Multiple Classifier Systems for Embedded String Patterns

Barbara Spillmann, Michel Neuhaus, and Horst Bunke

Institute of Computer Science and Applied Mathematics, University of Bern, Neubrückstrasse 10, CH-3012 Bern, Switzerland

Abstract. Multiple classifier systems are a well proven and tested instrument for enhancing the recognition accuracy in statistical pattern recognition problems. However, there has been reported only little work on combining classifiers in structural pattern recognition. In this paper we describe a method for embedding strings into real vector spaces based on prototype selection, in order to gain several vectorial descriptions of the string data. We present methods for combining multiple classifiers trained on various vectorial data representations. As base classifiers we use nearest neighbor methods and support vector machine. In our experiments we demonstrate that this approach can be used to significantly improve the classification accuracy of string patterns.

1 Introduction

Building multiple classifier systems (MCSs) has been a topic of intensive research for many years [1,2]. The goal is to outperform the classification accuracy of a set of individual classifiers by combining them in an appropriate way. That is, one aims at creating a set of classifiers with a large diversity such that the weakness and errors of one classifier are compensated by other classifiers. A large number of methods for producing multiple classifier systems have been proposed and the success of these methods has been impressively demonstrated [3,4,5,6,7].

However, almost all papers in the field of multiple classifier systems have concentrated on vectorial pattern representations. Almost no work using structural data, such as strings or graphs, has been reported in the literature [8,9,10]. Using a structural representation of patterns rather than feature vectors has some advantages and has been proven to be a powerful means for many applications [11,12,13,14]. In the current paper we focus on pattern representations in terms of strings, i.e. sequences of symbols. To perform recognition tasks, one needs to define a distance measure, which is, in case of strings, usually the edit distance [15]. String edit distance allows one to implement nearest neighbor classifiers. Consequently, building multiple classifier systems is normally restricted to the creation of an ensemble of k-nearest-neighbor classifiers (kNN).

In this paper, we present a multiple classifier approach applicable to string patterns. The key idea is to use the transformation method proposed in [16,17] for embedding strings into dissimilarity spaces by means of prototype selection. This method has also been used in [18]. It is a general approach that is suitable

F. Schwenker and S. Marinai (Eds.): ANNPR 2006, LNAI 4087, pp. 177–187, 2006.

to make the whole spectrum of classifiers known from statistical pattern recognition available to string representations. It has been shown that the classification accuracy of strings can be significantly increased by applying such an embedding and by classifying the vectorial data gained from this procedure. In this paper we go one step further. As each concrete transformation from the string to the vectorial domain depends on various parameters, it is straightforward to generate several such embeddings by varying these parameters. Then the vectorial representations obtained from different embeddings are utilized to build a classifier ensemble.

This paper is organized as follows. Section 2 gives an overview of the embedding mechanism of strings into vector spaces. In Section 3, the architecture of our multiple classifier systems is described, including the creation of the classifier ensembles. Experimental results of the method, applied to handwritten digits, are discussed in Section 4. Finally, in Section 5 some conclusions are drawn.

2 From the String Domain to the Vector Space

Let A be a finite alphabet of symbols and A^* be the set of all strings over A. Furthermore, let ϵ denote the empty symbol. A string can be modified by edit operations: The replacement of a symbol $a \in A$ by $b \in A$ is called a substitution, and if $a = \epsilon$ or $b = \epsilon$ we term it an insertion or deletion, respectively. In order to measure the dissimilarity of strings, a cost c is assigned to each edit operation. Given a sequence $S = e_1, \ldots, e_n$ of edit operations, its cost is defined as $c(S) = \sum_{i=1}^{n} c(e_i)$. Considering two strings $x, y \in A^*$ and all sequences of edit operations that transform x into y, the edit distance, $d(x, y)$, of x and y is the sequence with minimum cost. The edit distance can be computed by dynamic programming in $O(nm)$ time and space, where n and m are the lengths of the two strings under consideration [15].

With the notation introduced above, a transformation of a string pattern into a vector representation can be defined. The transformation is based on a set of selected strings, the *prototypes*. A string is transformed into a vector by calculating its edit distances to all prototypes, where each resulting distance represents one vector component. More formally, let $\mathcal{X} \subset A^*$ denote a set of string patterns over the alphabet A, and $\mathcal{P} = \{p_1, \ldots p_n\} \subset \mathcal{X}$ a set of selected prototypes. For a given string $x \in \mathcal{X}$ a vectorial description of x is defined by the transformation $t_n^{\mathcal{P}}$:

$$t_n^{\mathcal{P}} : \mathcal{X} \to \mathbb{R}^n \text{ with } t_n^{\mathcal{P}}(x) = (d(x, p_1), \ldots, d(x, p_n)) \tag{1}$$

As a consequence, the number of prototypes, n, defines the dimensionality of the vector space, $\mathbb{R}^n$.

Obviously, the characteristics of the transformation depend on the size of $\mathcal{P}$ as well as on the patterns selected as prototypes. An algorithm that selects the prototypes p_i $(i = 1, \ldots, n)$ out of $\mathcal{X}$ is called *prototype selection strategy* s. With $s(\mathcal{X}) = \mathcal{P}$ we denote the procedure of building $\mathcal{P}$ out of $\mathcal{X}$ by applying s. Examples of different selection strategies, such as the *border prototype selector*,

the *center prototype selector*, the *spanning prototype selector* and the *k-medians prototype selector*, have been discussed in [16,18].

In order to make available classifiers from statistical pattern recognition to string classification by means of the transformation procedure introduced above, the transformation $t_n^{\mathcal{P}}$ is applied to each element of a given dataset. After the transformation is accomplished and the whole dataset is embedded into a vector space, one can train any classifier suitable for vector spaces. Experiments with the kNN classifier and the support vector machine have been described in [18].

3 Multiple Embedding MCS

The idea underlying the construction of our multiple classifier system is to apply different prototype selection strategies. Each individual selection strategy yields a different embedding of the original string data, i.e. a different set of vectors. For each such set of vectors, an individual classifier is constructed. Eventually these individual classifiers are combined in a multiple classifier system.

3.1 Triple Embedding MCS

The prototype selection strategies used in this work are the following.

Spanning Prototype Selector (*s-ps*) [18]. This is a method that selects prototypes, such that they are evenly distributed over the given set of strings. The procedure is the following. The first prototype selected is the set median string, an approximation of the generalized median string [19]. The next prototypes are iteratively determined by selecting the string with largest sum of the edit distances to the previously selected prototypes.

Random k-Medians Prototype Selector (*rkm-ps*). This strategy has been applied in [18], where it is referred to as *k-median prototype selector*. It performs a k-medians clustering and defines the resulting cluster centers to be the prototypes. An important point is that the initial cluster centers are chosen randomly. Thus, the algorithm is non-deterministic.

Spanning k-Medians Prototype Selector (*skm-ps*). This method is a deterministic variant of the *rkm-ps*. The initial cluster centers are not chosen randomly, but as specified by the *s-ps* method. That is, the prototype selection according to the *s-ps* method is improved with respect of the k-medians algorithm's clustering properties.

One of the base classifiers applied in this work is the kNN classifier (with Minkowski metric). By applying any of the three strategies *s-ps*, *rkm-ps* and *skm-ps* we can transform a dataset of string patterns into a vector space. Formally, we denote a set of string patterns by $\mathcal{X} = \{x_1, \ldots, x_N\}$ and sets of prototypes by $\mathcal{P}_{s\text{-}ps}$, $\mathcal{P}_{rkm\text{-}ps}$, $\mathcal{P}_{skm\text{-}ps} \in \mathcal{X}$, where the indices indicate the selection strategy used for their construction. We use parameters $n_{k\text{NN}_{s\text{-}ps}}$, $n_{k\text{NN}_{rkm\text{-}ps}}$ and $n_{k\text{NN}_{skm\text{-}ps}}$ to refer to the number of prototypes, and define the following three transformation functions:

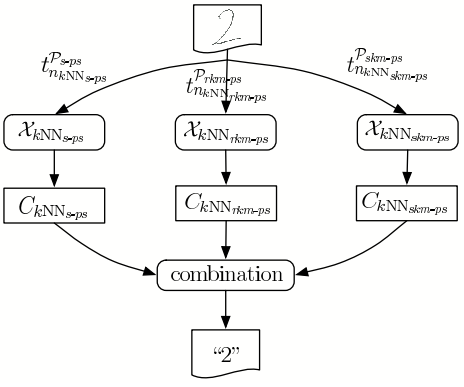

Fig. 1. Example classification of a handwritten digit with the *triple embedding MCS* and the kNN classifier as base classifier (*TE-MCS, kNN*)

$$
\begin{aligned}
t^{\mathcal{P}_{s\text{-}ps}}_{n_{k\mathrm{NN}_{s\text{-}ps}}} &: \mathcal{X} \to \mathbb{R}^{n_{k\mathrm{NN}_{s\text{-}ps}}} \\
t^{\mathcal{P}_{rkm\text{-}ps}}_{n_{k\mathrm{NN}_{rkm\text{-}ps}}} &: \mathcal{X} \to \mathbb{R}^{n_{k\mathrm{NN}_{rkm\text{-}ps}}} \\
t^{\mathcal{P}_{skm\text{-}ps}}_{n_{k\mathrm{NN}_{skm\text{-}ps}}} &: \mathcal{X} \to \mathbb{R}^{n_{k\mathrm{NN}_{skm\text{-}ps}}}
\end{aligned}
\tag{2}
$$

By applying the transformations (2) to the whole dataset $\mathcal{X}$ we get three vectorial representations of $\mathcal{X}$, denoted by $\mathcal{X}_{k\mathrm{NN}_{s\text{-}ps}}$, $\mathcal{X}_{k\mathrm{NN}_{rkm\text{-}ps}}$ and $\mathcal{X}_{k\mathrm{NN}_{skm\text{-}ps}}$, respectively. For each of these three sets a specific classifier of the k-nearest-neighbor type, denoted by $C_{k\mathrm{NN}_{s\text{-}ps}}$, $C_{k\mathrm{NN}_{rkm\text{-}ps}}$ and $C_{k\mathrm{NN}_{skm\text{-}ps}}$, is constructed. When classifying a pattern $x_i \in \mathcal{X}$, each classifier produces a class prediction. Given an unknown input pattern x_i, the combination of the three classifiers' outputs results in the final prediction of the system, $c_{k\mathrm{NN}(x_i)}$.We call this setup *triple embedding MCS for the kNN classifier (TE-MCS, kNN)*. Fig. 1 gives an illustration of this setup with an example of a handwritten digit "2" to be recognized.

We can now analogously build a *triple embedding MCS* using as base classifier a support vector machine (SVM) with radial basis function as kernel function. The SVM [20,21] is a classifier for statistical data that makes use of a kernel function to transform vector data into higher-dimensional feature spaces. The key idea is to find a separating hyperplane in the feature space with a maximal margin between the classes. This is an optimization problem usually solved by quadratic programming. For this classifier type we define the transformation functions:

$$
\begin{aligned}
t^{\mathcal{P}_{s\text{-}ps}}_{n_{\mathrm{SVM\text{-}R}_{s\text{-}ps}}} &: \mathcal{X} \to \mathbb{R}^{n_{\mathrm{SVM\text{-}R}_{s\text{-}ps}}} \\
t^{\mathcal{P}_{rkm\text{-}ps}}_{n_{\mathrm{SVM\text{-}R}_{rkm\text{-}ps}}} &: \mathcal{X} \to \mathbb{R}^{n_{\mathrm{SVM\text{-}R}_{rkm\text{-}ps}}} \\
t^{\mathcal{P}_{skm\text{-}ps}}_{n_{\mathrm{SVM\text{-}R}_{rkm\text{-}ps}}} &: \mathcal{X} \to \mathbb{R}^{n_{\mathrm{SVM\text{-}R}_{rkm\text{-}ps}}}
\end{aligned}
\tag{3}
$$

We get another three embeddings into vector spaces $\mathcal{X}_{\text{SVM-R}_{s\text{-}ps}}$, $\mathcal{X}_{\text{SVM-R}_{rkm\text{-}ps}}$ and $\mathcal{X}_{\text{SVM-R}_{skm\text{-}ps}}$. Using these embeddings, an ensemble of three SVM classifiers with radial basis function, $C_{\text{SVM-R}_{s\text{-}ps}}$, $C_{\text{SVM-R}_{rkm\text{-}ps}}$ and $C_{\text{SVM-R}_{skm\text{-}ps}}$, are trained. This ensemble yields output $c_{\text{SVM-R}(x_i)}$ for an input pattern $x_i \in \mathcal{X}$. We call this setup *triple embedding MCS for the SVM with radial basis function (TE-MCS, SVM-R)*.

The third setup of that type is the support vector machine with linear kernel function as base classifier. We define the transformations:

$$\begin{aligned} t^{\mathcal{P}_{s\text{-}ps}}_{n_{\text{SVM-L}_{s\text{-}ps}}} &: \mathcal{X} \to \mathbb{R}^{n_{\text{SVM-L}_{s\text{-}ps}}} \\ t^{\mathcal{P}_{rkm\text{-}ps}}_{n_{\text{SVM-L}_{rkm\text{-}ps}}} &: \mathcal{X} \to \mathbb{R}^{n_{\text{SVM-L}_{rkm\text{-}ps}}} \\ t^{\mathcal{P}_{skm\text{-}ps}}_{n_{\text{SVM-L}_{skm\text{-}ps}}} &: \mathcal{X} \to \mathbb{R}^{n_{\text{SVM-L}_{skm\text{-}ps}}} \end{aligned} \tag{4}$$

and denote the transformed datasets by $\mathcal{X}_{\text{SVM-L}_{s\text{-}ps}}$, $\mathcal{X}_{\text{SVM-L}_{rkm\text{-}ps}}$ and $\mathcal{X}_{\text{SVM-L}_{skm\text{-}ps}}$. The classifiers to be trained are called $C_{\text{SVM-L}_{s\text{-}ps}}$, $C_{\text{SVM-L}_{rkm\text{-}ps}}$ and $C_{\text{SVM-L}_{skm\text{-}ps}}$, and the resulting class prediction for a string $x_i \in \mathcal{X}$ is abbreviated with $c_{\text{SVM-L}(x_i)}$. This MCS is referred to as *triple embedding MCS for the SVM with linear kernel function (TE-MCS, SVM-L)*.

3.2 Hierarchical Multiple Embedding MCS

In the previous section, three MCSs have been presented that use different string embeddings to generate ensembles. However, the classifier ensembles are always of the same type. Now, we want to make use of the possibility of applying different classifier types, i.e. we define an MCS that aggregates the kNN, the SVM with radial basis function, and the SVM with linear kernel function. The idea is to build a hierarchical system that consists of the three *triple embedding MCS* described in Section 3.1. In order to do so we simply combine the class predictions, $c_{k\text{NN}(x_i)}$, $c_{\text{SVM-R}(x_i)}$ and $c_{\text{SVM-L}(x_i)}$, of the three *TE-MCS*. Obviously, this MCS uses a total of nine different embeddings into vector spaces. Let's call this setup *hierarchical multiple embedding MCS (HME-MCS)*. Fig. 2 illustrates the recognition of a string pattern using an *HME-MCS*.

4 Experimental Results

In this section we provide experimental results for the *triple embedding MCSs* and the *hierarchical multiple embedding MCS* introduced in Section 3. For our experiments we use the *Pendigits* database described in [22] (original, unnormalized version). The original version consists of 10,992 instances of handwritten digits labeled with one of the ten class names "0" to "9", where 7,494 digits are used for training and 3,498 for testing (see Fig. 3). Each digit is originally given as a sequence of two-dimensional points. Certainly, in case of handwritten digits

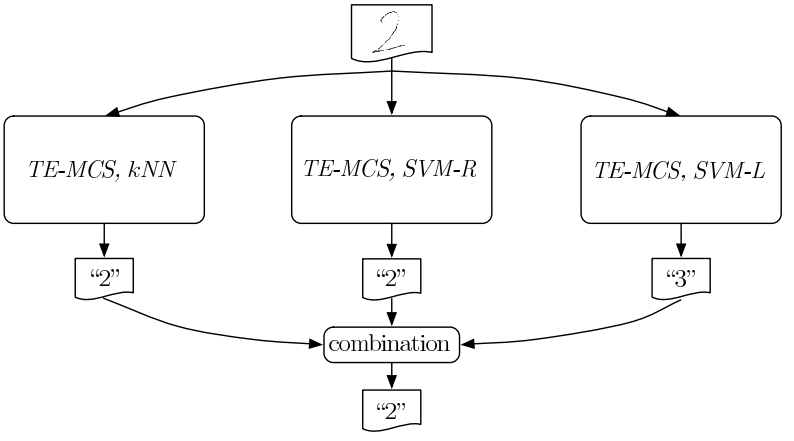

Fig. 2. Example classification of a handwritten digit with the *hierarchical multiple embedding MCS* (*HME-MCS*)

it might also be useful to directly extract features instead of extracting a suitable string representation. However, as we want to demonstrate the feasibility and the possible power of our multiple classifier approach, the *Pendigits* dataset can be regarded as an exemplary representative for any set of string patterns.

To obtain a suitable string representation, each digit curve is approximated by a sequence of vector segments of fixed length l, such that each start and end point lies on the original curve. An optimal value of l is determined on a validation set (see below) by performing a k-nearest-neighbor classification with the edit distance as a distance measure. Given the sequence of vector segments as a string representation, the substitution costs are defined as the absolute value of the vector difference to the power of q_v, where q_v is an arbitrary positive real number. As deletion and insertion costs we take the arithmetic mean of the extremal values (0 and $(2l)^{q_v}$) of the substitution costs, which is $2^{q_v-1}l^{q_v}$. This cost function is referred to as *vector cost function*. Another way of defining a string representation is to consider the sequence of angles between pairs of successive vector segments. The costs assigned to the edit operations are constantly set to $0 \leq q_a \leq \frac{\pi}{2}$ in case of angle insertions and deletions, and for substitutions the costs are given by the absolute difference between the angles. We call this cost function *angle cost function*. Notice that also the values of the cost function parameters q_v and q_a are optimized on the validation set.

Our experimental evaluation consist of six independent runs, where three of them use a vector-based string representation, and the other three use an angle-based one. We use three different splits into training, test and validation set. The first split, referred to as `pen1`, follows the original division into training and test set, where one fifth of the training set is used for validation. The second and third split, `pen2` and `pen3`, are further setups, where the size of each set is approximately the same, but different partitions have been used. The validation set is used for the purpose of optimizing the following parameters:

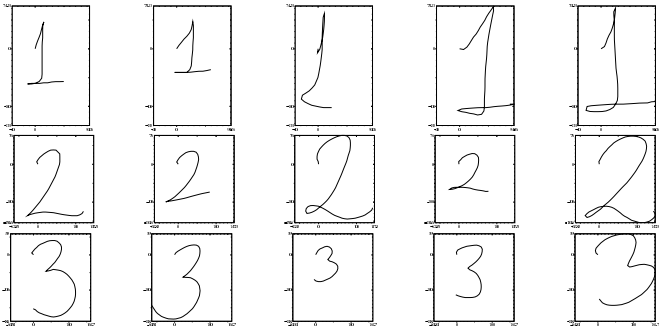

Fig. 3. Example patterns of the *Pendigits* dataset with the class labels "1", "2" and "3"

- **String representation parameters** q_v, q_a, and l. To find appropriate values of these parameters we perform a k-nearest-neighbor classification in the string domain and use the edit distance with either the *angle cost function* or the *vector cost function*. Once these parameters are optimized, they are kept unchanged for the rest of the experiment.
- **Dimensionality parameters** n_{c_s}, where $c \in \{k\text{NN}, \text{SVM-R}, \text{SVM-L}\}$ and $s \in \{s\text{-}ps, rkm\text{-}ps, skm\text{-}ps\}$. They determine the dimensionality of the vector spaces. Practically, we transform the validation set to various vector spaces with the dimensions 50, 100, 150, 200, 300, 400, 500, 800, 1000. In case of the *angle cost function*, also the values 1500 and 2000 are evaluated. The optimized values are determined by the classifier in the vector space. Once optimal values are found, the transformations $t^{\mathcal{P}_s}_{n_{c_s}}$ are applied, i.e. the whole dataset, including the elements of the training and test set, are embedded into vector spaces.
- **Classifier parameters** for the kNN classifier and the support vector machines. For each vectorial representation $\mathcal{X}_{c_s}$ a classifier C_{c_s} is trained on the validation set. For example, in case of the kNN this includes the number of nearest neighbors k and a Minkowski metric parameter.
- **Combination rule.** Normally, in a multiple classifier system, each participating classifier casts one vote, that is, each one chooses the most plausible class. Afterwards these votes have to be combined. There has been much of research on combining decision results in past years [23]. In this work, the classifiers' results are combined by the following three methods.

  The *plurality voting* method decides for the class which reaches the most votes among the involved classifiers [24]. It is a simple voting mechanism that only counts the occurrence of each class label output by a classifier. A similar voting method is the *runoff voting* where the voting process is performed in two steps [25]. First, each classifier votes for its most plausible class. The two candidates with the highest number of votes get another chance, and each classifier can vote for one of those two in a second round. The one with the most votes among the two wins the voting. A method different from the

Table 1. Recognition rates for the *triple embedding MCSs* and the *hierarchical multiple embedding MCS* using the *angle cost function*, where sequences of angles are used for the string representation. The labels `pen1`, `pen2` and `pen3` denote three different splits of the data into training, test and validation set.

	`pen1`	`pen2`	`pen3`
kNN string domain	88.56	92.48	92.71
TE-MCS with kNN	**90.62**	92.96	91.86
TE-MCS with SVM-R	**94.77**	**96.24**	**95.59**
TE-MCS with SVM-L	**93.85**	**95.23**	**95.33**
HME-MCS	**94.68**	**96.48**	**95.86**

Table 2. Recognition rates for the *triple embedding MCSs* and the *hierarchical multiple embedding MCS* using the *vector cost function*, where sequences of vectors are used for the string representation. The labels `pen1`, `pen2` and `pen3` refer to the same splits of the data into training, test and validation set as in Tab. 1.

	`pen1`	`pen2`	`pen3`
kNN string domain	97.48	99.33	99.33
TE-MCS with kNN	97.57	99.36	99.01
TE-MCS with SVM-R	**98.20**	**99.55**	**99.55**
TE-MCS with SVM-L	97.74	**99.60**	99.33
HME-MCS	**98.31**	**99.68**	**99.57**

two methods mentioned above is the *Borda count* [26,27]. It belongs to the category of ranking methods, and is based on a complete preference ranking from all classifiers over all classes. For each class the mean rank is computed. Then the top ranked class is declared the winner of the voting.

The validation set is used to determine the best voting strategy among these three methods. The method with highest performance on the validation set is then selected to classify the test set.

The training set is used to select the set of prototypes $\mathcal{P}$ from, i.e. the set of prototypes $\mathcal{P}$ is always a subset of the training set. And of course, it is also used for the training of the classifiers. In case of the kNN classifier, the nearest neighbors are selected from the whole training set, while for the SVM the support vectors are chosen from the training set.

The final results are produced on the test set. In Tab. 1 the results on the test set for the string representation with *angle cost function* are listed. Tab. 2 shows the results for the case of *vector cost function*. The first row shows the recognition results achieved with a kNN classifier in the original string domain, with optimized cost function parameters q_v, q_a and l. These three classification results shown in the first row are used as reference values for the classification in the vector domain and are meant to be outperformed by the multiple classifier systems presented in this paper. In rows 2 to 4, the results of the *triple embedding MCSs* with the base classifiers kNN, SVM-R, and SVM-L are listed. The results for the *hierarchical multiple embedding MCS* can be found in row 5.

Recognition rates printed in bold face refer to a statistically significant improvement compared to the string classification (row 1) at a significance level of 0.95. Note that the evaluation of each single classifier used for our multiple classifier systems has been presented in [18], where also the detailed classification results can be found.

The first point to notice is that 15 out of 18 experiments for the *triple embedding MCS* clearly outperform the classification in the string domain. There are only two setups, all based on the `pen3` split, where the the kNN classifier in the string domain performs better.

All *HME-MCS* have statistically significant better recognition rates than the string domain classification. And in 5 of 6 cases, they outperform all three *TE-MCS* from which they are built. Only in the `pen1` setup with *angle cost functions*, the recognition rate of the *HME-MCS* is slightly below the *TE-MCS* with the SVM-R base classifier. In all the other cases, the experiments with the *HME-MCS* provide the best results. In contrast to the *TE-MCSs* the *HME-MCS* consists of fundamentally different classifiers. Whereas the *TE-MCS* are based on an ensemble of the same classifier type, the *HME-MCS* provides a combination of kNN and support vector machine classification. We conclude that the embedding of string patterns into vector spaces using prototype selection allows one to build classifiers of essential diversity. By combining their results the traditional nearest-neighbor string classification can be significantly improved.

In [28], several MCS approaches have been tested on the same data. The methods *bagging*, *boosting*, *random subspace*, *random tree B*, *random forest-lg*, *random forest-1* and *random forest-2* were investigated by applying a 10-fold cross-validation. The ensembles were built using nine sets, the remaining set was used for testing, and the *Borda count* method applied for combination. The best result, 99.30, was achieved with the random subspace method. However, due to differences in the test procedure one has to be careful comparing these numbers to the results in Tab. 1 and 2. Yet we can state that two of three *HME-MCS* tests and five of nine *TE-MCS* tests with *vector cost function* outnumber the 99.30% correct recognition rate of the *random subspaces* method reported in [28].

5 Conclusions

In the present paper we propose a method for creating multiple classifier systems for string patterns. We apply a transformation procedure to embed strings into real vector spaces based on prototype selection. By selecting and applying three different selection strategies we gain vectorial representations for the string data. Given various vector space embeddings, kNN classifiers with Minkowski metric and SVM classifiers with radial basis function and linear kernel function are trained and combined. In a number of experiments we show that especially the combination of different classifier types leads to significantly better classification results than a nearest neighbor classification in the original string domain. This shows that our method can be an effective means to improve string clas-

sification. In the future, we would like to investigate further combinations of various selection strategies which might allow us to even improve the current methodology.

Acknowledgements

M. Neuhaus has been supported by the Swiss National Science Foundation NCCR program *Interactive Multimodal Information Management (IM)2* in the Individual Project *Multimedia Information Access and Content Protection.*

References

1. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience (2004)
2. Oza, N., Polikar, R., Kittler, J., Roli, F., eds.: Multiple Classifier Systems. Volume 3541 of LNCS., Seaside, CA, USA, Springer (2005)
3. Kittler, J., Roli, F., eds.: Multiple Classifier Systems. Number 1857 in LNCS, Cagliari, Italy, Springer (2000)
4. Kittler, J., Roli, F., eds.: Multiple Classifier Systems. Volume 2096 of LNCS., Cambridge, UK, Springer (2001)
5. Roli, F., Kitler, J., eds.: Multiple Classifier Systems. Volume 2364 of LNCS., Cagliar, Italy, Springer (2002)
6. Windeatt, T., ed.: Multiple Classifier Systems. Volume 2709 of LNCS., Guildford, UK, Springer (2003)
7. Roli, F., Kittler, J., Windeatt, T., eds.: Multiple Classifier Systems. Volume 3077 of LNCS., Cagliari, Italy, Springer (2004)
8. Schenker, A., Bunke, H., Last, M., Kandel, A.: Building graph-based classifier ensembles by random node selection. In: 5th International Workshop on Multiple Classifier Systems. Volume 3077 of LNCS., Springer (2004) 214–222
9. Neuhaus, M., Bunke, H.: Graph-based multiple classifier systems – a data level fusion approach. In: Proc. 13th Int. Conf. on Image Analysis and Processing. LNCS 3617, Springer (2005) 479–486
10. Marcialis, G.L., Roli, F., Serrau, A.: Fusion of statistical and structural fingerprint classifiers. In Kittler, J., Nixon, M.S., eds.: Audio-and Video-Based Biometrie Person Authentication, 4th International Conference. Volume 2688 of LNCS., Guildford, UK, Springer (2003) 310–317
11. Cha, S.H., Shin, Y.C., Srihari, S.N.: Approximate stroke sequence matching algorithm for character recognition and analysis. In: 5th International Conference on Document Analysis and Recognition. (1999) 53–56
12. Bunke, H., Bühler, U.: Applications of approximate string matching to 2D shape recognition. Pattern Recognition **26** (1993) 1797–1812
13. Chen, S.W., Tung, S.T., Fang, C.Y., Cheng, S., Jain, A.K.: Extended attributed string matching for shape recognition. Computer Vision and Image Understanding **70** (1998) 36–50
14. Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G.: Biological sequence analysis. Cambridge University Press, Cambridge, UK (1998)
15. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. Journal of the ACM **21** (1974) 168–173

16. Pekalska, E., Duin, R.P., Paclík, P.: Prototype selection for dissimilarity-based classifiers. Pattern Recognition **39** (2006) 189–208
17. Pekalska, E., Duin, R.P.W.: The Dissimilarity Representation for Pattern Recognition, Foundations and Applications. Volume 64 of Machine Perception Artificial Intelligence. World Scientific (2005)
18. Spillmann, B., Neuhaus, M., Bunke, H., Pekalska, E., Duin, R.P.: Transforming strings to vector spaces using prototype selection. submitted (2006)
19. Kohonen, T.: Median strings. Pattern Recognition Letters **3** (1985) 309–313
20. Vapnik, V.: The Nature of Statistical Learning Theory. 2nd edn. Springer-Verlag (2000) ISBN: 0-387-98780-0.
21. Vapnik, V.: Statistical Learning Theory. Wiley-Interscience (1998) ISBN: 0-471-03003-1.
22. Alpaydin, E., Alimoglu, F.: Department of Computer Engineering, Bogaziçi University, 80815 Istanbul Turkey (1998) `ftp://ftp.ics.uci.edu/pub/mlearn/databases/pendigits`.
23. Xu, L., Krzyzak, A., Suen, C.Y.: Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Transactions on Systems, Man, and Cybernetics **22** (1992) 418–435
24. Day, W.: Consensus methods as tools for data analysis. In Bock, H., ed.: classification and related methods for data analysis, Elsevier Sciencce Publishers B.V. (North Holland) (1988) 317–324
25. van Erp, M., Vuurpijl, L., Schomaker, L.: An overview and comparison of voting methods for pattern recognition. In: 8th International Workshop on Frontiers in Handwriting Recognition. (2002)
26. de Borda, J.C.: Memoire sur les elections au scrutin. Histoire de l'Academie Royale des Sciences, Paris (1781)
27. Ho, T.K., Hull, J.J., Srihari, S.N.: Decision combination in multiple classifier systems. IEEE Transactions on Pattern Analysis and Machine Intelligence **16** (1994) 66–75
28. Banfield, R.E., Hall, L.O., Bowyer, K.W., Bhadoria, D., Kegelmeyer, W.P., Eschrich, S.: A comparison of ensemble creation techniques. In Roli, F., Kittler, J., Windeatt, T., eds.: The Fifth International Conference on Multiple Classifier Systems, Cagliari, Italy. Volume 3077 of LNCS., Cagliari, Italy, Springer (2004) 223–232