit parametric curves to the feature trajectories of the phonemes [80]. There is, however, a much simpler methodology [60][61] that applies non-uniform smoothing and sampling in order to parametrize any phoneme with the same number of features, independent of its length. The advantage of this uniform parametrization is that it allows us to apply any sort of machine learning algorithm for the phoneme classification task. This is why we chose this type of segmental modeling for the experiments performed and also for our speech recognition system [104].

Hidden Markov Models describe the class conditional likelihoods $P(O|c)$. These type of models are called generative, because they model the probability that an observation O was generated by a class c. However, the final goal of classification is to find

the most probable class c. We can readily compute the posterior probabilities $P(c|O)$ from $P(O|c)$ using Bayes' law since

$$P(c|O) = \frac{P(O|c)P(c)}{P(O)} \tag{4.3}$$

Another approach is to model the posteriors directly. This is how discriminative learners work. Instead of describing the distribution of the classes, these methods model the surfaces that separate the classes and usually perform slightly better than generative models.

In the following chapters in the phoneme classification tests we work both with generative and discriminative methods. But before doing this we shall briefly introduce the OASIS speech recognition system [60][61], developed at the Research Group on Artificial Intelligence, which served as a framework for all the tests.

4.1 The OASIS System

The OASIS system was designed with the aim of creating a general framework that is flexible enough to allow the experimentation with a wide range of techniques in speech recognition. In the following we will give a short overview of the system.

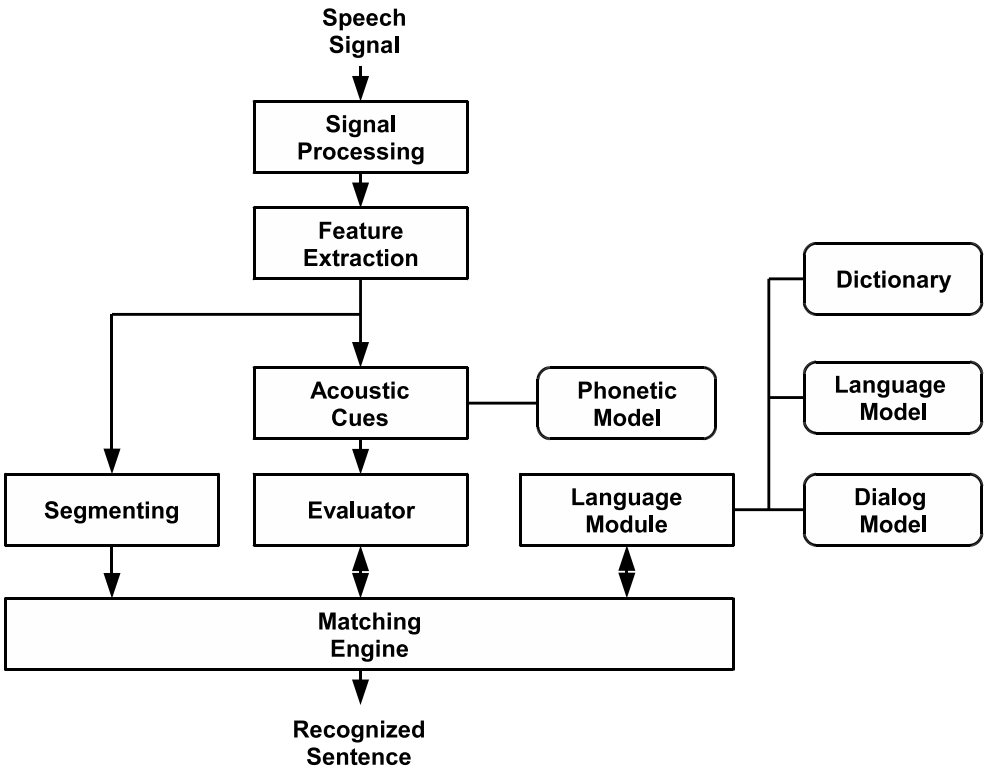

Figure 4.2: Modular structure of the OASIS Speech Lab

4.1.1 The Modular Structure and The Script-based User Interface

The basic execution units of the OASIS Speech Lab are the so-called objects. Similar to the VBScript system of Microsoft, the objects are handled by the component object model (other examples are COM, JavaBeans). Most of the objects may contain further objects and one can assign names to them for identification. On each object services or functions may be defined, and these may depend on other objects given as parameters.

Most of the objects used in the OASIS Speech Lab are so-called modules. The modules are, practically speaking, the kind of special objects that execute some sub-task of the whole signal processing or recognition process. The modules can be interconnected to form a processing work-flow graph – a directed acyclic graph that defines the data flow between the modules. The user's task is to construct a graph from modules and to start the processing. Then the system automatically performs the computations while some of the modules receive a new data block as its input.

The objects of the OASIS Speech Laboratory can be handled through a script-based user interface. Via the Oasis Script Language the user can create the objects, construct a graph from them by specifying their input-output relations, and finally start the processing chain. We will not give a detailed description here of the keywords of the script language and its syntax; rather we will only present an example script at Appendix B, for the reader to get an impression of how the system works.

4.1.2 Auxiliary Modules

The collective term "auxiliary module" here refers to all those modules that do not perform such scientific tasks as signal processing, machine learning or speech decoding. Rather, they are there to make the system more user-friendly. The most important from this group is the **"DatTraverse"** module. It facilitates the batch processing of files by scanning the lines of a file list and passing its items to the input of the subsequent module one at a time. The other group of important auxiliary modules are those that allow the user to graphically display some data. Spectral maps, feature values and segmentation boundaries can all be displayed using them. A special display module helps visualizing the winning hypothesis of a recognition step.

A third category of important auxiliary modules is those routines that can read in and write out data blocks. In particular, sound files (in Microsoft PCM WAV format) can be read in and written out, but there are of course many other types of data that can be exported or imported (for example, spectrographic representations may be saved in BMP format). An interesting case is when we save train and test feature vectors in a text file, so that they can later be processed by machine learning algorithms. The system saves these data blocks in the data format common to the C4.5 learner and the UCI data repository [75].

A very special input module is the **"MicIn"** module that can accept sound data from the microphone. It has to be combined with the **"VoiceDetect"** module that

detects speech activity – but this is now signal processing and leads us to the next group of modules.

4.1.3 Signal Processing and Feature Extraction Modules

The OASIS System implements most of the common signal processing algorithms such as FFT-based spectrum calculation, linear prediction coding and the extraction of cepstral coefficients. The FFT-based spectrum can be transformed to a logarithmic frequency scale by the simulation of Bark-band frequency filters. From these the conventional MFCC coefficients can also be readily obtained. But the HCopy routine of the HTK package [123] can also be called as an external executable, this way guaranteeing a front-end processing identical to that of the HTK.

The pre-processing algorithms listed above all result in a series of vectors – which are all 2-dimensional data sets, and hence in the OASIS system they are called **"Maps"**. The other group of data are of those that are 1-dimensional – in the system they have the collective name **"Feature"**. Of course, any component of a 'map' can be extracted and converted to a 'feature' stream (for example the ith cepstral coefficient or the energy of the ith spectral band). But such basic features as the short-term energy of the signal can also be calculated. Also, several different processing steps can be applied to the features like mean and deviance normalization, differentiation in time (i. e. the computation of Δ coefficients), RASTA filtering, adaptive gain control, and so on.

A special characteristic of the OASIS speech decoders is that they require a list of hypothesized segment boundaries – in short, a segmentation. Segmentations are stored in the so-called **"ClusterBound"** objects of the sytem. As the simplest type of segmentations, we can create a 'fake' segmentation of the signal by assuming a possible segment boundary at each frame. Using this fake segmentation in the decoder, the search space will be the same as that of the conventional frame-based recognizers. But we also have the option of constructing sophisticated segmentation algorithms that yield a much sparser segmentation – thus reducing the search space and speeding up the decoding process. In addition, as a special case of segmentations, we can read in the manually positioned phonetic segment boundary markers of a labelled database. This can be useful, for instance, when we are interested in evaluating the classification abilities of a learning algorithm.

For segment-based recognition every segment has to be represented by the same number of features, independent of its duration. This feature set is called the segment-based features or acoustic cues. Such **"ACue"** objects can be constructed from frame-based features by calculating their mean, deviance, cosine transform coefficients and so on over the duration of the segment. Another way of creating segment-based features is to extract the value of certain frame-based features at special positions such as the start, end or middle points of the segment. Last, but not least, the duration of the segment is yet another important cue that can be extracted as a segment-based feature.

4.1.4 Evaluators

The task of the "Evaluator" modules of the system is to associate probabilities to a given set of a data. In the default case the data is a block of segment-based features, and hence the evaluator returns segment-based phone posteriors or class-conditional phone likelihoods. It is also possible to run evaluator modules over a set of frame-based features – but, as the decoders work over segments, in this case an additional **"CombineEvaluator"** module is required to fuse the frame-based probability estimates into a segment-based one. Both the segment-based and frame-based evaluators have implementations that work with an artificial neural network (ANN), Gaussian mixture models (GMMs), support vector machines (SVMs) and a projection pursuit learner (PPL) – but so far have we conducted extensive tests only with the ANN and GMM based evaluators.

Currently there are two special kinds of evaluators in the system that do not work with spectral data. One of them is the **"AprioriEvaluator"**. As its name suggests, it simply returns the a priori probabilities of the phone classes, based on the frequency counts of the phone labels in the train set. The other one is the **"DurationEvaluator"** that models the phone durations using advanced techniques. Their estimates can be combined with the estimates of the conventional evaluators using proper **"CombineEvaluator"** modules.

4.1.5 The Matching Engine

The task of the matching engine is to traverse all the possible hypotheses (the search space being defined by the segmentation and the phone set), evaluate them (the score of a hypothesis being defined by the acoustic evaluators, the language model and the aggregation strategy inherent to the engine), and to return a ranked list of the best hypotheses. Currently there are three different matching engine modules implemented in the system; they differ in the strategy they apply for traversing the search space. The 'Viterbi Engine' performs a Viterbi-style decoding, that is, a time-synchronous or breadth-first search. The 'Priority Queue Engine' implements stack-decoding that corresponds to a best first-search. The 'Multiple Priority Queue Engine' is a refined version of the previous one in the sense that it stores the hypothesis belonging to different time end points in separate queues.

Although in theory the evaluation of all possible hypotheses guarantees optimal performance, in practice the processing time required for this is prohibitively long. Hence, for fast execution it is very important to find search space pruning heuristics that can throw away unpromising hypotheses without losing the good solutions. In Viterbi encoding it can be done by applying the so-called beam search. In the stack decoding scheme a natural solution is to limit the size of the stacks and thus allow them to discard the least promising partial solutions. These techniques are both implemented in OASIS; more details about efficient decoding in OASIS can be found in the articles by Gábor Gosztolya who developed the matching engines of the system [37, 38, 39, 40].

The result of the recognition is evaluated by comparing it to the transcript belonging

the sound file in the database. This may be an orthographic or a phonetic transcript, depending on whether we perform word or sentence-level recognition, or just phonetic decoding. In the case of isolated word recognition the comparison is quite simple and can be performed by the **"CompareResult"** module. In the case of recognizing phone or word sequences the comparison corresponds to an edit distance calculation. This can be executed by the **"CompareEditDist"** and the **"CompareSentence"** modules (for word and phone sequences, respectively).

4.1.6 Language Models

In most recognition tasks we have linguistic restrictions on the possible phone sequences. The role of the language model is to provide the decoder with the possible phone sequences, along with their corresponding probabilities. In line with the philosophy of the OASIS system, the language models are special kinds of evaluators, but because of their complexity and the conventional separation of the acoustic and language models we choose to discuss them in this separate subsection.

Essentially, we can group the language models into three main categories. In the simplest case we are dealing with an isolated word recognition task. In this case it is sufficient to construct a pronunciation dictionary that simply lists the possible pronunciation(s) of each word. This simple form of language models is implemented by the **"Dictionary"** module of the OASIS system. For efficient storing and decoding, the dictionary is stored in a tree-like compressed form.

Another group of language models are the statistical ones. From these the so-called N-grams [49] are the most popular in speech recognition. These estimate the probability of a word or phone based on its 'history', that is the previous $N-1$ words or phones. The OASIS system is capable of supporting the usage of both word and phone N-grams. They are implemented via the **"BLanguage"** module of OASIS.

In the most difficult case the language model is formal (grammar-based), or a combination of formal and probabilistic techniques. In Hungarian the creation of such a language model raises special problems because of the agglutinative nature of the language. The **"SimpleRTN"** module of the OASIS system contains an implementation of a complex language model that combines context-free grammars and finite state systems to solve these problems. We intended to keep the structure of this module as similar to the language description techniques of other recognizers as possible. So, when designing this sophisticated language model we initially followed the interface of the Microsoft Speech API. It provides an XML description scheme for the definition of context-free grammars, the words themselves being the terminals of the language. However, in a Hungarian listing having all the agglutinated forms of a word stem is intractable. As it happens, Hungarian morphology can be well modelled by finite state systems [35]. Moreover, we observed that the agglutinated forms of a stem can be stored in a much smaller space with transducers than with a traditional compression algorithm. This led us to extend the SAPI description so that transducers could be embedded in the place of terminals. This results in a context-free grammar with its

terminals being the words recognized by the transducer. Further compression can be achieved by applying special automaton compression algorithms which create the smallest possible transducer that models the same language [55]. Additional savings in storage are possible by storing the resulting transducer in a special data structure [56].

As regards the technical details, the implementation of the storage and traversal of the transducers was relatively straightforward. Managing the context-free grammar, however, required the implementation of a recursive transition network that was built on a stack automaton. We also had to store the actual values of the stack, which required special technical solutions.

The SAPI handles probabilities by allowing the user to associate weights with the right hand side alternatives of a rule. The transducers embedded in our extended scheme also allow the weighting of the transitions. So, by combining the two levels, the system is able to associate a probability to any phone sequence.

Independent of the type of modelling, the interface of the language models is adjusted to suit the requirements of the decoder modules. During the extension of a hypothesis the decoders ask for the possible extensions of a phone sequence, so the task of the language model is to return all the possible subsequent phones of a prefix. Based on this, the interface of the language models consists of two functions, together making it possible to iteratively traverse all the phone sequences of the model. These functions are:

`Enter`: Returns the first possible extension of a prefix, along with its probability (or returns a null pointer if there is no extension).

`Next`: Return the next possible extension of the same prefix, along with its probability (or returns a null pointer if there are no more extensions).

4.2 Summary

In this chapter we provied a brief introduction to the speech recognition technology, we compared the frame-based and segmental-based recognition strategies, and presented our speech recognizer framework called OASIS. This sophisticated system served as a designing and testing environment for speech technology research of the Research Group on Artificial Intelligence.

The system was designed and its kernel functions and the auxiliary modules were implemented by the author of this dissertation. He also contributed in coding the signal processing modules (e.g "MicIn" and "SoundDetect"), in the feature extraction modules (especially in implementing real-time processing), and in the language processing modules ("Dictionary" and the first versions of RTN processing). The OASIS System contains combination strategies as a subsystem of the Evaluation modules, which was also designed by the author. The results obtained by this module will be presented in the next chapter.

5

Phoneme Classification

The quality of a speech recognizer application is highly dependent on the performance of the phoneme classifier module applied. Thus, to build better speech recognizer systems, it is advisable to consider the combination strategies of phoneme classifier modules.

In this chapter, we will focus on the phoneme recognition using our segment-based speech recognizer system called "OASIS" as a testing environment. In the first section we will describe the evaluation domain, then we will present the initial frame-based features and show how these are converted into segmental features. This is followed by an elaboration of how the feature extraction techniques of chapters 3 and 4 were applied along with a brief explanation of the learning algorithms used. Finally we discuss the test settings, the results and, of course, their evaluation.

5.1 Evaluation domain

The various hybrid techniques were compared using a relatively small corpus that consists of several speakers pronouncing Hungarian numbers. More precisely, 20 speakers were used for training and 6 for testing, and 52 utterances were recorded from each person. The ratio of male and female speakers was 50%:50% in both the training and the testing sets. The recordings were made using an inexpensive commercial microphone in a reasonably quiet environment, at a sample rate of 22050Hz. The whole corpus was manually segmented and labelled. Because the corpus contained only numbers, we had occurrences of only 32 phones, which is approximately two-thirds of the Hungarian phoneme set. Because some of these labels represented only allophonic variations of the same phoneme, some labels were fused, and so we actually worked with a set of just 28 labels. We performed tests as well with two other groupings where the labels were grouped into 11 and 5 classes, respectively, based on phonetic similarity. We had two good reasons for doing experiments with these gross phonetic classes. First, we

could increase the number of training examples in each class and inspect the effects of this on the learning algorithms. Second, our speech recognition system has a first-pass stage in which the segments are classified into gross phonetic categories only.

Hence we had three phonetic groupings, which will be denoted by *grp1*, *grp2*, and *grp3* from this point on. With the first grouping, the number of occurrences of the different labels in the training set was between 40 and 599. This value was between 120 and 1158 for the second grouping and between 716 and 2158 for the third grouping.

5.1.1 Acoustic features

In the following we describe the initial feature extraction techniques which were employed in our tests. For each test a certain subset of these features was chosen.

Critical Band Log-Energies (CBLE)

Before the initial feature extraction, the energy of each word was normalized. After this the signals were processed in 512-point frames (23.2 ms), where the frames overlapped by a factor of 3/4. A Fast Fourier Transform was applied on each frame. Next, critical band energies were approximated by the use of triangular-shaped weighting functions. 24 such filters were used to cover the frequency range from 0 to 11025Hz, the bandwidth of each filter being 1 bark. The energy values were then measured on a logarithmic scale.

Mel-Frequency Cepstral Coefficients (MFCC)

In order to incorporate the most common preprocessing method (namely MFCC) into our features, we made additional tests after taking the discrete cosine transform (DCT) of the critical band log-energies calculated above. The test used the first 16 coefficients (including the zeroth one). A point which should be mentioned here is that since the spectrum has already been smoothed by the critical band filters, the calculation of the cepstrum does not fulfil its original task of removing the effect of pitch. Instead, its supposed benefit is the decorrelation of features. In fact, it can be proved that the DCT approximates the PCA quite closely for most signals, so it is worth comparing the classification results for MFCC with critical band log-energies plus PCA.

Formant Band Gravity Centres (FBGC)

Besides the ones above we also wanted to do experiments with some more phonetically based features like formants. However, since we had no reliable formant tracker (which is known to be a very difficult task anyway), we instead used gravity centres as a crude approximation for formants . The gravity centres were calculated from the power spectrum in the following four frequency bands:

200 Hz - 1400 Hz,
1000 Hz - 3000 Hz,
2500 Hz - 4000 Hz,
3000 Hz - 11025 Hz

The formula for the gravity centre $G(a,b)$ of a band $[a,b]$ is

$$\mathcal{G}(a,b) = \frac{\int_a^b fS(f)df}{\int_a^b S(f)df}, \tag{5.1}$$

where $S(f)$ denotes the power spectrum.

The tests were performed on five feature sets[62] described later. Because all sets contained duration information, we do not mention this separately. *Set1* consisted of the MFCC coefficients, because these are the most commonly used features. To have the chance of studying the usefulness of the cosine transform, we also carried out tests on the filter bank energies themselves (*Set3*). By augmenting *Set1* and *Set3* with gravity centre features, we got two new sets, *Set2* and *Set4*. We hoped that the addition of these phonetics-based features would lead to a slight improvement. Lastly, the largest set *Set5* contains all the features, that is, filter bank energies, MFCC coefficients, and gravity centres. The same trials were performed on the three phoneme groupings *grp1, grp2, grp3*.

5.1.2 Classifiers

Each of the classifiers used in the experiments was modified so as to make them capable of providing a posteriori probabilities for each class ω_k.

Decision Tree Learner:

Our version of DTL used in the experiments is based on the C4.5 [86] tree learning algorithm. It is able to learn predefined discrete classes from labelled examples. The result of the learning process is an axis-parallel decision tree. This means that during the training, the sample space is divided into subspaces by hyper-planes which are parallel to every axis but one. In this way, we obtain many n-dimensional rectangular regions that are labelled by class labels and organized in a hierarchical way so it can be encoded into a tree. For knowledge representation, DTL uses the "divide and conquer" technique, which means that regions are split during learning whenever they are insufficiently homogenous, and left untouched when they are homogenous. Splitting is done by axis parallel hyper-planes and, thanks to this, the learning process is quite fast. Hence the greatest advantage of the method is time complexity. Unfortunately, the simple learning strategy in certain cases results in a huge number of regions that are needlessly split.

Gaussian Mixture Models:

GMM[24] assumes that the class-conditional probability distribution $p(\mathbf{x_i}|\omega_k)$ can be well-approximated by a distribution of the form

$$f(\mathbf{x_i}) = \sum_{j=1}^{l} c_j \mathcal{N}(\mathbf{x_i}, \mu_\mathbf{j}, \mathbf{C_j}), \tag{5.2}$$

where $\mathcal{N}(\mathbf{x_i}, \mu_\mathbf{j}, \mathbf{C_j})$ denotes the multidimensional normal distribution with mean $\mu_\mathbf{j}$ and covariance matrix $\mathbf{C_j}$, l is the number of mixtures and c_j are nonnegative weighting factors that sum to 1.

As luck would have it, there is no closed formula for the optional parameters of the mixture model. Normally the expectation-maximization (EM) algorithm is utilized to find proper parameters, but it guarantees only a locally optimal solution. This iterative technique is very sensitive to initial parameter values, so we used k-mean clustering to find a good starting parameter set. Since k-mean clustering again guarantees only a local optimum, we ran it 15 times with random parameters and took the one with the highest log-likelihood to initialize the EM algorithm. In each case the covariance matrix was made diagonal because training full matrices would have required much more training data and computation time.

Support Vector Machines:

SVM[113] was developed by Vapnik for binary classification. It selects a hyperplane with maximal margin to separate points with different class labels, but prior to that it applies a nonlinear transformation to map the patterns to a higher dimensional space where the classification is easier. The problem of nonlinearity is handled by kernel functions, which makes Support Vector Machines a very powerful tool for machine learning.

Artificial Neural Networks:

ANNs[8] now count among the conventional pattern recognition tools, so we will not describe them here. In the trials performed we used the most common feed-forward multi-layer perceptron network with the back-propagation learning rule. The number of neurons in the hidden layer was set at three times the number of features (this value was chosen empirically based on preliminary trials). Training was stopped when, for the last 20 iterations, the decrease in the error between two consecutive iteration step remained below a certain threshold.

k Nearest Neighbour:

kNN[24] is a well known classifier used in pattern recognition. Because no rule or decision is made before the actual classification, this approach is called *lazy learning*. Typically, this kind of machine learning has a very short training time but the classification of new data takes rather a long time. The storing and processing of millions of

	ANN	SVM	GMM	kNN	DTL
g1set1	**13.71%**	14.01%	21.16%	20.09%	35.58%
g1set2	**12.23%**	13.71%	24.65%	19.86%	33.75%
g1set3	11.88%	**11.76%**	26.77%	21.34%	32.80%
g1set4	12.23%	**11.82%**	25.59%	21.99%	32.09%
g1set5	11.88%	**11.41%**	24.36%	19.24%	34.22%
g2set1	10.64%	**9.93%**	18.68%	13.83%	22.10%
g2set2	10.40%	**9.69%**	21.69%	13.48%	22.46%
g2set3	8.63%	**8.22%**	18.68%	12.12%	21.75%
g2set4	10.76%	**7.92%**	18.26%	13.00%	21.45%
g2set5	9.93%	**8.16%**	20.33%	12.41%	24.76%
g3set1	7.15%	**6.03%**	11.76%	8.51%	13.38%
g3set2	6.32%	**5.56%**	11.82%	7.51%	12.71%
g3set3	**5.38%**	5.61%	10.22%	6.86%	10.87%
g3set4	5.32%	**5.14%**	10.46%	7.39%	12.00%
g3set5	5.61%	**4.96%**	10.11%	6.86%	12.35%

Table 5.1: Classification errors of the individual classifiers

examples can also be a serious handicap. Despite this, being a *stable* inducer it is an excellent tool for machine learning.

5.1.3 Results of the standalone classifiers

In the first stage all the classifiers were tested on the same data set. As can be seen in Table 5.1, SVM performed the best, ANN also had a good score, but the other classifiers only produced poor results.

5.1.4 Selecting Classifier Set

One can expect a different performance depending on which classifiers are combined. To obtain the optimal classifier selection, a different subset of the classifiers was selected for combining with the same method. The subsets were generated sequentially by inserting the next element from the strictly ordered list of classifiers into the previous subset. The combination rule applied in the test was the Product rule.
The above test results in Table 5.2 show that there is little point in using all the classifiers in combination schemes, as the optimal solution is a combination of SVM, ANN, and kNN. Including GMM and DTL only leads to a deterioration in the classification performance.

5.1.5 Comparing combination rules

In the next stage of the testing we combined the outputs of the selected classifiers (SVM, ANN, and kNN) by applying various combination rules. Table 5.3 suggests that there is no definite optimal rule for combining classifiers using this database. Combiners

		ANN SVM	***A*NN SVM kNN**	**ANN SVM kNN GMM**	**ANN SVM kNN GMM DTL**
g1set1	13.71%	12.46%	**12.00%**	15.07%	29.91%
g1set2	12.23%	**11.70%**	11.76%	18.14%	27.42%
g1set3	11.76%	**11.05%**	11.70%	18.44%	27.96%
g1set4	11.82%	**11.05%**	12.77%	19.62%	28.25%
g1set5	11.41%	10.99%	**10.87%**	19.39%	27.66%
g2set1	9.93%	10.11%	**8.63%**	10.93%	17.43%
g2set2	9.69%	9.69%	**8.76%**	12.12%	16.61%
g2set3	8.22%	7.80%	**7.03%**	12.29%	16.19%
g2set4	**7.92%**	9.10%	8.98%	13.18%	17.14%
g2set5	**8.16%**	9.46%	8.51%	13.95%	19.33%
g3set1	6.03%	6.21%	**5.14%**	7.98%	8.04%
g3set2	5.56%	5.67%	**5.02%**	7.74%	9.04%
g3set3	5.38%	**4.96%**	5.14%	7.92%	7.21%
g3set4	5.14%	5.02%	**4.67%**	8.22%	9.40%
g3set5	**4.96%**	5.50%	5.02%	7.74%	9.10%

Table 5.2: Combination error obtained using the Product Decision Rule

which applied the Sum rule performed the best, but the improvement compared with the others was only marginal.

5.1.6 Results using Bagging

In this part the Bagging algorithm was applied to each of the classifiers. During the trials 15 bootstrap samples were generated, each of them with a size two-thirds that of the size of the original training-set.
As can be seen (Table 5.4), Bagging can improve classification performance, almost to the same level of the previous combination methods, but it requires more processing time.

5.1.7 Results using Boosting

Because Boosting is an improvement on Bagging, we expected a better performance. Testing Boosting on this data-set, however, produced roughly the same classification error values. The explanation for this is that the classifiers are "too strong", they generated very small classification errors when using the training set. After the first step of Boosting, only the "noise" remained in the bootstrap sample, which was too difficult to separate, and the classification error on this sample generally hit the 50% limit. Here the algorithm exits, but in practice a standard Bagging (uniform bootstrapping) step can be performed instead. The result (Table 5.5) is very close to that for the Bagging

	Prod	Sum	Max	Min	Borda	Voting
g1set1	12.00%	**11.64%**	12.59%	12.77%	12.77%	13.23%
g1set2	11.76%	**11.41%**	12.06%	13.06%	12.06%	11.47%
g1set3	11.70%	11.35%	13.18%	12.29%	11.64%	**10.87%**
g1set4	12.77%	12.41%	14.24%	12.35%	12.59%	**11.41%**
g1set5	10.87%	**10.70%**	11.88%	11.17%	11.41%	11.17%
g2set1	8.63%	**8.45%**	9.22%	9.10%	8.87%	9.16%
g2set2	8.75%	**8.57%**	9.87%	9.16%	9.10%	9.04%
g2set3	**7.03%**	7.09%	8.04%	7.33%	7.80%	7.15%
g2set4	8.98%	7.98%	9.87%	9.34%	8.69%	**7.74%**
g2set5	8.51%	8.22%	8.98%	**7.98%**	8.81%	8.51%
g3set1	**5.14%**	**5.14%**	6.32%	5.61%	5.61%	5.56%
g3set2	5.02%	5.50%	5.73%	**4.85%**	5.08%	5.08%
g3set3	5.14%	**4.91%**	5.26%	5.20%	5.08%	**4.91%**
g3set4	4.67%	**4.61%**	5.02%	4.79%	5.02%	4.91%
g3set5	5.02%	**4.96%**	5.08%	5.14%	5.14%	5.14%

Table 5.3: Classification error of hybrid combinations using ANN, SVM, and kNN

	ANN	SVM	GMM	kNN	DTL
g1set1	12.71%	12.59%	19.80%	20.45%	26.36%
g1set2	11.76%	12.23%	23.05%	19.21%	22.70%
g1set3	10.99%	10.28%	22.70%	21.10%	22.87%
g1set4	11.88%	11.29%	22.94%	22.28%	21.57%
g1set5	10.70%	10.82%	21.22%	20.21%	21.39%
g2set1	11.17%	9.52%	14.83%	13.95%	16.84%
g2set2	9.75%	9.40%	18.38%	13.65%	17.32%
g2set3	8.16%	7.45%	15.96%	12.83%	16.90%
g2set4	8.69%	7.51%	16.67%	13.12%	16.31%
g2set5	9.57%	7.86%	16.67%	12.65%	16.37%
g3set1	6.80%	5.44%	11.05%	8.87%	11.23%
g3set2	6.38%	5.08%	10.64%	7.80%	10.52%
g3set3	5.79%	5.50%	10.11%	7.51%	11.82%
g3set4	5.26%	4.61%	09.57%	7.15%	10.22%
g3set5	6.09%	5.02%	9.63%	7.09%	10.40%

Table 5.4: Classification error of Bagging classifiers

	ANN	SVM	GMM	kNN	DTL
g1set1	12.51%	12.44	18.55%	20.32%	25.12%
g1set2	11.56%	11.97	22.05%	19.42%	22.25%
g1set3	10.79%	9.67	20.12%	20.87%	21.98%
g1set4	11.88%	10.43	20.05%	22.16%	21.60%
g1set5	10.66%	10.34	21.22%	19.87%	20.47%
g2set1	11.17%	9.72	14.87%	13.95%	16.75%
g2set2	9.75%	9.31	17.12%	13.87%	15.88%
g2set3	8.42%	7.14	15.27%	12.83%	16.36%
g2set4	8.64%	7.62	14.98%	13.07%	16.59%
g2set5	9.77%	7.36	15.14%	12.68%	15.72%
g3set1	6.76%	5.32	11.23%	8.97%	11.17%
g3set2	6.41%	4.87	9.96%	7.72%	10.69%
g3set3	5.63%	5.50	10.03%	7.31%	11.47%
g3set4	5.28%	4.82	9.98%	7.22%	9.93%
g3set5	6.02%	4.91	9.61%	6.98%	10.05%

Table 5.5: Classification error of Boosting classifiers

algorithm.

5.2 Conclusions and Summary

In this chapter we surveyed the various combination schemes available using speech recognition oriented data-sets. The author designed and implemented the combination methods, and integrated them as submodules into the "OASIS" speech recognizer framework. The experimental results show that making classifier hybrids improved the discrimination performance, the best results being obtained by aggregating the output of SVM, ANN, and kNN. The performance of the combiners applying different decision rules was not significantly different, but the "Sum" rule outperformed the others. Comparing the traditional Bagging and Boosting techniques, we found that they had nearly the same classification improvement, but their applicability was limited because they were too CPU intensive. The findings suggest that it is worth applying combination techniques in phoneme-level speech recognition systems because they will hopefully produce better scores, hence improve the overall results.

6

Vocal Tract Length Normalization

As we mentioned earlier, the efficiency of a speech recognizer application is highly dependent on the performance of the phoneme classifier module used. Also, it is just as important for a phonetic teaching system like our Phonological Awareness Teaching System called "Speech-Master". Since the system should work well both for children and adults of different ages, the recognizer has to be trained with the voices of users of both genders and of practically any age. The task is also special because it has to recognize isolated phones, so it cannot rely on language models. Consequently, there is a heavy burden on the acoustic classifier, and we need to apply any helpful trick that might improve the overall performance.

One of the available techniques is speaker normalization, or more specifically, vocal tract length normalization (VTLN), which proves very useful when the targeted users vary greatly in age and gender. Applying off-line vocal tract length normalization algorithms [21, 25, 83, 111, 119, 120, 121], one can build recognizers that work robustly with voice samples from males, females and children.

In this chaper we will examine how the combination techniques affect the performance of the on-line normalization techniques, and we will show that with a careful selection of combiners it is possible to surpass the accuracy of the off-line methods. This chapter is organized as follows. The following section give a brief introduction to the the Vocal Tract Length Normalization methods, and then it examines the various parameter estimation techniques. Section 2 describes the evaluation domain, the database, the available features and the classifier algorithms used for the inspection. In Section 3 is concerned with the combination techniques, focusing on the traditional and linear combination schemes. The experiments section compares the performance of the various VTLN and combination methods. Lastly, we give some brief conclusions, and make some suggestions about classifier combinations.

6.1 The Model used for VTLN

One of the major physiological sources of inter-speaker variation is the vocal tract length of the speakers (Fig. 6.1). In [21] the average vocal tract length for men was reported

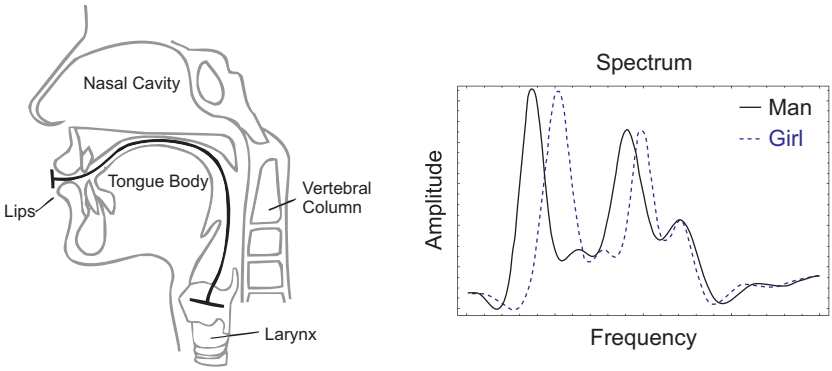

Figure 6.1: Vocal Tract Length and its frequency shifting. The graph drawn with solid and dashed line shows the spectrum of a vowel uttered by a man and a girl, respectively.

to be 17 cm, for women it was 15 cm, and for children it was 14 cm. The formant frequency positions are inversely proportional to vocal tract length and this causes a shift of the formant centre frequencies. Consequently, VTLN is usually performed by warping the frequency scale.

Modelling the vocal tract as a uniform tube of length L, the format frequencies are proportional to 1/L. Thus the simplest approaches use a linear warp. In reality, however, the vocal tract is more complex than a uniform tube. That is why many more sophisticated warping functions have been proposed in the literature [25, 119]. Some of the commonly applied warping functions are:

- Linear warping function
$$f' = kf \tag{6.1}$$

- Exponential warping function
$$f' = k_s^{3f} f, \tag{6.2}$$

- Bilinear warping function
$$f' = f + \frac{2}{\pi}\tan^{-1}\frac{(k-1)\sin(\pi f)}{1-(k-1)\cos(\pi f)}, \tag{6.3}$$

- Piecewise linear warping function (vertical)
$$f' = \begin{cases} kf & 0 \le f \le t \\ k_2 f_(1-k_2) & t < f \le 1 \end{cases}, \tag{6.4}$$

where $t = \frac{h}{k}$ $k_2 = \frac{1-kt}{1-t}$.

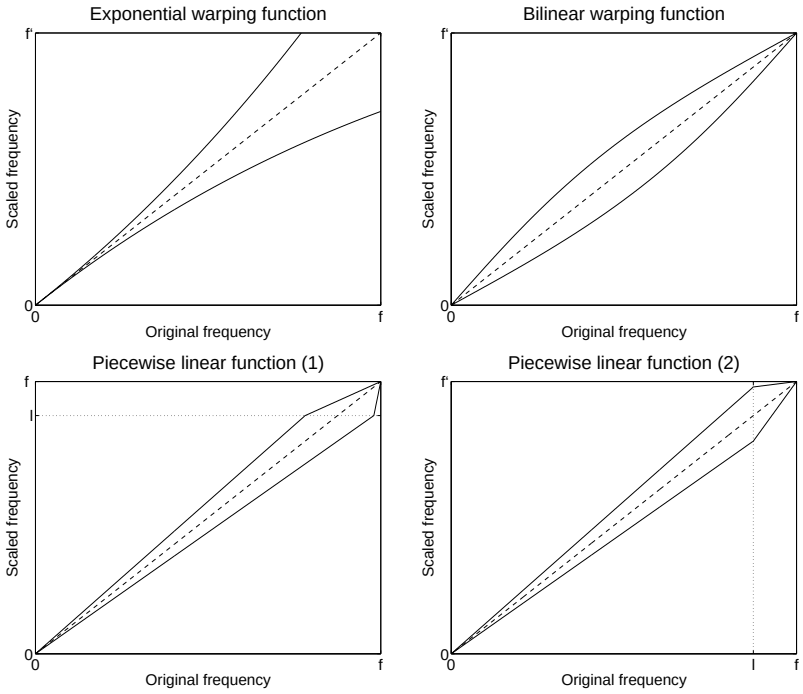

Figure 6.2: Examples of VTL warping functions. The figures show the mapping between the original (horizontal axis) and the warped (vertical axis) frequencies.

- Piecewise linear warping function (horizontal)

$$f' = \begin{cases} kf & 0 \leq f \leq t \\ k_2 f_(1 - k_2) & t < f \leq 1 \end{cases} , \tag{6.5}$$

where $t = h$ $k_2 = \frac{1-kt}{1-t}$,

Given a warping function, normalization can be implemented either by re-sampling and interpolating the spectrum or modifying the width and centre frequencies of the mel (Bark) filter bank.

6.1.1 VTLN parameter estimation

The linear discriminant (LD) criterion is defined using the covariance matrices of a given sample set over a speech database. Each sample is placed in a phonetic class and the samples that belong to a given speaker are extracted using the same warping parameter. The task is then to optimize these parameters for each speaker according to the LD criterion:

$$LD = \frac{|B|}{|W|} , \tag{6.6}$$

where B is the between-class and W is the within-class covariance matrix. The value of the LD criterion is small if the different classes are spaced out and each of them has a small scatter around the class centres. While optimizing the warping parameters of all the speakers at the same time is impractical, an iterative process (Algorithm 3) can be applied.

Algorithm 3 LD-VTLN parameter estimation

Choose an initial warping factor for each speaker and warp the samples
while the average warping factor variation is above the set threshold **do**
  for all speakers **do**
    calculate the LD criterion for each α value in a small neighbourhood of the current warping parameter
    select the best warping parameter
  end for
  Update the sample set using the optimal warping factors that are obtained
end while

This optimization method, however, works off-line. So a natural question then arises. Is it possible to efficiently estimate the optimal parameters obtained by off-line algorithms using machine learning regression methods that work on-line?

6.1.2 Real-time VTLN

LD-VTLN parameter estimation requires all the utterances in advance. Real-time recognition systems, however, require an instantaneous response. To have the advantages of speaker normalization in on-line systems, machine learning methods can be applied to the estimation of the correct parameters. Out of the many possible regression techniques we chose to experiment with neural nets. Their task is to estimate the optimal LD-VTLN warping parameter for each speaker based on the actual spectral frame without warping. Paczolay [82] compared the performance of these kinds of on-line method with those off-line LD-VTLN parameter estimation techniques.

6.2 Evaluating Domain

Firstly we will describe the corpus, the feature extraction technique, then the classifiers and regression algorithms used in the tests.

For training and testing purposes we recorded samples from 240 speakers, namely 60 women, 60 men, 60 girls and 60 boys. The children were aged between 6 and 9. The speech signals were recorded and stored at a sampling rate of 22050 Hz in 16-bit quality. Each speaker uttered all the Hungarian vowels, one after the other, separated by a short pause. Since we decided not to discriminate their long and short versions, we only worked with 9 vowels altogether.

6.2.1 Feature Sets and Classifier

The signals were processed in 10 ms frames, the log-energies of 24 critical-bands being extracted using FFT and triangular weighting [87]. The energy of each frame was normalized separately, which meant that only the spectral shape was used for classification.

In all the classification experiments the Artificial Neural Nets (ANNs) [8] employed

here were the well-known three-layer feed-forward MLP networks trained with the back-propagation learning rule. The number of hidden neurons was equal to 16.

6.2.2 Warping parameter estimation

In order to test the performance of the VTLN techniques, we transformed the original features of the databases using the calculated α warping parameter for each pattern and then generated a database for each set of output data. To estimate parameter values the following methods were applied:

- **LD-VTLN:** The initial value of the warping parameter α was set to 0. For the optimization the value of α in this interval was quantized – it could take one of 15 discrete values. The iteration was stopped when the average change in the warping parameter fell below 10^{-2}.
- **RT-VTLN:** For the learning of the parameter α of the warping function a special MLP network was constructed with one output neuron and two hidden layers with 32 and 24 neurons, respectively. Training was performed with respect to mean square error.

6.3 Combination strategies

Since the applied classifier algorithm, a Multilayer Perceptron network supplies the class-conditional probabilities for all the class labels, any of the confidence level combination strategies can be utilized for improving the efficiency the Vocal Tract Length Normalization.

6.3.1 Multi-Model classifier

Multi Model Classifiers is a special classifier combination, where the combined class probabilities are given by:

$$\hat{p}^j(x) = \sum_{i=1}^{N} w_i(x) p_i^j(x) \tag{6.7}$$

Here the weight vector w depends on the current pattern. In addition only one component of w can be non-zero, so this weighting selects a classifier whose output appears as the output of the given combination, i.e.:

$$w_i(x) = \begin{cases} 1 & \text{if } i = \sigma(x) \\ 0 & \text{otherwise} \end{cases}, \tag{6.8}$$

where σ is a special function which selects a classifier for the current pattern. To implement this function any possible machine learning method can be applied.

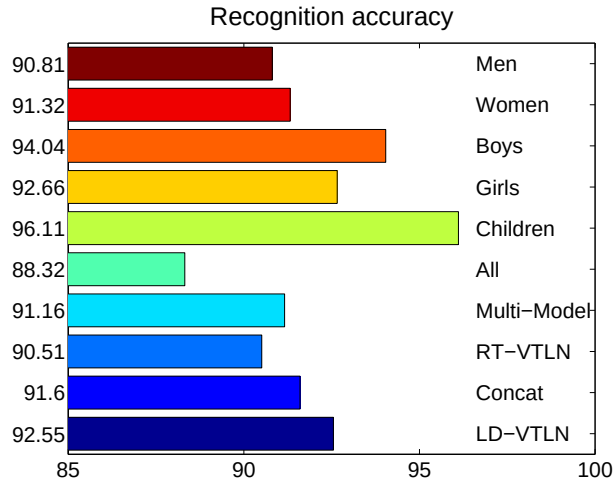

Table 6.1: Recognition accuracies of the standalone classifiers (in percent). The parameter estimation of LD-VTLN requires all pattern data in advance, thus this method cannot be utilized in real recognition systems, its performance can be regarded as a reference value for the other normalization techniques.

6.4 Classification Results

The experiments were conducted as follows. We employed 5-fold leave-one-out cross-validation, keeping the ratio of boys, girls, men and women uniform in each case. So the train and test sets had a ratio of 4 : 1. Separating the original database named "All" according to the speakers gender and age, we obtained databases for "Men", "Women", "Boys", "Girls", and "Children". On each database we trained an MLP classifier.

For "Multi model" evaluations, we divided the train set into the following 3 categories: men, women, and children. On each of them we trained an MLP as classifier, and afterwards we trained an MLP network with 16 hidden neurons to select the category of a given pattern.

To make our regression-based normalization RT-VTLN more robust, we generated a new concatenated database called "Concat" that, besides the warped features, contains the original features as well.

Because the parameter estimation of LD-VTLN requires all pattern data in advance, this method cannot be utilized in real recognition systems, so we treated its performance as a reference value for the other normalization techniques.

Table 6.1 shows the classification accuracies measured on the chosen databases. It was clear that the performance on the separated categories was significantly better than that of the original, the 'Multi Model' method being based on this experiment. Of the VTLN techniques we applied, LD-VTLN performed the best, improving the accuracy by 36 %. But being off-line technique, it requires all data in advance to work, so this cannot be applied to real classification problems. The run-time VTLN produced a moderate performance compared to the LD-VTLN, but this difference could be halved using not only the warped but also the original features ("Concat").

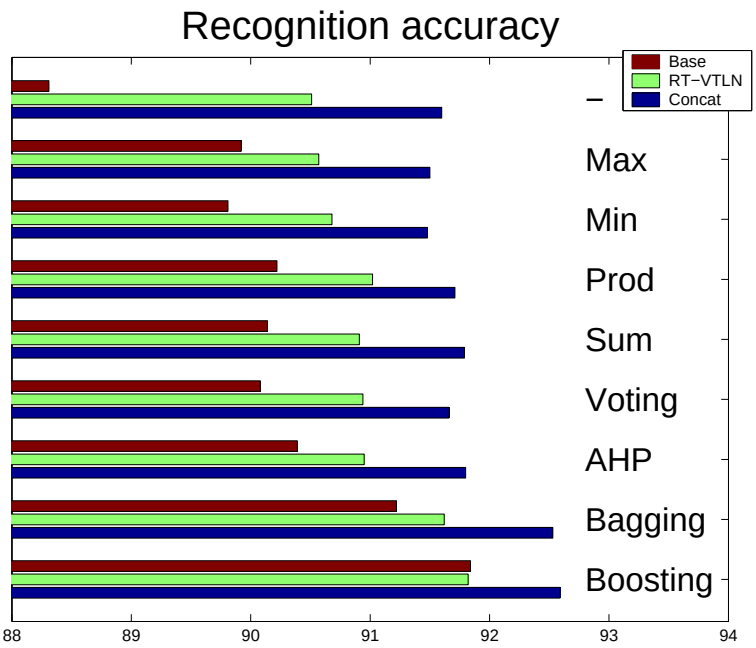

Table 6.2: Recognition accuracy on the databases with combination (in %). The various bars in each triplet correspond to the databases, and bar-triplets represent the applied combiner.

6.5 Combination results

For the combiners "Max", "Min", "Prod", "Sum", and "Voting" we exploited the 8-fold rotation of the database to generate 8 data-sets as training sets for the classifiers. After training MLP networks on the corresponding sets we utilized a given set of combiners on them, and measured their performances on a common test data-set.

To calculate the pairwise comparison matrices needed for the AHP method, we took the quotient of classification errors of the two competing networks on a random test set generated by bootstrapping (resampling the training data set with replacement) of the training set.

"Bagging" and "Boosting" generate their own sequence of training data-sets; in these cases we ran the methods on the original database directly, setting the max iteration number to 50.

The experimental results are shown in Table 6.2. For each combiner the recognition accuracy was measured on 3 different databases: "All", "RT-VTLN", and "Concat". The effect of the classifier combination depends on the database complexity. With the "All" database, each of the combiners has a better performance than that for the original classification. On the warped databases ("RT-VTLN" and "Concat") the traditional combinations have less influence. Bagging and Boosting gave the best scores due to the large number of classifiers used.

Comparing the above results with the reference figure of LD-VTLN (92.55%), we may conclude that with a properly selected combination scheme the regression based

real-time VTLN method (Boosting on Concat, 92.67%) can outperform the results of the off-line method LD-VTLN.

6.6 Conclusions and Summary

In this chapter we examined the effect of Vocal Tract Length Normalization techniques and their combination strategies on the phoneme classification performance. The author showed that the on-line parameter estimation methods can be handled as special combination structures, and he implemented the adapted versions of combiner strategies like "Bagging" and "Boosting". The results demonstrated that using combination stategies of the on-line methods, the overall system can achieve nearly the the same recognition quality as that with the off-line normalization version, while applying Bagging and Boosting may produce classifiers with better performances than those for LD-VTLN. With these positive results the implemented module was integrated into the award-winning Phonological Awareness Teaching System called "Speech-Master".

7

POS Tagging

Part-of-speech tagging (POS tagging or POST), also called grammatical tagging, is the process of marking words in a text that correspond to a particular part of speech, based on both their definition and their context, i.e. the relationship between adjacent and related words in a phrase, sentence, or paragraph. A simplified form of this is commonly taught to schoolchildren, in the identification of words as nouns, verbs, adjectives, adverbs, and so on. Being a building block of many Human Language Technology applications like NP parsing, a number of strategies have been proposed in the literature to improve its performance.

In this chapter we will review the current state-of-the-art in Hungarian POS tagging, and investigate the classifier combination strategies that can improve the overall classification performace of such systems. In the first section, the most important published results of the last few years in Hungarian POS tagging are summarized. After we discuss our choice of the TBL algorithm as POS-tagger in details in Section 7.1. In Section 7.3 we examine the combination stuctures used in the POS tagging application, and then we propose a new strategy for adaptive POS tagger ensemble systems. The results of the boosted tagger are presented in Section 7.4.

7.1 POS Tagging of Hungarian Texts

Standard POS tagging methods were applied to Hungarian as soon as the first annotated corpora appeared that were big enough to serve as a training database for various methods. The TELRI corpus [23] was the first corpus that was used for testing different POS tagging methods. This corpus contains approximately $80,000$ words. Later, as the Hungarian National Corpus [115] and the Manually Annotated Hungarian Corpus (the Szeged Corpus) [3] became available, an opportunity was provided to test the methods on bigger corpora (153M and 1.2M words, respectively).

In recent years several authors have published many useful POS tagging results in

Hungarian. It is generally believed that, owing to the fairly free word order and the agglutinative property of the Hungarian language, there are more special problems associated with Hungarian than those of the Indo-European languages. However, the latest results are comparable to results achieved in English and other well-studied languages. Fruitful approaches for Hungarian POS tagging are Hidden Markov Models, Transformation Based Learning and rule-based learning methods.

One of the most common POS tagging approaches is to build a tagger based on Hidden Markov Models (HMM). Tufis [109] reported good results with the Trigrams and Tags (TnT) tagger [12]. A slightly better version of TnT was employed by Oravecz [79], and it achieved excellent results. In their paper, Oravecz and Dienes [79] argue that regardless of the rich morphology and relatively free word order, the POS tagging of Hungarian with HMM methods is possible and effective once one is able to handle the data sparsity problem. They used a modified version of TnT that was supported by an external morphological analyzer. In this way the trigram tagger was able to make better guesses about the unseen words and therefore to get better results. An example of the results achieved by this trigram tagger is presented in the first row of Table 7.2.

Another approach which is distinct from the statistical methods is the rule-based learning one. A valuable feature of the rule-based methods is that the rules these methods work with are usually more intelligible to humans than the parameters of statistical methods. For Hungarian, a few such approaches are available in the literature.

In a comprehensive investigation, Horváth et al. [47] applied five different machine learning methods to Hungarian POS tagging. They tested C4.5, PHM, RIBL, Progol and AGLEARN [2] methods on the TELRI corpus. The results of C4.5 and the best tagger found in this investigation (RIBL) are presented in the second and third rows of Table 7.2. Hócza [44] used a different rule generalization method called RGLearn. Row 4 shows the test results of that tagger in Table 7.2. Transformation Based Learning is a rule-based method that we will discuss in depth in the next section. Megyesi [73] and Kuba et al. [68] produced results with TBL taggers that are given in Table 7.2, in rows 5 and 6, respectively.

7.2 The TBL tagger

Transformation Based Learning (TBL) was introduced by Brill [17] for the task of POS tagging. Brill's implementation consists of two processing steps. In the first step, a lexical tagger calculates the POS tags based on lexical information only (word forms). The result of the lexical tagger is used as a *first guess* in the second run where both the word forms and the actual POS tags are applied by the contextual tagger. Both lexical and contextual taggers make use of the TBL concept.

During training, TBL performs a greedy search in a rule space in order to find the rules that best improve the correctness of the current tagging. The rule space contains rules that change the POS tag of some words according to their environments. From these rules, an ordered list is created. In the tagging phase, the rules on the rule list

Test Domain	Baseline	TBL
full corpus	94.94%	96.52%
business news	97.56%	98.26%
cross-domain	79.51%	95.79%

Table 7.1: Accuracy of the baseline tagger and the TBL tagger.

Tagger	Accuracy
TnT + Morph. Ana.	98.11%
C4.5	97.60%
Best method (RIBL)	98.03%
RGLearn	97.32%
TBL	91.94%
TBL + Morph. Ana.	96.52%
Best combination	96.95%

Table 7.2: Results achieved by various Hungarian POS taggers.

are applied one after another in the same order as the rule list. After the last rule has been applied, the current tag sequence is returned as a result.

For the Hungarian language, Megyesi applied this technique initially with moderate success. [71] The weak part of her first implementation was the lexical module of the tagger, as described in [73]. With the use of extended lexical templates, TBL produced a much better performance but still lagged behind the statistical taggers.

We chose a different approach that is similar to the method in [68]. The *first guess* of the TBL tagger is the result of the baseline tagger. For the second run, the contextual tagger implementation we employed is based on the fnTBL learner module [78]. Here we used the standard parameter settings included in the fnTBL package.

7.2.1 Baseline Tagger

The baseline tagger relies on the Humor [84] morphological analyzer to get the list of possible POS tags. If the word occurs in the training data, the word gets its most frequent POS tag in the training. If the word does not appear in the training, but representatives of its ambiguity class (words with the same possible POS tags) are present, then the most frequent tag of all these words will be selected. Otherwise, the word gets the first tag from the list of possible POS tags.

Some results produced by the baseline tagger and the improvements obtained by the TBL tagger are given in Table 7.1.

7.3 Combination strategies

7.3.1 Related works

An exhaustive investigation on combinations of different POS taggers is available in [110]. Voting strategies and multi-level decision methods (stacking) have been investigated also. For Hungarian, a third possible approach was studied by Horvath et. al [47]. This was a cascade combinations of the taggers, which means that the output of one tagger is the input of another. Kuba et al. [68] performed experiments by applying cascade and "Majority voting" combinations of various tagger methods. Cascading TBL with any other tagger is a quite straightforward idea. The *first guess* that TBL starts with can be the output of another tagger. This kind of concatenation naturally requires that training of the TBL tagger should take place on examples tagged by the first tagger. "Majority voting" is another simple way of combining methods. In the case of three taggers, say, it means that the chosen tag will be the one that is suggested by at least two of the taggers. If no such tag exists, the tag suggested by the preferred tagger will be chosen. It is worth noting that for a majority voting strategy there is no need to run all the taggers for each word. In this case, only two of them must be run for every word and, should they disagree, the tag suggested by third tagger will be chosen. This strategy provided the same result as majority voting with preference given to the third tagger. It is clear that the selection of the preferred tagger will not greatly influence the final outcome. Hence, when applying majority voting in a real world tagger, other aspects like speed can determine the tagger preference. According to the test results in [68], the combinations outperformed their component taggers in almost every case. However, in the various test sets, different combinations proved the best, so no conclusion could be drawn about the best combination. The combined tagger that performed best on the largest test set is shown in row 7 of Table 7.2.

7.3.2 Combination strategies of the TBL tagger

In this chapter we are concerned with the combination of more TBL algorithms. TBL belongs to the group of learners that generates abstract information, i.e. only the class label of the source instance. Although it is possible to transcribe the output format to the confidence type, this limitation reduces the range of the applicable combination schemes. "Min", "Max" and " Prod" rules cannot produce a competitive classifier ensemble, while "Sum" Rule and "Borda Count" are equivalent to "Majority voting".

To build better combination strategies we need to investigate how to adapt the more sophisticated adaptive techniques like "Boosting" to POS tagging applications.

7.3.3 Context-dependent Boosting

The task of the tagger is to select the proper Part of Speech code for a given word in a sentence. The Boosting algorithm is based on weighting the independent instances

based on the classification error of the previously trained learners. It builds classifiers at each iteration that work well on the data instances with high weights, while the instances with lower weights have less importance in the learning process. Most of the algorithms like TBL cannot handle data instance weights directly. In such cases Boosting creates bootstrap versions of the database, where the instances are drawn randomly with replacement according to distribution determined from the weights. Training the classifier algorithm on the bootstrapping databases simulates the weighted training, but this strategy cannot be applied to context-dependent applications like POS tagging. Here the words of the corpus are not independent instances because their context and the position in the sentence both affect their meaning.

In this thesis we propose a general solution for these kind of applications. We will treat the classifier as a black box that assigns class labels for a sequence of instances, where the sequences are handled independently, but the context of the instances in the sequence can have an effect on the labelling process. For context-free applications the sequences contain just one instances, while in POS-tagging the sequences represent the sentences.

The generalized version of Boosting assigns weights to the sequences instead of the instances, and creates bootstrap samples by drawing sequences from the original datasets. The classification error of the sequences can then be calculated from the errors of the instances in the sequence. In the current implementation we use the arithmetic mean. Despite this, the combined final error is expressed as the relative number of misclassified instances.

Algorithm 4 Generalized context-dependent Boosting algorithm

Require: Training Set S, sequence number m, Inducer $\mathcal{I}$
Ensure: Combined classifier C^*
  $S' = S$ with weights set to be $1/m$
  for $i = 1 \dots T$ **do**
    $S' =$ bootstrap sample of sequences from S
    $C_i = \mathcal{I}(\mathcal{S}')$
    $\epsilon_i = \sum_{\mathbf{x_j} \in S' : C_i(\mathbf{x_j}) \neq \omega_j}$ weight of $\mathbf{x_j}$
    if $\epsilon_i > 1/2$ **then** Exit
    $\beta_i = \frac{\epsilon_i}{(1-\epsilon_i)}$
    for all $\mathbf{x_j} \in S'$ such $C_i(\mathbf{x_j}) = \omega_j$ **do**
      weight of $\mathbf{x_j}$ = weight of $\mathbf{x_j} \cdot \beta_i$
    end for
    normalize weights of sequences to sum to 1
  end for
  $C^*(\mathbf{x}) = \underset{j}{\operatorname{argmax}} \sum_{i:C_i(\mathbf{x})=\omega_j} \log \frac{1}{\beta_i}$

7.4 Experimental results

The training and test data sets utilized here are part of the Szeged Corpus [3]. This corpus contains text samples taken from the following six domains: fiction, school compositions, newspaper articles, computer-related texts, laws and short business news. Out of these, business news texts have been separated and were used as a dataset for domain-specific tests. The rest of the corpus was treated as a dataset for tests on a general text. We will call this latter part of the corpus 'the complete corpus'.

The grammatical structure and the vocabulary of business news are rather different from texts of other domains. Hence we expect business news to be representative of domain-specific corpora in cross-domain tests (in tests where the training and the test data come from different domains).

The Szeged Corpus was annotated manually with morpho-syntactic codes from the Hungarian version of the MSD (Morpho-Syntactic Description) code set. Throughout this investigation, all tests were executed with the full MSD tag set. The complete tag set contains several thousand possible codes, out of which 1113 tags appear in the corpus.

Because the MSD tag set we used supports a fine distinction between different parts-of-speech, the ratio of ambiguous words in all tokens is higher than usual: 43.4%. The corresponding ratio is 27.7% and 10.0% for the Hungarian National Corpus and the TELRI corpus, respectively [109, 47].

The training and testing datasets were chosen from the business news domain of the Szeged Corpus. The train database contained about 128,000 annotated words in 5700 sentences. The remaining part of the domain, the test database, has 13,300 words in 600 sentences.

The results of the training and testing error rates are shown in Figure 7.1. The classification error of the standalone TBL algorithm on the test dataset was 1.74%. Boosting is capable of reducing it to below 1.31%, which means a 24.7% relative error reduction. As the graphs show, boosting achieves this during the first 20 iterations, so further processing steps cannot make much difference to the classification accuracy. It can also be seen that the training error does not converge to a zero-error level. This behaviour is due to the fact that the learner cannot maintain the 50% weighted error limit condition. Bagging achieved only a moderate gain in accuracy, its relative error reduction rate being 18%.

7.5 Summary

In this chapter we investigated the conventional classification strategies of POS tagger applications. Simple, static combinations have become popular tools of research in the last decade. However, the advanced adaptive techniques cannot be applied directly. The author designed and implemented a new combination method, a context-dependent variant of the Boosting algorithm, and compared the measured overall tagging scores

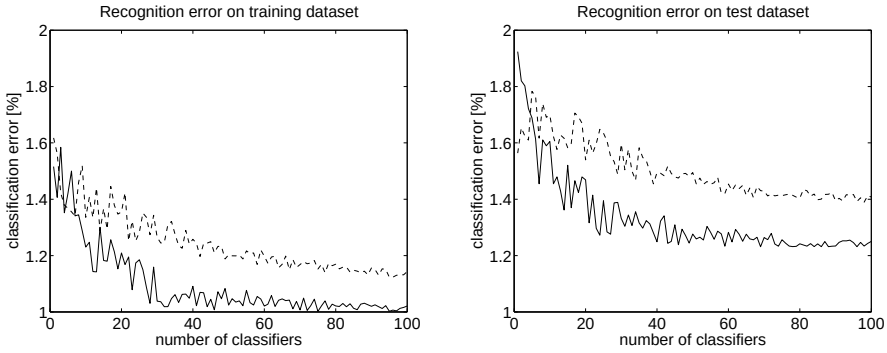

Figure 7.1: Classification error of *Bagging* and *Boosting* algorithm on the training and training datatsets (dashed: Bagging, solid: Boosting).

with the results of the standalone parser and the traditional combination systems. The results indicate that the proposed algorithm can reduce the classification error of the TBL tagger by 24.7%, thus it can be applied in POS tagger systems that require very high accuracies.

8

NP Parsing

Syntactic parsing is the process of determining whether sequences of words can be grouped together. It is an important part of the field of natural language processing and it is useful for supporting a number of large-scale applications including information extraction, information retrieval, named entity identification, and a variety of text mining applications.

In this chapter we will examine the NP parsing of Hungarian texts, describe a parser algorithm called PGS, and overview the combination structures that are able to improve the parsing quality. The chapter is organized as follows. Section 8.1 summarizes the related works on syntactic parsing, then the properties of Hungarian NP parsing are described in sections 8.2 and 8.3. Section 8.4 presents the method used for learning grammar from an annotated corpus, followed by Section 8.5, which investigates the available combination strategies, and offers a Boosting variant method for this problem. In the last section we describe how the proposed methods were tested, then we briefly discuss the results of our experiments.

8.1 Related Works

Several authors published results of syntactic parsing especially designed for English. Generally the performance is measured with three scores: precision, recall and an $F_{\beta=1}$ rate which is equal to $2\frac{precision \cdot recall}{precision+recall}$. The latter rate has been used as the target for optimization. Ramshaw and Marcus [88], for instance, built a chunker by applying transformation-based learning ($F_{\beta=1}$=92.0). They applied their method to two segments of the Penn Treebank [70] and these are still being used as benchmark data sets. Tjong Kim Sang and Veenstra [102] introduced cascaded chunking ($F_{\beta=1}$=92.37). These novel approaches attain good accuracies using a system combination. Tjong Kim Sang [103] utilized a combination of five classifiers for syntactic parsing ($F_{\beta=1}$=93.26).

8.2 Hungarian NP Parsing

Hungarian is an agglutinative, free word order language with a rich morphology. These properties make its full analysis difficult compared to Indo-European languages. Unambiguous marks for the automatic recognition of NP boundaries do not exist. The right bound of NPs can be the head part of the NP, but in some cases the NP head can replace its positions with its modifiers. Determining the left bound of NPs can be more difficult, because it could be a determinant element, and, due to the possibility of a recursive insertion, it is not easy to decide which determinant and NP head belong together. Some of these difficulties are illustrated in the following:

- Free word order:
  ($_{NP}$ Péter) olvas ($_{NP}$ egy könvvet): ($_{NP}$Peter) is reading ($_{NP}$ a book).
  ($_{NP}$ Egy könyvet) olvas ($_{NP}$ Péter) : ($_{NP}$Peter) is reading ($_{NP}$ a book).

- Missing determiners:
  ($_{NP}$ Péter) olvas ($_{NP}$ egy könyvet): ($_{NP}$Peter) is reading ($_{NP}$ a book).
  ($_{NP}$ Péter) ($_{NP}$ könyvet) olvas: ($_{NP}$Peter) is reading ($_{NP}$ a book).

- Missing NP head:
  ($_{NP}$ Péter) ($_{NP}$ a sárga könyvet) olvassa, ($_{NP}$ Mari) pedig ($_{NP}$ a pirosat):
  ($_{NP}$ Peter) is reading ($_{NP}$ the yellow book) and ($_{NP}$Mary) ($_{NP}$ the red one).
  where: ($_{NP}$ a pirosat) = ($_{NP}$ a piros könyvet)

Up till now there has been no good-quality syntactic parser available for the Hungarian language. Benchmark data sets for correctly comparing results on Hungarian do not exist yet either. The HuMorESK syntactic parser [58] developed by MorphoLogic Ltd uses attribute grammar, assigning feature structures to symbols. The grammar part employed in the parser was made by linguistic experts. Another report on the ongoing work of a Hungarian noun phrase recognition parser [116] is based on an idea of Abney's [1] using a cascaded regular grammar and it has been tested on a short text of annotated sentences ($F_{\beta=1}$=58.78). The idea of using cascaded grammars seems beneficial, this technique being used in all Hungarian parser developments. A noun phrase recognition parser [45] is applied machine learning methods to produce grammar of noun phrase tree patterns from an annotated corpus ($F_{\beta=1}$=83.11).

8.3 The Training Corpus

Initially, corpus words were morpho-syntactically analysed with the help of the HuMor3 automatic preprocessor and then manually POS tagged by linguistic experts. The Hungarian version of the internationally acknowledged MSD (Morpho-Syntactic Description) scheme [27] was used for the encoding of the words. Due to the fact that the MSD encoding scheme is extremely detailed (one label can store morphological information on up to 17 positions), there is a large number of ambiguous cases,

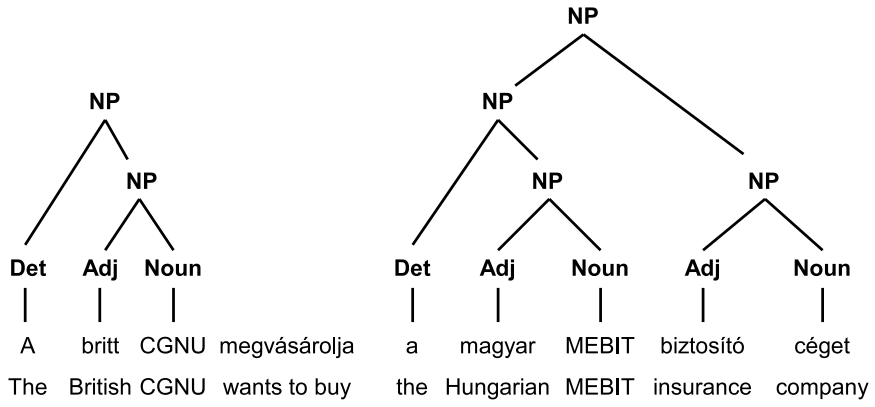

Figure 8.1: A parsing tree of a Hungarian sentence (with its English equivalent) from the Szeged Corpus "Short Business News"

i.e. roughly every second word of the corpus is ambiguous. Disambiguation therefore required accurate and detailed work. It required 64 person-months of manual annotation. Currently all possible labels as well as the selected ones are stored in the corpus. About 1800 different MSD labels are used in the annotated corpus. The MSD label corresponds to the partof- speech determined attribute, and specific characters in each position indicate the value for that attribute. For example, the MSD label "Nc-pa" specifies the following properties:

POS:	Noun,
Type:	common,
Gender:	- (not applicable to Hungarian),
Number:	plural,
Case:	accusative.

All the texts of Szeged Corpus were parsed manually, that is annotators marked various phrase structures. The extensive and accurate manual annotation of the texts, which required 124 person-months of manual work, is a good feature of the corpus. The syntax trees of annotated sentences contain various type of phrases, shown by following list:

Noun phrase (NP)	Verb prefix (PREVERB)
Adjective phrase (ADJP)	Conjunction (C)
Adverb phrase (ADVP)	Pronoun phrase (PP)
Verb phrase (VP)	Clause (CP)
Infinitive(INF)	Sentence (S)
Negative (NEG)	

In general, the NP building process of a sentence produces detailed NP trees much like Figure 8.1. These general NP trees must be simplified because, of course, simpler trees more readily support information extraction. This simplification was done by linguistic experts in the manual annotation phase.

8.4 Learning tree patterns

In this section the learning task of syntactic tree patterns will be described which contains the preprocessing of training data, generalization and specialization of tree patterns. An improved version of RGLearn [45] named PGS (Pattern Generalization and Specialization) was used as a tree pattern learner.

8.4.1 Preprocessing of training data

The initial step for generating training data is to collect syntactic tree patterns from an annotated training corpus. The complete syntax tree of sentence must be divided into separate trees and cascade tree building rules to prepare the parser to reconstruct it. In parsing, using context free grammar has a lot of advantages, but the conditions of pattern usage may completely disappear. Some structural information can be salvaged if tree patterns are used. To generate cascaded grammar, linguistic experts have defined the following processing levels for the Hungarian language:

- Short tree patterns of noun, adjective, adverb and pronoun phrases.
- Recursive complex patterns of noun, adjective, adverb and pronoun phrases.
- Recursive patterns of verb phrases.
- Recursive patterns of sub-sentences.

8.4.2 Generalization of tree patterns

Using the collected tree patterns the syntactic parser is able to reconstruct the tree structure of training sentences. But, in order to perform the syntactic parsing of an unknown text to a fair accuracy, the collected tree patterns must be generalized. Generalization means that the lexical attributes of each tag are neglected except for the POS codes. In this phase the learning problem is transformed into a classification problem. Namely, which set of lexical attributes would supply the best result for the decision problem of tree pattern matching, i.e whether a given tree structure covers a given example or not. To support the learner, positive and negative examples are collected from a training set for each tree type. The example in Figure 8.2 shows the complete tree pattern learning process.

8.4.3 Specialization of tree patterns

Negative examples are the bad classifications of generalized tree pattern and they must be eliminated. Therefore specialization selects each possible lexical attribute from positive examples by making new tree patterns and tries to find the best tree patterns with unification.

The initial step of specialization generates all possible new tree patterns by extending generalized tree patterns with exactly one attribute from the covered positive examples. The next step of specialization extends the set of tree patterns with all possible new tree patterns by a combination of each pair of tree patterns. The combination of two tree

Sentence parts (examples):
1: . . . $(_{NP}\ Tf(_{AdjP}\ Afp-sn)Np-sn)$ . . .
2: . . . $(_{NP}\ Tf(_{NP}\ Afp-pn)\ Nc-pa---s3)$ . . .
3: . . . $(_{NP}\ (_{NP}\ Ti(_{NP}\ Afs-sn))(_{NP}/Nc-s2)$ . . .
4: . . . $(_{NP}\ Tf(_{AdjP}\ Afp-sn)Nc-sn)$ . . .
5: . . . $(_{NP}\ Tf(_{AdjP}\ Afp-sn)(_{AdjP}\ Afp-sn))$ . . .

Generalized pattern (one of four possible): $(_{NP}\ T*(_{AdjP}\ A*)\ N*)$
Coverage: positive {1,4}, negative {2,3}, uncovered {5}
Specialized pattern: $(_{NP}\ T*(_{AdjP}\ A*)\ N???n)$
Coverage: positive {1,4}, negative {}, uncovered {2,3,5}

Notations:
In the lexical codes each letter is a lexical attribute, the first one being the part of speech: $T*$: determiner, $A*$: adjective, $N*$: noun, $N???n$: noun with a lexical attribute, ?: a letter of any kind, $*$: one or more letters of any kind.
The noun and adjective phrases are represented by $(_{NP}$ ...) and $(_{AdjP}$ ...), respectively

Figure 8.2: A tree pattern learning example.

patterns means the union of their lexical attributes. To avoid the exponential growth of a tree pattern set, weak tree patterns are excluded by applying error statistics on positive and negative examples. Here the following score of a given tree pattern is used as the target for maximization:

$$\text{score} = \lambda_1 * (pos\text{-}neg)/pos + \lambda_2 * (pos\text{-}neg)/(pos+neg),$$

where pos is the number of covered positive examples, neg is the number of covered negative examples and $\lambda_1 + \lambda_2 = 1$.

Fruitful unifications dramatically decrease the negative coverage, resulting in a positive coverage almost at the same time. The score maximization runs parallel on every positive example. A new tree pattern is stored only if a covered positive example exists where the score of new tree pattern is greater than the previous maximum value. Specialization stops when the current step did not improve any maximum value.

An appropriate setting of λ factors in linear combination can provide the optimal tree pattern set. A greater λ_1 may result in tree patterns with high coverage, while a greater λ_2 may result in a high accuracy but there is a possibility of low tree patterns appearing with a low coverage.

8.4.4 Generating Probalistic Grammar Rules

The syntax tree of a sentence can generally be derived in different ways, especially when using a large grammar. The choice of a best derivation from the derivation forest requires additional information. One of the possible ways of doing this is a making of PCFG (Probability Context Free Grammar). Using a PCFG, the probability of a derivation is the product of probabilities of the applied rules, but the product may be

replaced by other operators, e.g. a max operator. After evaluating each derivation, the derivation with a higher probability will selected. The conditional probability of a rule is computed with the following formula of Maximum Likelihood Estimation:

$$P(T \rightarrow \beta|T) = \frac{Count(T \rightarrow \beta)}{Count(T)} \tag{8.1}$$

The formula shows that the conversion of a CFG to a PCFG is based on rule application statistics on the training data, namely, counting the proper coverage of each rule and counting the node labels in the annotated syntax trees. This method is applicable for tree patterns as well. Counting the coverage of tree patterns is the same as counting coverage of one level rules.

8.4.5 Syntactic parsing

The main task of the syntactic parser is to find the most likely parse tree for input sentences. Input data contains disambiguated POS tag labels and it may come from the Szeged Corpus during the testing phase of the method, or it may be provided by a POS tagger tool [66] as a practical application of the method. There are natural language processing modules used to determine various linguistic features for syntactic parsing when the process starts with plain text: tokenization, sentence segmentation, morpho-syntactic analysis and part-of-speech (POS) tagging.

Parsing with a PCFG can be done in polynomial time – but disambiguation – namely the selection of best possible parse tree from the parse forest, is an NP-hard problem. Sima'an [99] demonstrated that there is no deterministic polynomial time algorithm for finding the most probable parse of a sentence. Owing to this fact, it is not efficient if the parser determines all possible derivations with probabilities, and then selects the best one. Hence, the Viterbi algorithm [118] is applied in our parser to find the most probable derivation, because it can perform it in cubic time. The basic idea behind the Viterbi approach is the elimination of low probability subderivations in a bottom-up fashion. Two different subderivations for the same part of sentence and with the same root can both be included by derivations in the same way. Thus, if one of these two subderivations has a lower probability, it will be eliminated.

The Viterbi elimination method is applicable for higher levels in bottom-up parsing and finally for the selection among S roots of the derivation forest. There are a lot of low probability subderivations that are pruned through parsing to speed up the process.

8.5 Combination Strategies

To build better NP parser systems, a number of combination strategies have been proposed in the literature. Tjong Kim Sang and Veenstra [102] for instance introduced cascaded chunking ($F_{\beta=1}$=92.37). These novel approaches attain good accuracies using combination systems. Utilizing a combination of five classifiers for syntactic parsing,

Parser	$F_{\beta=1}$
Standalone parser	78.5
Prod Rule	79.4
Max Rule	77.5
Min Rule	78.7
Sum Rule	82.4
Borda Count	81.9
Bagging	83.6
Boosting	86.2

Table 8.1: Results achieved by combining RGLearn parsers

Tjong Kim Sang [103] got a slightly better performance ($F_{\beta=1}$=93.26).

In the following we shall investigate the combination strategies of the RGLearn parser. Since this parser algorithm provides confidence information type, the majority of the combination schemes discussed in the previous chapters can be employed. The overall syntax parser generates all the possible syntax tree for the input test, queries all the the parser instances for the scores on these tree structures, then based on the scores it selects the syntax tree with the best overall combined scores.

This framework allows the static combination techniques like "Prod", "Sum", "Max", "Min" and "Borda Count" to be integrated into the parser system. In the case of adaptively trainable combiner methods like "Bagging" and "Boosting", the algorithm generates several parser instances that should concentrate on instances that behave badly in previous iterations. Treating sentences as data instances, Boosting forces the parsers to focus on the tree structures in problematic sententes, and then it creates better parsing systems. Since the RGLearn algorithm can handle instance weights, this combination strategy can be applied in our NP parsing applications as well.

8.6 Experiments

The training and test datasets were converted from a subset of the business news domain of the Szeged Corpus. During the experiments we generated 50 learners by training the PGS algorithm on different training sets, these sets being created by randomly drawing 4000 instances with replacement from the original training set. The λ_1 parameter of the PGS algorithm was selected for optimal performance on the original train dataset. From our preliminary investigations we found that the PGS algorithm attains its maximal recognition accuracy when λ_1 is set to 0.7, hence this setting was used during the combination experiments.

8.6.1 Results

The syntactic tree pattern recognition accuracy of the standalone RGLearn parser was $F_{\beta=1}$=78.5 on the business-news domain using 10-fold cross-validation.

Based on their performance, the combination schemas can be divided into 3 groups. The schemas Max, Min, and Prod have roughly the same performance: they cannot significantly improve the classification accuracy of the PGS learner. The Borda Count and Sum rule can take advantage of combinations, and produce an $F_{\beta=1}$=82 score on the test data-set. The best classification was obtained by using the Boosting algorithm, achieving the value $F_{\beta=1}$=86. Note that the context-dependent Adaboost algorithm cannot reduce the training error rate to zero owing to the fact that the Boosting algorithm requires that the weighted error should be below 50%, and this condition is not always fulfilled. Comparing these results with the performance of the standalone learner, we see that combinations can improve the classification scores by some 10%.

8.7 Summary

In this chapter we examined the NP parsing of Hungarian texts. A general machine learning method called GPS was described and evaluated on the Szeged Corpus. To obtain a better parsing efficiency the author examined several combination strategies and implemented a Boosting-variant algorithm. As the results showed, the accuracy of tree pattern recognition was effectively improved using the combination schemas, the best performance being achieved by the proposed Boosting algorithm.

9

Conclusions

In this thesis, we covered various aspects of classifier combination techniques, from both the theoretical and practical perspectives. In particular, we studied classifier combination techniques and showed how they could be applied with success to speech recognition and natural language processing tasks.

In the last few decades, multiple classifier systems have become a powerful tool for machine learning. Combination algorithms, especially the boosting methods, have proved useful in improving classification scores. However, despite their success in theoretical investigations, applying the boosting algorithms is not always efficient in practice. In cases like speech recogition applications in mobile devices, simpler combination strategies should be applied. The proposed strategy, based on the Analytic Hierarchy Process, can be an alternative solution for these problems, providing a better performance than that for the averaging methods.

As shown in this thesis, applications in Human Laguage Technologies can exploit the benefits of classification improvements of the combination techniques. In the evaluation tests on our speech recognizer system called "OASIS", we investigated the efficiency of the combination strategies of phoneme classifiers. Recognizing the importance of phoneme classification, we examined another promising method for this task, namely the Vocal Tract Length Normalization procedure. We found that combination strategies can be applied successfully both for adapting the model for the speakers in real-time and for improving the overall phoneme classification. Following our successful results, these structures were integrated into our award-winning Phonological Awareness Teaching System "SpeechMaster".

Other tasks of Human Language Processing, like POS tagging and NP parsing, are also known to be sensitive to the performance of the machine learning method applied. To achieve better results, simple static combination techniques are available in the literature. But due to the context dependence of these applications, more advanced boosting methods cannot be used directly. The proposed context-dependent versions of the boosting algorithm offer a solution for these problems, and the results demonstrate

that they can undoubtedly improve the parsing scores in the given Hungarian POS tagging and NP parsing applications.

APPENDIX A

Databases Used in the Dissertation

A.1 The OASIS-Numbers Database

The OASIS-Numbers database consists of spoken Hungarian numbers. It was collected at the Research Group on Artificial Intelligence of the Hungarian Academy of Sciences within the framework of the SZT-IS-10 national grant. Thanks to the governmental support, the database is freely accessible to everyone. The recordings of the corpus are of reasonably good quality, having been recorded with several types of microphones at a sampling rate of 22050 Hz in 16-bit quality. The speakers of the database are mostly university students – 62 males and 49 females.

The utterances recorded can be grouped into two main categories. One of them contains the so-called base words. These correspond to 26 words that are selected so that from them all the Hungarian numbers between 0 and 1,000,000 can be constructed. All the base word recordings of the corpus are manually segmented and labelled at the phone level. Altogether 28 different phonemic labels occur in these transcripts.

The other group of recordings contain randomly chosen numbers between 0 and 1,000,000; these files are intended to be used for testing.

In the selection of the the train and test utterances we followed the recommendation of the database documentation. Thus, 2185 base word recordings were used for training and 1247 random utterances for testing purposes, respectively. For the test utterances we applied the pronunciation dictionaries given with the database. The phonetic transcripts for the compound numbers were simply generated by concatenating the transcripts of the proper base words.

A.2 The MTBA Hungarian Telephone Speech Database

The MTBA Hungarian Telephone Speech Database is the result of an IKTA project carried out in 2001-2003 by the Department of Informatics, University of Szeged, and the Department of Telecommunications and Media Informatics, Technical University of

Budapest. The MTBA Hungarian Telephone Speech Database is the first Hungarian speech corpus that is publicly available and has a reasonably large size. Besides several groups of recordings that contain isolated words (numbers, company names, city names, etc.), the database contains 6000 sentences recorded from 500 speakers (12 sentences from each). These sentences are relatively long (40-50 phones per sentence) and were selected so that their phonetic transcripts contains evey possible phone connection that occurs in Hungarian. Recordings were made via both mobile and line phones, and the phone calls were organized so that the recordings covered the whole area of the country. The speakers were chosen so that their distribution corresponded to the age and gender distribution of the Hungarian population. All the sentences were manually segmented and labelled at the phone level. A set of 58 phonetic symbols was used for this puprose, but after fusing certain rarely occurring allophones, we worked with only 52 phone classes in the experiments.

For the selection of training and test utterances, we first removed those sentences from the database that contained significant noise and/or half-cut phones (denoted by [spk] and [cut] symbols in the phonetic transcript). From the remaining sentences 1367 were randomly chosen for training purposes (containing 68333 phone instances). For phone recognition tests we used another set of 687 sentences (containing 34532 phone instances). The word recognition results reported in the dissertation are isolated word recognition tests performed on another block of the database that contained city names. All the 500 city names (each pronounced by a different caller) were different. Of the 500 recordings only 431 were employed in the tests as the rest contained significant non-stationary noise or were misread by the caller. The language model employed in the word recognition tests was a simple pronunciation dictionary (created by an automatic phonetic transcription routine) that contained one phonetic transcript for each word and assumed that each of them had equal priors.

In certain experiments reported in this dissertation several parameters will be fine-tuned on the city name recordings. In these cases further testing is required on an independent data block. For this purpose we chose an additional group of 438 recordings from the database, again containing city names, but this time over a smaller vocabulary.

More details about the construction and contents of the MTBA database can be found in [117] (in Hungarian).

A.3 The BeMe-Children Database

The BeMe-Children database was collected as part of an IKTA project carried out in 2002-2004 by the Department of Informatics, University of Szeged, the Gyula Juhász Teacher Training School of the University of Szeged and the School for the Hearing Impaired in Kaposvár. The goal of the project was the construction of the "SpeechMaster" software package for speech therapy and teaching reading, and the BeMe-Children corpus was originally recorded for the purpose of training and testing the software. The corpus contains recordings from 500 children from the lower classes of elementary schools and from a further 200 pupils with various levels of hearing impairment. In the

experiments reported in this dissertation just the former block of data was used, so we give details only on these recordings.

The database contains samples of 100 words from each of the 500 children. From these 40 words were the same in every case and the remaining 60 words varied from speaker to speaker. Only the latter recordings were made use of in the experiments. To construct this data set the most frequent 2000 words were collected from 14 teaching reading books that are currently used in elementary schools. These 2000 words were distributed in the 500*60 recordings according to their frequency in the books, that is the more frequent words occur in more recordings. The recordings were collected in 14 schools all around the country from children of age 6-7, from 250 boys and 250 girls. The database is phonetically segmented and labelled.

For the experiments presented in this dissertation 4000/920 utterances were selected for training/testing purposes, respectively. For language modelling the phonetic transcripts of the 2000 words were created automatically. Owing to the high variability in the children's voices and the recording conditions, and because of the many similar-sounding words in the dictionary, this recognition task appeared to be quite difficult.

More details about the construction and contents of the BeMe-Children database can be found in [97]. The "SpeechMaster" software is described in [4] (both papers are in Hungarian).

A.4 The Szeged Corpus

The Szeged Corpus is a manually annotated natural language corpus currently comprising 1.2 million word entries, 145 thousand different word forms, and an additional 225 thousand punctuation marks. With this, it is the largest manually processed Hungarian textual database that serves as a reference material for research in natural language processing as well as a learning database for machine learning algorithms and other software applications. Language processing of the corpus texts so far included morpho-syntactic analysis, POS tagging and shallow syntactic parsing. Semantic information was also added to a preselected section of the corpus to support automated information extraction.

A.4.1 Text of Szeged Corpus

selecting texts for the Szeged Corpus, the main criteria was that they should be thematically representative of different text types. The first version of the corpus, therefore, contains texts from five genres, roughly 200 thousand words each. Due to its relative variability, it serves as a good reference material for natural language research applications, and proves to be large enough to guarantee the robustness of machine learning methods. Genres of Szeged Corpus 1.0 include:

- fiction (two Hungarian novels and the Hungarian translation of Orwell's 1984)
- compositions of 14-16-year-old students

- newspaper articles (excerpts from three daily and one weekly paper)
- computer-related texts (excerpts from a Windows 2000 manual book and some issues of the ComputerWorld, Számítástechnika magazine)
- law (excerpts from legal texts on economic enterprises and authors' rights).

During further developments, the first version of the corpus was extended with a 200 thousandword- long sample of short business new2. The newly added section served as an experimental database for learning semantic frame mapping to be later integrated in an IE technology. Table 1. shows data referring to Szeged Corpus 2.0.

A.4.2 Annotation of the Szeged Corpus

Morpho-syntactic analysis and POS tagging of the corpus texts included two steps. Initially, words were morpho-syntactically analysed with the help of the Humor [84] automatic pre-processor. The program determined the possible morpho-syntactic labels of the lexicon entries, thereby creating the ambiguous version of the corpus. After the preprocessing, the entire corpus was manually disambiguated (POS tagged) by linguists. For the tagging of the Szeged Corpus, the Hungarian version of the internationally acknowledged MSD (Morpho-Syntactic Description) scheme (Erjavec, Monachini, 1997) was selected. Due to the fact that the MSD encoding scheme is extremely detailed and refined (one label can store information on up to 17 positions), there is a large number of ambiguous cases, i.e. one word is likely to have more than one possible labels. Experiences show that by applying the MSD encoding scheme, roughly every second word of the corpus is ambiguous. Disambiguation, therefore, required accurate and detailed work cumulating up to 64 person-months of manual annotation. Currently all possible labels as well as the selected ones are stored in the corpus. A unique feature of the corpus is that parallel to POS tagging, users' rules have been defined for each ambiguous word in a pre-selected (202 600- word-long) section of the corpus. The aim of applying users' rules was to mark the relevant context (relevant set of words) that determines the selection of a certain POS tag. Users' rules apply $before_1$, $before_2$, ..., $after_1$, $after_2$, ... predicates for marking the relevant context of a word. The manually defined rules can then be generalised to regular disambiguation rules applicable to unknown texts as well. Out of the selected 202 600 words 114 951 were ambiguous. Annotators defined users' rules for these cases among which 26 912 different ones were found. The major advantage of the defined rules lies in their accuracy and specificity, wherefore they are an interesting and valuable source of additional linguistic information that can e.g. support the more precise training of machine learning algorithms. After the completion of POS tagging, a project was initiated to encompass shallow syntactic parsing of the Szeged Corpus. The linguistic information identified by shallow syntactic parsing proves to be rich enough to support a number of large-scale NLP applications including information extraction (IE), text summarisation, machine translation, phrase identification in information retrieval, named entity identification, and a variety of text-mining operations. In order to achieve their goal, researchers of the

University of Szeged, Department of Informatics, the MorphoLogic Ltd. Budapest, and the Research Institute for Linguistics at the Hungarian Academy of Sciences had to conduct some research concerning the syntax of Hungarian sentences, NP annotation schemes, and rules covering the recognition of phrases. Results showed that in Hungarian, nominal structures typically bear the most significant meaning (semantic content) within a sentence, therefore NP annotation seemed to be the most reasonable step forward. Shallow parsing was carried out on the entire Szeged Corpus 2.0 (1.2 million words). Automated pre-parsing was completed with the help of the CLaRK program, in which regular syntactic rules have been defined by linguistic experts for the recognition of NPs. Due to the fact that the CLaRK parser did not fully cover the occurring NP structures (its coverage was around 70%), manual validation and correction could not be avoided. In total, 250 thousand highest level NPs were found, and the deepest NP structure contained 9 NPs embedded into each other. The majority of the hierarchic NP structures were between 1 to 3 NPs deep. Manual validation and correction lasted 60 person-months. As a continuation of shallow parsing, the clause structure (CPs) of the corpus sentences was also marked. Labelling clauses followed the same approach as earlier phases of NLP: it comprised an automatic pre-annotation followed by manual correction and supplementation.

APPENDIX B

An Example OASIS Script

The following code is a typical configuration script of the OASIS Speech Lab System. Lines start with "//" contain comments for the easier understanding.

```
// creating a window for displaying the results
sys "win = new Window()";
mod {
  // listing the elements of the work flow graph a boolean variable
  // controlling when we are going to train or test
  train = 0;

  // reading the phone symbol table from a file
  ph   = new SimplePhonemes(root.mnt.data.'phonemes.gr');

  // loading the pronunciation dictionary
  dict = new Dictionary(root.mnt.data.'dict.txt', ph);

  // this module goes through the elements of the file list
  // one by one
  dt = new DatTraverse(root.mnt.'filelist.txt');

  // reading the wave file obtained from the DatTraverse module
  wfi = new WavFileIn(dt);

  // the following modules calculate the MFCC coefficients along
  // with their 'delta' and 'delta-delta' values; the processing
  // steps are: preemphasis - Fourier spectrum - mel filter bank
  // energy estimation - cosine transform - delta and delta-delta
  // coefficient calculation
  wfp  = new PreEmpSB(wfi, 0.97);
  sp   = new Spectrum(wfp, 400, 160, 512, 1);
  fbb  = new FilterBankBA(sp, 26);
```

```
mfcc = new MFCCBA(fbb, 12, 22, 1);
de1  = new DeltaMapBA(mfcc);
de2  = new DeltaMapBA(de1);

// collecting the coefficients into the feature vector fe[0..38]
for i:0..12 fe[i] = new FBand(mfcc, [i]);
for i:0..12 fe[i+13] = new FBand(de1, [i]);
for i:0..12 fe[i+26] = new FBand(de2, [i]);

// extraction of segment-based acoustic cues; here they are
// simply the feature averages over the segment parts divided
// in a 3-7-3 ratio
for i:0..11 a[i] = new ACMean(fe[i], 0.0, 0.3);
for i:0..11 b[i] = new ACMean(fe[i], 0.3, 0.7);
for i:0..11 c[i] = new ACMean(fe[i], 0.7, 1.0);

//a further cue will be the segment duration
acd  = new ACDuration();

// reading in the annotation file belonging to the wave file;
// it contains the orthographic transcript and may also contain
// manual segmentation and labelling info; the former is required
// for testing; the latter are required for training
df   = new DatFile(dt, sp, ph);
if (not train) {
  // a block for testing the recognizer

  // fake clustering by placing a boundary marker at every
  // 2nd frame
  cfall = new FakeClusters(2, sp);

  // loading the parameters of the ANN-based Evaluator and
  // specifying  the segmental features as its input
  anne = new ANNEvaluator("ann.wts", 1,
                          a[0..11], b[0..11], c[0..11], acd);

  // the evaluator results are cached in order to avoid
  // processing the same segment twice
  canne = new EvalCache2(anne);

  // recognition using the Multiple Priority Queue Engine;
  // its input modules are the evaluator, the segmentation,
  // and the dictionary (along with the phonetic symbol table);
```

```
    // the segments are restricted to be at least 3 frames and
    // at most 200 ms long; the size of the stacks is set to 150
    te = new MPQEngine(canne, 0, cfall, dict, ph, 200, 0, 3, 150);

    // the resulting word is compared with the orthographic
    // transcript given in the annotation file; the results are
    // collected in cr
    cr = new CompareResult(te, df);

    // a block that displays the spectrogram, the manual
    // segmentation markers and the segment boundaries of
    // the winning hypothesis
    md   = new MapDisplay(sp, parent.win, "SP", 1, 50, 0, 32767);
    cbd  = new ClusterBoundDisplay(df, parent.win, "CB");
    cbd  = new HypothesisDisplay(te, cfall, parent.win, "HYP");

    // building the graph of the modules and starting processing
    build();
    start();

    // after processing all the files, this module displays the
    // recognition statistics collected by CompareResult
    ? cr
  }
  if (train) {
    // a block that extracts training data from the files
    // this module goes through all the segments given by
    // the manual segmentation, labels them according to the
    // labels given in df, and extracts the segmental features
    // from them; the strategy of how to create anti-phone
    //  examples is specified by the code "162";
    // the data extracted is then saved by StringFileOut
    mkt  = new MKTrain(df, "162", ph,
                       a[0..11], b[0..11], c[0..11], acd);
    sfo = new StringFileOut(mkt, "traindata.data");

    //building the graph of the modules and starting processing
    build();
    start();
  }
}
```

APPENDIX C

Summary in English

For years, the pattern recognition community has focused on developing optimal learning algorithms that can produce very accurate classifiers. However, experience has shown that it is often much easier to find several relatively good classifiers instead of one single very accurate predictor. The advantages of combining classifiers instead of a single classifier are twofold: it helps reducing the computational requirements by using simpler models, and it can improve the classification performance.

Most Human Language Technology applications are based on pattern matching algorithms, thus improving the performance of the pattern classification has a positive effect on the quality of the overall application. Since combination strategies proved very useful in reducing classification errors, these techniques have become very popular tools in applications such as Speech Technology and Natural Language Processing.

This dissertation consists of two main parts. The first part discusses the theoretical aspects of the combination methods, while the second part deals with the applications of these methods to speech technology and natural language processing.

Combination Strategies

Given infinite training data, consistent classifiers approximate the Bayesian decision boundaries to arbitrary precision, therefore providing a similar generalization. However, often only a limited portion of the pattern space is available or observable. Given a finite and noisy data set, different classifiers typically provide different generalizations. It is thus necessary to train several classifiers when dealing with classification problems so as to ensure that a good model or parameter set is found.

Having a set of independent inducers, the simplest way of building a classifier system is to select one with the best behaviour on a given testing database. During the classification just the output of the selected classifier is computed, and only this will affect the resulting decision. This selection is an "*early*" combination scheme widely used in Pattern Recognition. However, selecting such a classifier is not necessarily the ideal choice since potentially valuable information may be wasted by discarding the results of the other classifiers. In order to avoid this kind of loss of information, the output of each available classifier should be examined for making the final decision.

In Chapter 2 we briefly outlined the simple static combination strategies, including "Prod", "Sum", "Min", "Max" and "Borda Count" rules. Although these techniques have become popular in multiple-classifier systems owing to their simplicity, they cannot be adapted to the special aspects of particular applications. For this reason, adaptive methods like additive combination schemes have become the focus of research, these being reviewed in Chapter 3.

Analytic Hierarchy Process

Linear combination schemes, especially Boosting algorithms, are frequently used in machine learning applications due to their ability to improve the classification performance. The Adaboost algorithm and its variants create a sequence of classifier instances by training the same classifier algorithm on special bootstrap samples of the training database. The classification performance of the original classifier method can be dramatically increased, especially in the cases of "weak" classifers, but for the final solution it requires hundreds or thousands of iterations. In the practice, however, most of the applications cannot provide the required huge amount of resources for applying such a great number of classifier instances.

In Chapter 4 we provided a short introduction to Multi-Criteria Decision-Making (MCDM) and its powerfull strategy, the Analytic Hierarchy Process (AHP). Based on this technique the author designed and implemented a novel linear combination method, and then he compared its performance with those of other combiners. As shown in the experiments, AHP-based combinations proved an effective generalization of the Weighted Averaging rule; they outperformed the conventional methods in almost every case.

Speech Technology Applications

Automatic speech recognition is a special pattern classification problem which aims to mimic the perception and processing of speech in humans. For this reason it clearly belongs to the fields of machine learning and artificial intelligence. For historical reasons, however, it is mostly ranked as a sub-field of electrical engineering, with its own unique technologies, conferences and journals. In the last two decades the dominant method for speech recognition has been the hidden Markov modeling approach. Meanwhile, the theory of machine learning has developed considerably and now has a wide variety of learning and classification algorithms for pattern recognition problems.

Phoneme Classification

In this thesis the phoneme classification tests were performed within the framework of the OASIS speech recognizer. The aim of these tests was to study how the application

of the combination methods affects the classification performance applied on a set of standard classification algorithms given in the speech recognition literature.

In Chapter 6 we surveyed the various combination schemes available using speech recognition oriented data-sets. The experimental results showed that making classifier hybrids improved the discrimination performance, the best results being obtained by aggregating the output of SVM, ANN, and kNN. The performance of the combiners applying different decision rules was not significantly different, but the "Sum" rule outperformed the others. Comparing the traditional Bagging and Boosting techniques, we found that they had nearly the same classification improvement, but their applicability was limited because they were too CPU intensive. The findings suggest that it is worth applying combination techniques in phoneme-level speech recognition systems because they will hopefully produce better scores, hence improve the overall results.

Vocal Tract Length Normalization

As we mentioned earlier, the efficiency of a speech recognizer application can be highly dependent on the performance of the phoneme classifier module used. Also, it is just as important for a phonetic teaching system like our Phonological Awareness Teaching System called "Speech-Master". Since the system should work well both for children and adults of different ages, the recognizer has to be trained with the voices of users of both genders and of practically any age. The task is also special because it has to recognize isolated phones, so it cannot rely on language models.

In Chapter 7 we focused on a procedure called Vocal Tract Length Normalization (VTLN), which has proved very useful when the targeted users vary greatly in age and gender. We showed that the on-line parameter estimation methods can be handled as special combination structures and we described the evaulation process. The results demonstrate that using combination stategies of the on-line methods, the overall system can achieve nearly the the same recognition quality as that with the off-line normalization version, while applying Bagging and Boosting may produce classifiers with better performances than those for LD-VTLN. With these positive results the implemented module was integrated into the award-winning Phonological Awareness Teaching System called "Speech-Master".

POS Tagging

Part-of-speech tagging (POS tagging or POST), also called grammatical tagging, is the process of marking words in a text that correspond to a particular part of speech, based on both their definition and their context, i.e. the relationship between adjacent and related words in a phrase, sentence, or paragraph. A simplified form of this is commonly taught to schoolchildren, in the identification of words as nouns, verbs, adjectives, adverbs and so on. Being a building block of many Human Language Technology applications like NP parsing, a number of strategies have been proposed in the literature to improve its performance.

Chapter 8 gives a brief overview to the Part-Of-Speech tagging, To improve the performance of the tagger, we introduced a special, context-dependent variant of the popular Boosting algorithm, and demonstrated that this combination strategy effectively reduced the parsing error. The results showed that the proposed algorithm can reduce the classification error of the TBL tagger by 24.7%, thus it can be applied in POS tagger systems that require very high accuracies.

Syntactic Parsing

Syntactic parsing is the process of determining whether sequences of words can be grouped together. It is an important part of the field of natural language processing and it is useful for supporting a number of large-scale applications including information extraction, information retrieval, named entity identification, and a variety of text mining applications.

In Chapter 9 we examined the effect of the combination methods in syntactic parsing. A general machine learning method called GPS was described and evaluated on the Szeged Corpus. To obtain better parsing efficiency the author examined several combination strategies and implemented a Boosting-variant algorithm. As the results show, the accuracy of tree pattern recognition is effectively improved using the combination schemas, the best performance being achieved by the proposed Boosting algorithm.

C.1 Key Points of the Thesis

In the following a thesis-like listing of the most important results of the dissertation is given. Table C.1 shows which thesis is described in which publication by the author.

I.) The author developed a new linear combination strategy based on the Analytic Hierarchy Process,which is able to improve the classification performance using a small number of classifiers. He compared the results of other competing algorithms and demonstrated that in most cases it results in better classification scores than those for the conventional strategies.

II.) The author designed and implemented the kernel parts of the speech recognition framework Oasis System. Using the integrated combination module, he compared the efficiency of various combination techinques in the field of speech technology. The experiments confirm that combining the results of multiple classifier algorithms effectively enhances the quality of phoneme recognition.

III.) The author investigated the application of classifier combinations in Vocal Tract Length Normalization to improve the phoneme recognition performance. The proposed schemas have been integrated into the award-winning Phonological Awareness Teaching System "SpeechMaster".

	[28]	[29]	[81]	[67]	[46]
I.	•				
II.		•	•		
III.			•		
IV.				•	•
V.					•

Table C.1: The relation between the thesis topics and the corresponding publications

IV.) The author developed a novel context-dependent variant of the Adaboost algorithm. Applied on a POS tagging application of Hungarian texts and compared with the existing combinations techinques, he found that the proposed method resulted in a significant improvement in the classification accuracy.

V.) The author examined the efficiency of several combination strategies applied in a Natural Language Processing task called Noun Phrase parsing. To improve the classification performance, he adapted the Boosting algorithm to the special requirements of the problem. In experiments he found that the proposed algorithms significantly enhanced the accuracy of tree pattern recognition.

APPENDIX D

Summary in Hungarian

A mintaillesztéssel kapcsolatos kutatások hőskorában a lehető legjobb tanulóalgoritmusok, minél pontosabb osztályozók keresése állt a középpontban. Az eredmények és a tapasztalatok azonban arra világítanak rá, hogy általában jóval egyszerűbb több, viszonylag jó osztályozót találni, mint egyetlen kimelkedően jót. Egy-egy osztályozó használata helyett azok kombinálása kétszeres előnnyel is jár: egyrészt csökkenti az egyszerű modellek számítási igényét, másrészt javítja az osztályozási teljesítményt.

A legtöbb természetes nyelvi technológiai alkalmazás valamilyen mintaillesztési algoritmuson alapul, így a mintaillesztés hatékonyságának növelése a teljes rendszer működésére is pozitív hatással lehet. Mivel a kombinációs stratégiák sikeresen alkalmazhatóak osztályozóalgoritmusok hibáinak csökkentésére, egyre nagyobb tért hódítanak olyan alkalmazások esetén is mint például a a beszédtechnológia és a természetes nyelvi feldolgozás.

Jelen disszertáció két fő részre tagolódik. Az első rész a kombinációs módszerek elméleti aspektusait taglalja, a második rész pedig ezen módszerek beszédtechnológiai, illetve természetes nyelvi feldolgozásban történő alkalmazását mutatja be.

Kombinációs stratégiák

Végtelen mennyiségű tanító adat felhasználásával minden konzisztens tanulóalgoritmus a Bayes-féle optimális osztályozást állítaná elő. A gyakorlatban azonban mindig csak véges és zajos mintahalmaz áll rendelkezésünkre, így a különböző eljárások szükségszerűen különböző osztályozáshoz vezetnek. Ezért érdemes egy adott probléma megoldása érdekében több különböző algoritmust is betanítani, ekkor nagyobb eséllyel garantálhatjuk, hogy jó modellt, illetve paraméterbeállítást kapunk.

Amennyiben több független osztályozó is rendelkezésünkre áll, a különböző osztályozók által generált különböző eredményeket valamilyen módon figyelembe véve kell döntést hoznunk. A legegyszerűb lehetséges megoldás, hogy a tanítás során a lehetséges osztályozók közül kiválaszjuk azt, amelyik egy adott teszthalmazon a legjobban viselkedik. Ezek után a kiértékelésnél a többi osztályozó eredményét már nem vesszük figyeleme. Ez az igen egyszerű stratégia azonban a legritkább esetben optimális, hiszen értékes információt veszíthetünk a többi osztályozó kimenetének eldobásával. Ez az

információvesztés elkerülhető, ha az összes rendelkezésünkre álló osztályozó algoritmus kimenetét figyelembe vesszük a kombinációs módszerek segítségével.

A második fejezetben röviden áttekintettük az egyszerűbb statikus kombinálási stratégiákat, többek közt a "Prod", "Sum", "Min", "Max" és "Borda Count" szabályt. Habár ezek a kombinálási módszerek épp egyszerűségüknek köszönhetően igen népszerűek a szakirodalomban, nem lehet őket konkrét alkalmazások speciális igényeihez igazítani. Emiatt az adaptív módszerek, mint pl. az additív kombinációs sémák kerültek a kutatások előterébe. A harmadik fejezetben ezeket a módszereket vizsgáltuk meg részletesebben.

Analitikus hierarchikus eljárás

A lineáris kombinációs technikák, különösen a Boosting algoritmus különböző változatai rövid idő alatt igen nagy népszerűségre tettek szert a gépi tanulási alkalmazásokban, mivel számottevően képesek javítani az osztályozás hatékonyságát. Az Adaboost algoritmus osztályozók egy sorozatát állítja elő oly módon, hogy ugyanazt a tanulóalgoritmust az adatbázis különböző, speciálisan mintavételezett változataira alkalmazza. Ezzel a módszerrel – különösen az ún. "gyenge" osztályozók esetében – drasztikus javulást lehet elérni az osztályozó egyszeri tanításához képest, azonban az eljárás több száz vagy akár több ezer iterációs lépést is igényelhet. Gyakorlati alkalmazások esetében legtöbbször nem áll rendelkezésünkre kellő mennyiségű erőforrás ilyen hatalmas mennyiségű osztályozási példa kiértékeléséhez, amely a módszerek használhatóságát erősen korlátozza.

A 4. fejezetben összefoglalja a többtényezős döntések és az analitikus hierarchikus eljárás alapjait. Bevezettünk egy új, az analitikus hierarchikus eljáráson alapuló lineáris kombinálási módszert, majd összehasonlítottuk ennek hatékonyságát más kombinációs megoldásokéval. A futtatások eredményeiből kiderült, hogy az AHP-alapú módszer a súlyozott átlagolási kombinálás egy hatékony általánosítása: a bevezetett eljárás majdnem minden esetben jobb eredményt ért el, mint a hagyományos lineáris módszerek.

Beszédtechnológiai alkalmazások

Az automatikus beszédfelismerés egy speciális alakfelismerési probléma, amely az emberi beszédérzekelés és beszédértés utánzására törekszik. Célkitűzése alapján nyilvánvalóan a gépi tanulás, távolabbról pedig a mesterséges intelligencia témakörébe tartozik, viszont történeti okokból általában a mérnöki tudományok egyik ágaként tartják számon. Az utóbbi évtizedekben a rejtett Markov-modell vált a beszédfelismerés domináns technológiájává, azonban ez idő alatt a gépi tanulás elmélete is sokat fejlődött, és mostanra már a tanuló és osztályozó algoritmusoknak széles palettáját kínálja.

Fonémák osztályozása

A jelen disszertációban ismertetett fonémaosztályozási teszteket az OASIS beszédfelismerési keretrendszer segítségével végeztük el. A tesztek során megvizsgáltuk, hogy a dolgozat elméleti fejezeteiben ismertetett kombinációs módszerek hogyan befolyásolják a beszédfelismerési irodalomból ismert sztenderd osztályozó algoritmusok hatékonyságát. A kísérleti eredmények azt mutatták, hogy a különböző osztályozók kombinálásával az osztályozási pontosság általában javult; a legjobb eredményt akkor kaptuk, amikor az SVM, ANN és k-NN algoritmusok kimeneteit összesítettük. A különböző egyszerű kombinálási sémák teljesítménye között nem mutatkozott szignifikáns különbség, de a "Sum" szabály jobb eredményt szolgáltatott, mint a többi módszer. A Bagging és Boosting algoritmusokat megvizsgálva azt találtuk, hogy nagyjából ugyanolyan pontosságjavulást eredményeznek, de az alkalmazásukat korlátozza, hogy futtatásuk igen erőforrásigényes. Az eredmények alapján végül kijelenthető, hogy a fonémaszintű beszédfelismerés során érdemes kombinációs stratégiákat alkalmazni, mivel általában javítanak az oszályozás pontosságán, s így a beszédfelismerő működésén.

Artikulációs csatorna hossznormalizálása

Mint korábban már említettük, egy beszédfelismerő rendszer pontosságát nagyban meghatározza a fonémafelismerő modul hatékonysága. Méginkább igaz ez a tény a fonetikai tanítási célú alkalmazások eseténben, mint például a "Beszédmester" nevű beszédjavítás-terápiai és olvasásfejlesztő rendszer. Mivel ennek a rendszernek gyermekek és különböző korú felnőttek hangját is megfelelően kell kezelnie, a betanítás során mindkét nemű és gyakorlatilag mindenféle korú beszélőtől kell tanító hangmintát gyűjteni. A feladat azért is speciális, mert izolált fonémákat is fel kell tudnia a rendszernek ismerni, így egy esetleges nyelvi modul segítségére sem támaszkodhat.

A 7. fejezetben bemutattuk az ún. artikulációs csatorna normalizálási technikát (VTLN), amely hasznosnak bizonyult olyan esetekben, ahol a beszélők kora és neme nagyon nagy változatosságot mutat. Megmutattuk, hogy az on-line paraméterbecslő módszerek speciális kombinációs struktúrának tekinthetők, majd a kombinációs módszereket kiértékeltük. Az eredmények azt mutatták, hogy az on-line módszerek kombinációs stratégiáját alkalmazva a rendszer képes megközelíteni ugyanazt a felismerési hatékonyságot, mint az off-line normalizálási algoritmusok, továbbá Bagging és Boosting alkalmazásával az osztályozók teljesítménye képes túlszárnyalni az LD-VTLN módszerét. A kapott pozitív eredményekre alapozva az implementált algoritmusokat beépítettük a "Beszédmester" programunkba.

Szófaji egyértelműsítés

A szófaji egyértelműsítés (POS tagging) során egy szöveg szavaihoz szófaji címkéket rendelünk, figyelembe véve a szó szótárban feltüntetett szófaját és a szó szövegkörnyezetét (azaz az adott szintagmában, mondatban, bekezdésben betöltött szerepét) is. Mivel a

szófaji egyértelműsítés számtalan természetes nyelvi technológiának, mint pl. a nyelvi elemzésnek alapvető építőeleme, így hatékonyságának javítása érdekében több módszer is kidolgozásra került.

A 8. fejezetben röviden áttekintettük a szófaji egyértelműsítés problémáját, és bevezettük a Boosting algoritmus speciális, környezetfüggő változatát. Az eredmények szerint az implementált kombinációs módszer segítségével az illesztési hiba hatékonyan redukálható: a TBL tagger osztályozási hibaarányát 24,7 százalékkal tudtuk csökkenteni.

Szintaktikai elemzés

A szintaktikai elemzés feladata annak megállapítása, hogy adott szöveg szavainak milyen sorozata szintaktikailag összetartozó. Számos természetes nyelvi technologiai alkalmazás megoldásában játszik kulcsszerepet, mint pl. az információkinyerés, a névelemfelismerés, és más szövegbányászati alkalmazások.

A 9. fejezetben mevizsgáltuk a magyar nyelvű nyelvtani elemzés nehézségeit, majd bemutattunk egy szabályalapú gépi tanulási algoritmust, amely famintákat keres adott szövegben. Az algoritmus hatékonyságának növelése érdekében többféle kombinációs stratégiát is kiértékeltünk, és megállapítottuk, hogy a faminták felismerése számottevően javítható bizonyos kombinációs stratégiák használatával, a legjobb eredményt az adaptált Boosting algoritmus nyújtotta.

D.1. Az eredmények tézisszerű összefoglalása

Az alábbiakban öt tézispontba rendezve összegezzük a Szerző kutatási eredményeit. A kutatásokból származó publikációkat, valamint azok tartalmának az egyes tézispontokhoz való viszonyát a C.1 táblázat tekinti át.

I.) A Szerző kidolgozott egy újszerű, az AHP módszerre épülő lineáris kombinációs stratégiát, amely képes akár kis számú osztályozó használásával is hatékonyan növelni az osztályozási pontosságot. A Szerző összevetette módszerét más, hasonló célú ismert módszerekkel, és megmutatta, hogy a javasolt tecnhika az esetek többségében jobb osztályozási pontosságot eredményez, mint a hagyományos stratégiák.

II.) A Szerző megtervezte és implementálta az OASIS beszédfelismerő keretrendszer alapvető részeit. Az integrált kombinációs modult felhasználva kiértékelte számos kombinációs technika hatékonyságát. A kísérletek igazolták, hogy többféle osztályozóalgoritmus kombinálásával a fonémafelismerés pontossága számottevő mértékben javítható.

III.) A Szerző megvizsgálta a kombinációs stragégiák hatását az artikulációs csatorna hossznormalizáció során alkalmazott módszereken. Az eredmények azt mutatták, hogy az on-line módszerek kombinációjával a rendszer képes elérni akár az

off-line módszerek hatékonyságát is. A javasolt módszerek beépítésre kerültek a "Beszédmester" beszédjavítás-terápiai és olvasásfejlesztő rendszerbe.

IV.) A Szerző kifejlesztette az Adaboost algoritmus egy általánosított, környezetfüggő változatát. A módszert magyar nyelvű szófaji egyértelműsítési problémán kiértékelve azt találta, hogy szignifikánsan képes javítani az osztályozási pontosságot a korábban használt kombinálási módszerekhez képest.

V.) A Szerző megvizsgálta számos kombinációs stratégia alkalmazhatóságát egy természetes nyelvi probléma, a főnévi csoportok elemzése terén. Az osztályozási hatékonyság növelése érdekében a Boosting algoritmust a feladat speciális igényeihez igazította. Az futtatások eredménye szerint a javasolt algoritmus javított a faminták felismerési pontosságán.

Bibliography

[1] Abney S., Partial Parsing via Finite-State Cascades, in Proceedings of ESSLLI'96 Robust Parsing Workshop, pp. 1-8, 1996.

[2] Alexin, Z., Zvada, Sz., Gyimóthy, T., Application of AGLEARN on Hungarian Part-of-Speech Tagging, Second Workshop on Attribute Grammars and their Applications, WAGA'99, pp. 133-152, 1999.

[3] Alexin, Z., Csirik, J., Gyimóthy, T., Bibok, K., Hatvani, Cs.. Prószéki, G., Tihanyi, L., Manually Annotated Hungarian Corpus, Proceedings of the Research Note Sessions of the 10th Conference of the European Chapter of the Association for Computational Linguistics, EACL'03, pp. 53-56, 2003.

[4] Bácsi, J., Kerekes, J., Lódiné, Szabó K., Sejtes, Gy., "BeszédMester" – computer-aided teaching reading and speech therapy, Módszertani Közlemények, 2004/2, pp. 61-69, 2004. (in Hungarian)

[5] Bengio, Y., Neural Networks for Speech and Sequence Recognition, International Thomson Computer Press, London, UK, 1996.

[6] Beyerlein, P., Discriminative Model Combination, Proc. ICASSP'98, pp. 481-484, 1998.

[7] Bilmes, J. A., Kirchoff, K., Directed Graphical Models of Classifier Combination: Application to Phone Recognition, Proceedings of ICSLP'2000, pp., 921-924, 2000.

[8] Bishop C. M., Neural Networks for Pattern Recognition, Clarendon Press, 1995.

[9] Bourlard, H., Hermansky, H. and Morgan, N., Towards Increasing Speech Recognition Error Rates, Speech Communication, pp. 205-231, May 1996.

[10] Bourlard, H. A., Morgan, N., Hybrid HMM/ANN Systems for Speech Recognition: Overview and New Research Directions, In: Giles, C. L. and Gori, M. (eds.), Adaptive Processing, Springer LNAI 1387, pp. 389-417, 1998.

[11] Bourlard, H., Non-Stationary Multi-Channel (Multi-Stream) Processing Towards Robust and Adaptive ASR, Proceedings of the Workshop on Robust Methods for Speech Recognition in Adverse Conditions, pp. 1-10, 1999.

[12] Brants, T., TnT – A Statistical Part-of-Speech Tagger, Proceedings of the Sixth Applied Natural Language Processing, 2000.

[13] Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J., Classification and Regression Trees, Belmont, Wadsworth International Group, 1984.

[14] Breiman, L., Arcing classifiers, The Annals of Statistics, Vol. 26, No. 3, pp. 801-849, 1998.

[15] Breiman, L., Random Forests, Machine Learning, Vol. 45, No. 1, pp. 5-32, 2001.

[16] Breiman, L., Bagging Predictors, Machine Learning, Vol. 24, No. 2, pp. 123-140, 1996.

[17] Brill, E., Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging, Computational Linguistics, Vol. 21, No. 4, pp. 543-565, 1995.

[18] Chang, S., Greenberg, S., Application of Fuzzy-Integration-Based Multiple-Information Aggregation in Automatic Speech Recognition, Proceedings of the IEEE Conf. on Fuzzy Integration Processing, Beijing, 2003.

[19] Chang, S., Wester, M., Greenberg, S., An Elitist Approach to Automatic Articulatory-Acoustic Feature Classification for Phonetic Characterization of Spoken Language, Computer Speech and Language, Vol. 47, pp. 290-311, 2005.

[20] Chawla, N. V., Japkowicz, N. and Kolcz, A. (eds.), Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Data Sets, http://www.site.uottawa.ca/ nat/Workshop2003/workshop2003.html, 2003.

[21] Claes, T., Dologlou, I., Bosch, L., Compernolle, D., A Novel Feature Transformation for Vocal Tract Length Normalization in Automatic Speech Recognition, IEEE Trans. on Speech and Audio Processing, Vol. 6, pp. 549-557, 1998.

[22] T.G. Dietterich, *Machine-Learning Research: Four Current Directions,* The AI Magazine, Vol. 18, No. 4, pp. 97-136, 1998.

[23] Dimitrova, L., Erjavec, T., Ide, N. , Kaalep, H. J., Petkevic, V., Tufis, D., Multext-East: Parallel and Comparable Corpora and Lexicons for Six Central and Eastern European Languages, Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics, pp 315-319, 1998.

[24] Duda, R. O., Hart, P. E. and Stork, D. G., Pattern Classification, Wiley and Sons, 2001.

[25] Eide, E., Gish, H., A Parametric Approach to Vocal Tract Length Normalization, ICASSP, pp. 1039-1042, 1997.

[26] Ellis, D. P. W., Improved recognition by combining different features and different systems, Proceedings of AVIOS'2000, pp. 236-242, 2000.

[27] Erjavec, T., Monachini, M., Specification and Notation for Lexicon Encoding, Copernicus project 106 - MULTEXT-EAST, Work Package WP1 - Task 1.1 Deliverable D1.1F., 1997.

[28] Felföldi, L., Kocsor, A., AHP-based Classifier Combination, Proceedings of PRIS-2004, pp. 45-58, Porto, 2004.

[29] Felföldi, L., Kocsor, A., Tóth, L.: Classifier Combination in Speech Recognition, Per. Pol. Elec. Eng., Vol. 47, No. 1-2, pp. 125-140, 2003.

[30] Freund, Y., Boosting a weak learning algorithm by majority, Information and Computation, Vol. 121, No. 2, pp. 256-285, 1995.

[31] Freund, Y., Shapire, R., A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences, Vol. 55, pp. 119-139, 1997.

[32] Friedman, T. H. J., Tibshirani, R., Additive logistic regression, a statistical view of boosting, The Annals of Statistics, Vol. 28, No. 2, pp. 337-374, 2000.

[33] Fukunaga, K., Statistical Pattern Recognition, New York, Academic Press, 1989.

[34] Fumera, G., Roli, F., Linear Combiners for Classifier Fusion: Some Theoretical and Experimental Results, 4th Int. Workshop on Multiple Classifier Systems (MCS 2003), January, 2003.

[35] Futó, I. (ed.), Artificial Intelligence, Aula, 1999. (in Hungarian)

[36] Goos, G., Hartmanis, J. and van Leeuwen, J. (eds.), Multiple Classifier Systems, Lecture Notes in Computer Science Vol. 2709, Springer, 2003.

[37] Gosztolya, G. and Kocsor, A., Tóth, L., Felföldi, L., Various Robust Search Methods in a Hungarian Speech Recognition System, Acta Cybernetica, Vol. 16., pp. 229-240, 2003.

[38] Gosztolya, G. and Kocsor, A., Improving the Multi-stack Decoding Algorithm in a Segment-based Speech Recognizer, In: P. W. H. Chung et al. (eds.), Proceedings of the 16th Int. Conf. on IEA/AIE 2003, LNAI 2718, pp. 744-749, Springer Verlag, 2003.

[39] Gosztolya, G., Kocsor, A., Speeding Up Dynamic Search Methods in Speech Recognition, In: Ali, M., Esposito, F. (eds.), Proceedings of the 18th Int. Conf. on IEA/AIE, LNCS 3533, pp. 98-100, Springer Verlag, 2005.

[40] Gosztolya, G., Kocsor, A., A Hierarchical Evaluation Methodology in Speech Recognition, Acta Cybernetica, Vol. 17, pp. 213-224, 2005.

[41] Hagen, A., Morris, A., Recent advances in the multi-stream HMM/ANN hybrid approach to noise robust ASR, Computer Speech and Language, Vol. 19, pp. 3-30, 2005.

[42] Halberstadt, A. K., Heterogeneous Measurements and Multiple Classifiers for Speech Recognition, Ph.D. Thesis, Dep. Electrical Engineering and Computer Science, MIT, 1998.

[43] Hand D. J. and Yu K., Idiot's Bayes - Not so stupid after all? – Int. Statistical Review, Vol. 69, pp. 385-398, 2001.

[44] Hócza, A., Alexin, Z., Csendes, D., Csirik, J., Gyimóthy, T., Application of ILP methods in different natural language processing phases for information extraction from Hungarian texts, Proceedings of the Kalmár Workshop on Logic and Computer Science, pp. 107-116, 2003.

[45] Hócza, A. Noun Phrase Recognition with Tree Patterns, in Proceedings of the Acta Cybernetica, Szeged, Hungary, 2004.

[46] Hócza, A., Felföldi, L., Kocsor, A.: Learning Syntactic Patterns Using Boosting and Other Classifier Combination Schemas, Proceedings of the 8th International Conference on Text, Speech and Dialogue, TSD 2005, LNAI, 3658, pp. 69-76, Springer Verlag, 2005.

[47] Horváth, T., Alexin, Z., Gyimóthy, T., Wrobel, S., Application of Different Learning Methods to Hungarian Part-of-Speech Tagging, Proceedings of ILP99, LNAI, Vol. 1634, pp. 128-139, 1999.

[48] Van Horn, K. S., A Maximum-entropy Solution to the Frame-dependency Problem in Speech Recognition, Tech. Rep., Dept. of Computer Science, North Dakota State Univ., Nov. 2001.

[49] Huang, X. D., Acero, A. and Hon, H-W., Spoken language processing, Prentice Hall, 2001.

[50] Hwang, J. N., Lay, S. R., Maechler, M., Martin, R. D., Schimert, J., Regression modelling in back-propagation and projection pursuit learning, IEEE Trans. on Neural Networks, Vol. 5, No. 3, pp. 342-353, 1994.

[51] Jančovič, P., Ming, J., Hanna, P., Stewart, D., Smith, J., Combining Multi-Band and Frequency-Filtering Techniques for Speech Recognition in Noisy Environments, Proceedings of TSD'2000, pp. 265-270, 2000.

[52] Jain, A. K., Duin, R., Mao, J., Statistical Pattern Recognition: A Review, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 1, pp. 4-37, 2000.

[53] Japkowicz, N. (ed.), Proceedings of the AAAI'2000 Workshop on Learning from Imbalanced Data Sets, AAAI Tech. Report WS-00-05, 2000.

[54] Jelinek, F., Statistical Methods for Speech Recognition, MIT Press, 1997.

[55] Kertész-Farkas, A., Fülöp, Z. and Kocsor, A., Compressed Storage of Hungarian vocabularies using Nondeterministic Automaton, Proc. MSZNY, pp. 231-236, 2003. (in Hungarian)

[56] Kiraz, G. A., Compressed Storage of Sparse Finite-State Transducers, In: O. Boldt and H. Jürgensen (eds.), Proc. of WIA'99, LNCS Vol. 2214, pp. 109-122, Springer, 2001.

[57] Kirchhoff, K., Bilmes, J. A., Combination and joint training of acoustic classifiers for speech recognition, Proceedings of ISCA ASR 2000, pp. 17-23, 2000.

[58] Kis, B., Naszódy, M., Prószéki, G., Complex Hungarian syntactic parser system, in Proceedings of the MSZNY 2003, pp. 145-151, Szeged, Hungary, 2003.

[59] Kittler, J., Hatef, M., Duin, R.P.W., Matas, J., *On Combining Classifiers,* IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 20. No. 3, March 1998.

[60] Kocsor, A., Kuba, A. Jr., Tóth, L., An Overview of the OASIS speech recognition project, Proceedings of the 4th International Conference on Applied Informatics, Eger-Noszvaj, Hungary, pp. 94-102, 1999.

[61] Kocsor, A., Kuba, A. Jr., Tóth, L., Jelasity, M., Felföldi, L., Gyimóthy, T., Csirik, J., A Segment-Based Statistical Speech Recognition System for Isolated/Continuous Number Recognition, Proc. of the FUSST'99, Sagadi,Estonia, pp. 201-211, 1999.

[62] Kocsor, A., Tóth, L., Kuba Jr., A., Kovács, K., Jelasity, M., Gyimóthy, T., Csirik, J., A Comparative Study of Several Feature Space Transformation and Learning Methods for Phoneme Classification, International Journal of Speech Technology, Vol. 3, Number 3/4, pp. 263-276, 2000.

[63] Kocsor, A., Tóth, L., Felföldi, L., Application of Feature Transformation and Learning Methods to Phoneme Classification, Proceedings of IEA/AIE 2001, LNAI 2070, pp. 502-512, Springer, 2001.

[64] Kocsor, A. and Tóth, L., Application of Kernel-Based Feature Space Transformation and Learning Methods to Phoneme Classification, Applied Intelligence, Vol. 21, No. 2, pp. 129-142, 2004.

[65] Kocsor, A. and Tóth, L., Kernel-Based Feature Extraction with a Speech Technology Application, IEEE Transactions on Signal Processing, Vol. 52, No. 8, pp. 2250-2263, 2004.

[66] Kuba, A., Bakota, T. Hócza, A., Oravecz, Cs. Comparing different POS-tagging Techniques for Hungarian, In Proceedings of the MSZNY 2003, pp. 16-23. Szeged, Hungary, 2003.

[67] Kuba A. Jr., Felföldi L., Kocsor, A., POS tagger combinations on Hungarian text, Proceedings of the 2nd International Joint Conference on Natural Language Processing, IJCNLP, 2005.

[68] Kuba, A., Hócza, A., Csirik, J. POS Tagging of Hungarian with Combined Statistical and Rule-Based Methods, Proceedings of the 7th International Conference of Text, Speech and Dialogue, pp. 113-121, 2004.

[69] Manning, C. D. and Schütze, H., Foundations of Statistical Natural Language Processing, MIT Press, 2000.

[70] Marcus, M. P., Santorini, B., Marcinkiewicz, M. A. Building a large annotated corpus of English: the Penn Treebank, Association for Computational Linguistics, 1993.

[71] Megyesi, B., Brill's Rule-Based POS Tagger for Hungarian, Department of Linguistics, Stockholm University, Sweden, 1998.

[72] Megyesi, B., Brill's POS tagger with extended lexical templates for Hungarian, Workshop (W01) on Machine Learning in Human Language Technology, ACAI'99, 1999.

[73] Megyesi, B., Improving Brill's POS tagger for an agglutinative language, Proceedings of the Joint Sigdat Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, EMNLP/VLC '99, pp. 275-284, 1999.

[74] Meir, R., Rätsch, G., An introduction to boosting and leveraging, In Machine Learning Summer School, pp. 118-183, 2002.

[75] Merz, C. and Murphy, P., UCI repository of machine learning databases, http://www.ics.uci.edu/ mlearn/MLRepository.html, 1998.

[76] Mohammed, M., Gader, P., Generalized Hidden Markov Models - Parts I and II, IEEE Trans. on Fuzzy Systems, Vol. 8, No. 1, pp. 67-94, February 2000.

[77] Mohri, M., Pereira, F. and Riley, M., Weighted finite-state transducers in speech recognition, Computer Speech and Language, Vol. 16, pp. 69-88, 2002.

[78] Ngai, G., Florian, R. Transformation-Based Learning in the Fast Lane, Proceedings of North American ACL 2001, pp. 40-47, June, 2001.

[79] Oravecz, Cs., Dienes, P., Efficient Stochastic Part-of-Speech Tagging for Hungarian, Proceedings of the Third International Conference on Language Resources and Evaluation, LREC2002, pp. 710-717, 2002.

[80] Ostendorf, M., Digalakis, V., Kimball, O. A., From HMMs to Segment Models: A Unified View of Stochastic Modeling for Speech Recognition, IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. 4. pp. 360-378, 1996.

[81] Paczolay, D., Felföldi, L., Kocsor, A., Classifier Combination Schemes In Speech Impediment Therapy Systems, Acta Cybernetica, Vol. 17. No. 2, pp. 385-399, 2005.

[82] Paczolay, D., Kocsor, A., Tóth L., Real-Time Vocal Tract Length Normalization in a Phonolgical Awareness Teaching System, Text Speech and Dialogue, Vol. 2807, pp. 4-37, 2003.

[83] Pitz, P., Molau, S., Schlüter, R., Ney, H., Vocal Tract Normalization Equals Linear Transformation in Cepstral Space, Proc. EUROSPEECH, Vol. 4, pp. 2653-2656, 2001.

[84] Prószéky, G., Humor: a Morphological System for Corpus Analysis, Language Resources for Language Technology, Proceedings of the First European TELRI Seminar, pp 149-158, 1995.

[85] Pylkönnen, J., Kurimo, M., Duration Modeling Techniques for Continuous Speech Recognition, Proceedings of ICSLP' 2004, pp. 385-388, 2004.

[86] Quinlan, J. R., C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, California, 1993.

[87] Rabiner, L. R., Juang B. H., Fundamentals of Speech Recognition, Englewood Cliffs, NJ, Prentice Hall, 1993.

[88] Ramshaw, L. A., and Marcus, M. P. Text Chunking Using Transformational-Based Learning, in Proceedings of the Third ACL Workshop on Very Large Corpora, Association for Computational Linguistics, 1995.

[89] Rätsch, G., Onoda, T., Mueller, K. R., Soft margins for adaboost, Machine Learning, Vol. 43, No. 3 pp. 287, 2001.

[90] Richard, M. D. and Lippmann, R. P., Neural network classifiers estimate Bayesian a posteriori probabilities, Neural Computation, Vol. 3, No. 4, pp. 461-483, 1991.

[91] Rodriguez, J. J., Kuncheva, L. I., Alonso, C. J., Rotation forest: A new classifier ensemble method, IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI), Vol. 28, No. 10, pp. 1619-1630, 2006.

[92] Roli, F., Fumera, G., Analysis of Linear and Order Statistics Combiners for Fusion of Imbalanced Classifiers, 3rd Int. Workshop on Multiple Classifier Systems (MCS 2002), June, 2002.

[93] Saul, L. K., Rahim, M. G., Allen, J. B., A Statistical Model for Robust Integration of Narrowband Cues in Speech, Computer Speech and Language, Vol. 15, No. 2, pp. 175-194, April 2001.

[94] Schapire, R. The strength of weak learnability, Machine Learning, Vol. 5, pp. 197-227, 1990.

[95] Schapire, R., Singer, Y., Improved boosting using confidence-rated predictions, Machine Learning, Vol. 37, No. 3, pp. 297-336, 1999.

[96] Schölkopf, B., Smola, A. And Müller, K. -R., Nonlinear Component Analysis as a Kernel Eigenvalue Problem, Neural Computation, Vol. 10(5), 1998.

[97] Sejtes, Gy., Kocsor, A., The Database Specification of SpeechMaster, Alkalmazott Nyelvtudomány, Vol. IV, No. 1, pp. 81-89, 2004. (in Hungarian)

[98] Sharma, S. R., Multi-Stream Approach To Robust Speech Recognition, Ph.D. Dissertation, Oregon Graduate Institute of Science and Technology, 1999.

[99] Sima'an, K. Computational complexity of probabilistic disambiguation by means of tree grammars, In Proceedings of COLING-96, Copenhagen, 1999.

[100] Szarvas, M., Mihajlik, P., Fegyó, T. and Tatai, P., Automatic Recognition of Hungarian: Theory and Practice, International Journal of Speech Technology, Vol. 3, No. 3/4, pp. 277-287, 2000.

[101] Tax, D. M. J., van Breukelen, M., Duin, R. P W. and Kittler, J., Combining multiple classifiers by averaging or by multiplying?, Pattern Recognition, Vol. 33, pp. 1475-1485, 2000.

[102] Tjong Kim Sang, E. F., and Veenstra, J. Representing text chunks, in Proceedings of EACL '99, Association for Computational Linguistics, 1999.

[103] Tjong Kim Sang, E. F. Noun Phrase Recognition by SystemCombination, in Proceedings of the first conference on North American chapter of the Association for Computational Linguistics, Seattle, pp. 50-55, 2000.

[104] Tóth, L., Kocsor, A., Kovács, K., A Discriminative Segmental Speech Model and its Application to Hungarian Number Recognition, In: Sojka, P. et al. (eds.), Proceedings of Int. Conf. on Text, Speech and Dialogue TSD'2000, Lecture Notes in Artificial Intelligence Vol. 1902, pp. 307-313, Springer, 2000.

[105] Tóth, L., Kocsor, A., Gosztolya, G., Telephone Speech Recognition via the Combination of Knowledge Sources in a Segmental Speech Model, Acta Cybernetica, Vol. 16, pp. 643-657, 2004.

[106] Tóth, L., Kocsor, A., Csirik, J., On Naive Bayes in Speech Recognition, International Journal of Applied Mathematics and Computer Science, Vol. 15, No. 2, pp. 287-294, 2005.

[107] Tóth, L., Kocsor, A., Kovács, K., Felföldi, L., On the Integration of Linguistic Knowledge Sources in Discriminative Segment-Based Speech Recognizers, I. Magyar Számítógépes Nyelvészet Konferencia, Dec. 10-11., Szeged, pp. 169-175, 2003.

7. Tóth, L., Kocsor, A., Kovács, K., Felföldi, L.: (in Hungarian) *On the Integration of Linguistic Knowledge Sources in Discriminative Segment-Based Speech Recognizers*, I. Magyar Számítógépes Nyelvészet Konferencia, Dec. 10-11., Szeged, pp. 169-175, 2003

[108] Tumer, K., Ghosh, J., Analysis of Decision Boundaries in Linearly Combined Neural Classifiers, Pattern Recognition, Vol. 29, pp. 341-34, 1996.

[109] Tufis, D., Dienes, P. Oravecz, Cs., Váradi, T., Principled Hidden Tagset Design for Tiered Tagging of Hungarian, International Conference on Language Resources and Evaluation (LREC00), 2000.

[110] van Halteren, H., Zavrel, J., Daelemans, W., Improving data driven wordclass tagging by system combination, Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics, pp. 491-497, 1998.

[111] Uebel, L. F., Woodland, P. C., An Investigation into Vocal Tract Lenght Normalisation, Proc. EUROSPEECH, Vol. 6, pp. 2527-2530, 1999.

[112] Valiant, L. G., A theory of the learnable, Commun. ACM, Vol. 27, No. 11, pp. 1134-1142, 1984.

[113] Vapnik, V. N., Statistical Learning Theory, John Wiley & Sons Inc., 1998.

[114] Váradi, T., Oravecz, Cs., Morpho-syntactic ambiguity and tagset design for Hungarian, Proceedings of the EACL LINC Workshop on Annotated Corpora, pp. 8-13, 1999.

[115] Váradi, T., The Hungarian National Corpus, Proceedings of the Third International Conference on Language Resources and Evaluation, LREC2002, pp. 385-396, 2002.

[116] Váradi T., Shallow Parsing of Hungarian Business News, in Proceedings of the Corpus Linguistics 2003 Conference, pp. 845-851, Lancaster, 2003.

[117] Vicsi, K, Tóth, L., Kocsor, A., Csirik, J., MTBA – A Hungarian Telephone Speech Database, Híradástechnika, Vol. LVII, No. 8, pp. 35- 43, 2002. (in Hungarian)

[118] Viterbi, A., Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Trans Information Theory, vol IT-13, pp. 260-269, 1967.

[119] Wegmann, S., McAllaster, D. , Orloff, J., Peskin B., Speaker Normalization on Coversational Telephone Speech, ICASSP, Vol. 1, pp. 339-341, 1996.

[120] Westphal, M., Schultz, T., Waibel, A. , Linear Discriminant - A New Criterion For Speaker Normalization, ICSLP, No. 755, 1998.

[121] Zhan, P., Westphal, M., peaker Normalization based on Frequency Warping, ICASSP, Vol. 1, pp. 1039-1042, 1997.

[122] Wolpert, D. H., Stacked Generalization, Neural Networks, Vol. 5, pp. 241-259, 1992.

[123] Young, S. et al., The HMM Toolkit (HTK) – software and manual, http://htk.eng.cam.ac.uk, 2005.

[124] Yun, Y.-S. and Oh, Y.-H., A Segmental-Feature HMM for Speech Pattern Modeling, IEEE Signal Processing Letters, Vol. 7, No. 6, June 2000.