University of Szeged
Research Group on Artificial Intelligence

Classifier Combination Systems and their Application in Human Language Technology

Summary of the PhD Dissertation

by

László Felföldi

Supervisors:

Prof. János Csirik
Dr. András Kocsor

Szeged
2008

Introduction

When designing pattern recognition systems[8; 20; 30] the goal is to construct a classifier that will model the behaviour of a system. For each pattern of a pattern space the classifier has to select a class label from the set of available labels. The construction of a classifier (i.e. the learning process) is based on a set of labelled examples, and strongly depends on prior knowledge about the pattern space and the characteristics of the given examples. Given infinite training data, consistent classifiers approximate the Bayesian decision boundaries to arbitrary precision, hence 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 networks when dealing with classification problems so as to ensure that a good model or parameter set is found. However, selecting one from among the classifiers is not necessarily the ideal solution 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, various techniques can be employed for combining the output of all available classifiers.

A large number of combination schemes have been proposed in the literature [7; 20; 21], these schemes differing from each other in their architecture, the characteristics of the combiner, and the selection of the individual classifiers. Static strategies, such the "Prod", "Max", "Min" rules, are frequently used in multiple-classifier systems, due to their simplicity and the small resource requirements, but these systems cannot provide any significant improvement in the classification quality. On the other hand, the Adaboost algorithm and its variants can dramatically increase the performance of the original classifier, especially in the cases of "weak" classifers, but for the final solution they require hundreds or thousands of iterations. In practice, most of the applications cannot provide the huge amount of resources required for applying a very large number of classifier instances. In this dissertation an effective combination method will be proposed that ensures a good combination performance for these kind of applications.

In the applications of Automatic Speech Recognition the system use machine learning methods, like Artificial Neural Networks (ANNs) or Gaussian Mixture Models (GMMs), to assign labels to speech segments or frames. Improving the quality of this phone classification would also have an effect on the overall recognition quality. Thus it is important to examine the effects of the combination strategies on the overal system performance.

Another promising area for applying the combination techniques is Natural Language Processing. In Part-Of-Speech tagging and Noun Phrase parsing, applying static combinations like "Majority Voting" and "Prod" rules has proved to be effective in improving the performance of the overall system. The more advanced adaptive techniques, however, cannot be used directly for context-dependent applications. In this thesis we will examine the kind of adaptive combination strategies that can be applied to handle the special needs of these NLP tasks.

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 process 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 tasks. 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 to make the final decision.

Types of Knowledge Sources

The goal of designing a classifier 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 able to provide the label that it prefers, but a number of machine learning methods can supply more information that the combiner can exploit. Combination strategies can incorporate the following types of classifiers based on the kind of information they provide:

- **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 is 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.

Static Combination Schemes

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 $p(\mathbf{x}^{(i)}|\omega_k)$ 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

$$f(\mathbf{x}) = \omega_j, \quad j = \underset{k}{\operatorname{argmax}}\, P(\omega_k|\mathbf{x_1}, \ldots, \mathbf{x_R}).$$

The combination strategies differ in how they approximate the a posteriory probabilities from the values $p(\omega_k|\mathbf{x}^{(i)})$, and usually assume that the class priors are all equal. In the following we list some of the most frequently used combination rules:

- **Product Rule:**

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

- **Sum Rule:**

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

- **Max Rule:**

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

- **Min Rule:**

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

- **Median Rule:**

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

- **Majority Voting Rule** Hardening a posteriori

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

where Δ_{ki} are the hardened probabilities $P(\omega_k|\mathbf{x}^{(i)}$:

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

- **Borda count:**

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

where ρ_{ki} are the ranking information coefficients:

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

Additive Combintation Models

Although the simple static combination techniques have become popular in multiple-classifier systems, owing to their simplicity they cannot be adapted to the special characteristics of particular applications. For this reason, adaptive methods like additive combination schemes have become the focus of research.

These combination methods calculate the weighted sum of the output of the standalone classifiers. Assuming that the classifier supplies confidence information type, it can be represented by:

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

where $f_i^j(\mathbf{x})$ is output of the classifer $\mathcal{C}_j$ for class ω_j.

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.

1. *Simple Averaging.* In this simplest case, the weights can be selected so that they all have the same value:
$$w_j = \frac{1}{N}.$$

2. *Weighted Averaging.* Experiments show [26] that *Simple Averaging* can be out-performed when the chosen weights are inversely proportional to the error rate of the corresponding classifier:
$$w_j = \frac{1}{E_j},$$
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.

3. *Hierarchical methods.* To calculate the weights w_j one can take those values that minimize some kind of distance function between the computed and expected a posteriory probabilities:
$$\min_w \sum_{x \in \mathcal{X}} \sum_{i=1}^{l} \mathcal{L}(\hat{f}_i(x), p_i(x)),$$
where $\mathcal{L}(\hat{f}_i(x), p_i(x))$ 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 ML method.

Bagging

The Bagging (Bootstrap aggregating) algorithm[4] takes a vote on 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).

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.

Algorithm 1 Bagging algorithm

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

Boosting

The underlying idea of boosting is to combine classifiers rules iteratively to form an ensemble that will improve the performance of each single member. Boosting theory has its roots in Probably Approximately Correct (PAC) learning [29]. PAC provides 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 [29], Valiant showed that simple classifiers, 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 most polular Boosting method is the AdaBoost (Adaptive Boosting) algorithm [12]. 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 each misclassified example is increased.

In the following we shall deal with only the case of binary classification, i.e. when only two class labels $\{-1, +1\}$ are available. Let us consider 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 being misclassified in the previous iteration is increased.

The loss function $\mathcal{L}$ used for updating the weights is usually an exponential loss function, but other loss functions have been also proposed in the literature, as in the cases of Logitboost [13] and Arcing [5].

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

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

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 huge amount of resources needed for applying such a very large number of classifier instances.

In this dissertation we present an effective combination solution for these kinds of applications, based on the Analytic Hierarchy Process method, a popular tool for Multi-Ccriteria Decision Making. These applications finds the best possible decision based on the criteria given by pairwise comparisons. 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:

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

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 define the matrix of weight ratios W by

$$w_{ij} = \frac{w_i}{w_j}.$$

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

$$(Ww)_i \;\; = \;\; \sum_{j=1}^{n} W_{ij} w_j$$

$$= \sum_{i=1}^{n} \frac{w_i}{w_j} w_j = \sum_{j=1}^{n} w_i$$
$$= nw_i.$$

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:

$$a_{ij} \approx \frac{w_i}{w_j}.$$

Here the elements of the unknown vector w are the importance values. Following Saaty we shall assume that

$$a_{ij} > 0,$$

and

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

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.

In linear classifier combinations the combined a posteriory 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.

In the experiments we compared the performance of linear combination schemes with different weights computation methods. We examined 2 schemes of averaging, called "SA" and "WA" (simple and weighted averaging), and 4 schemes of "AHP", which differed in the database sizes when their comparison matrices were calculated. The results demonstrate that all the combinations here improved the generalization performance of the simple classifier. In almost every case AHP-based combinations outperformed the weighted averaging combinations.

	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 1: Classification errors [%] on the UCI Letter database. The error of the standalone ANN classifier was 13.78%

The design of the AHP combination method and the experiment were published in [10].

Phoneme Classification

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 units have to be chosen in a general speech recognition task. The most commonly used unit of this kind is the phoneme, hence in the following we will focus on the classification problem of phonemes.

All the experiments were performed using our segment-based speech recognizer system called "OASIS" as a testing environment. In the first step of testing we combined the outputs of the selected classifiers (SVM, ANN, and kNN) by applying various combination rules. Table 2 suggests that there is no definite optimal rule for combining classifiers using this database. Combiners which applied the Sum rule performed the best, but the improvement compared to the others was only marginal.

	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 2: Classification error of hybrid combinations using classifiers ANN, SVM, and kNN. The rows represent databases with different feature sets

The following steps examined the efficiency of Bagging and Boosting. Bagging could improve classification performance, almost to the same level of the previous combination methods, but it

required more processing time. Since 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 we applied were "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.

The experiments on Phoneme Classifications were published in [11].

Vocal Tract Length Normalization

The efficiency of a speech recognizer application is highly dependent on the performance of the given phoneme classifier module. Also, it is even more important for phonetic teaching system like our Phonological Awareness Teaching System called "Speech-Master". Since the system should work reliably 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 these techniques is 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 [6; 9; 31] one can build recognizers that work robustly with voice samples from males, females and children.

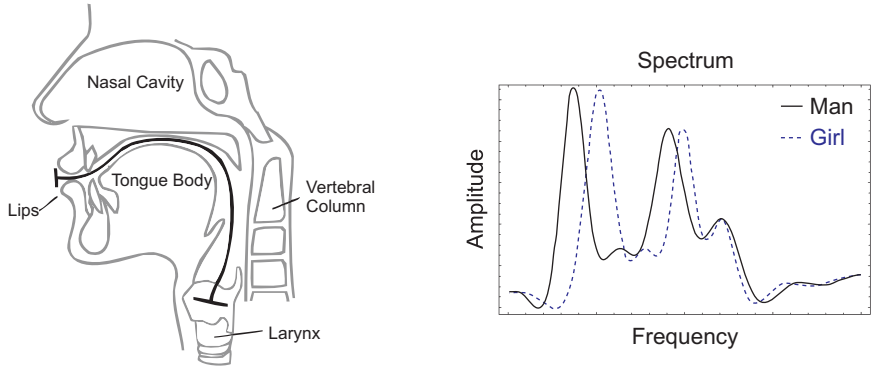

Figure 1: Vocal Tract Length and its frequency shifting. The graph drawn with solid and dashed line shows the spectrum of a vowel uttered by a man and a girl, respectively.

In [6] the average vocal tract length for men was reported 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 the 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 formant frequencies are inversely proportional to 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 [9; 31].

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. There

are numerous techniques for estimating the warping parameters, e.g. linear discriminant method (LD-VTLN) that minimizes the fraction of between-class and within-class covariance. These optimization methods, however, work off-line, and they require all the utterances in advance. To have the advantages of speaker normalization in on-line systems, machine learning methods can be applied for the estimation of the correct parameters. Out of the many possible regression techniques we chose to experiment with neural nets. The task of this real-time vocal tract length normalization method (RT-VTLN) is to estimate the optimal LD-VTLN warping parameter for each speaker based on the actual spectral frame without warping.

To improve the performance of the on-line system RT-VTLN, we employed static classifier combination strategies "Prod", "Sum", "Max", "Min", and "Majority Voting", and the advanced schemes "AHP", "Bagging", and "Boosting". The experimental results are shown in Table 3.

For each combiner the recognition accuracy was measured on 3 different databases: "All", "RT-VTLN", and "Concat", which besides the warped features, contain the original features as well. The effect of a combination depends on the database complexity. With the "All" database, each of the combiners gives a better performance than that for the original classification. On the warped databases ("RT-VTLN" and "Concat") the traditional combinations have less of an influence. Bagging and Boosting gave the best scores due to the large number of classifiers used.

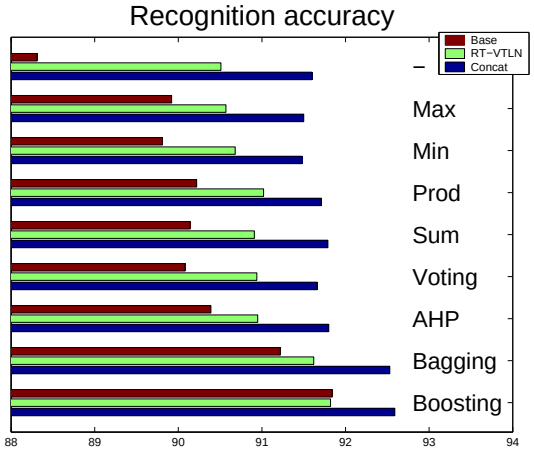

Table 3: Recognition accuracy on the databases with combination (in %). The various bars in each triplet correspond to the databases, and bar-triplets show the results for a given combiner.

Comparing the above results with the reference figure of LD-VTLN (92.55%), we may conclude that with a properly chosen combination scheme the regression based real-time VTLN method (Boosting on Concat, 92.67%) can outperform even the results of the off-line method LD-VTLN.

The experiments with Vocal Tract Length Normalization were published in [25].

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,

as well as its 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.

An exhaustive investigation on combinations of different POS taggers is available in [15]. Voting strategies and multi-level decision methods (stacking) have been investigated also. For Hungarian, a third possible approach was studied by Horvath et. al [19]. According to the results in [24], 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. To build more robust combination strategies we need to investigate how to adapt the more sophisticated adaptive techniques like "Boosting" to POS tagging applications.

The Boosting algorithm is based on weighting the independent instances based on the classification error of the previously trained learners. It builds classifiers, during each iteration, which 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[3] 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 a 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, and the combined final error is expressed as the relative number of misclassified instances.

For the experiments, the training and testing datasets were chosen from the business news domain of the Szeged Corpus. The results of the training and testing error rates are shown in Figure 2. 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. Bagging achieved only a moderate gain in accuracy, its relative error reduction rate being 18%.

The experiments with Part-of-Speech tagging were published in [22].

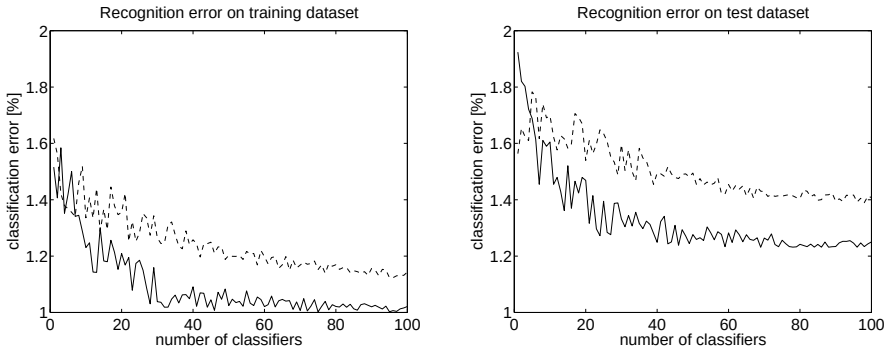

Figure 2: Classification error of *Bagging* and *Boosting* algorithm on the training and training datatsets (dashed: Bagging, solid: Boosting).

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 thesis we concentrated on the NP parsing of Hungarian texts, and applied PGS alorithm[17] on sentences from the Szeged Corpus. To make the method more robust, we investigated how the combination strategies affect its overall performance.

Since the PGS parser algorithm provides confidence information type, the majority of the combination schemes 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 parsers that should concentrate on instances that behaved 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.

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 4: Results achieved by combining PGS parsers

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

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. Comparing these results with the performance of the standalone learner, we see that combinations can improve the classification scores by some 10%.

The experiments with Syntactic Parsing were published in [18].

Conclusions

Combination algorithms, especially 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 of Speech Technology 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.

Summary of the Author's Contributions

In the following we summarize the results of the author by arranging them into five thesis points.

I.) The author developed a new linear combination strategy, based on the Analytic Hierarchy Process method, 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 called Oasis Speech Lab. Using the integrated combination module, he compared the efficiency of various combination techniques in speech recognition tasks. The experments justify 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 recogniton performance. The proposed schemas were integrated into the award-winning Phonological Awareness Teaching System "SpeechMaster".

IV.) The author developed a novel context-dependent variant of the Adaboost algorithm. Applied on a POS tagging application of Hungarian text 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 Syntactic parsing. Hhe adapted the combination algorithma to the special requirements of the problem. In experiments he found that the proposed combination strategies were indeed capable of enhancing the accuracy of tree pattern recognition.

	[10]	[11]	[25]	[22]	[18]
I.	•				
II.		•	•		
III.			•		
IV.				•	•
V.					•

Table 5: The relation between the thesis topics and the corresponding publications.

References

[1] 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.

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

[3] 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.

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

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

[6] 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.

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

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

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

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

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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, pp. 191-196, 2005.

[24] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] Tóth, L., Kocsor, A., Kovács, K., 2000. 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.

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

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

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