Spoken Language Classification using Hybrid Classifier Combination *

Sheila Garfield, Stefan Wermter and Siobhan Devlin

Centre for Hybrid Intelligent Systems
University of Sunderland, School of Computing and Technology
St. Peter's Way, Sunderland SR6 0DD, United Kingdom
www.his.sunderland.ac.uk
{sheila.garfield,stefan.wermter}@sunderland.ac.uk

Abstract. In this paper we describe an approach for spoken language analysis for helpdesk call routing using a combination of simple recurrent networks and support vector machines. In particular we examine this approach for its potential in a difficult spoken language classification task based on recorded operator assistance telephone utterances. We explore simple recurrent networks and support vector machines using a large, unique telecommunication corpus of spontaneous spoken language. The main contribution of the paper is a combination of techniques in the domain of call routing. First, we find that simple recurrent networks perform better than support vector machines for this task. Second, we claim that the combination of simple recurrent networks and support vector machines provides slightly improved performance compared to the performance of either simple recurrent networks or support vector machines.

Keywords: classification, spontaneous language, dialogue, recurrent neural networks, support vector machines

1 Introduction

The human ability for robust language processing is demonstrated particularly in the area of handling ill-formed or incomplete language (Abney 1996, Clark 1996, Heeman and Allen 1999, Core and Schubert 1999a, Core and Schubert 1999b, Wermter and Weber 1997). Humans can understand sentences that would be interpreted as ungrammatical by a restrictive grammar although recognition of the correct parse can be hindered by acoustic noise and conceptual and grammatical misinterpretations. Furthermore, sounds from the surrounding environment can be mixed with the speaker's voice. However, humans are able to adapt to, or correct, particular grammatically or acoustically ill-formed language input. Therefore, in order to build systems that can communicate naturally with humans, it is important to integrate robust processing for realistic interfaces.

In the context of this paper the focus is on helpdesk scenarios. Processing spontaneous spoken language in helpdesk scenarios is important for automatic telephone interactions for telecommunication companies, banks, or other services to increase efficiency. Consequently there is a need to develop systems that can handle spoken language processing robustly. Given this background, the aim of the work presented in this paper was to investigate the combination of alternative techniques that may have potential for use in a classification task where the aim is to classify spontaneously spoken telephone utterances into one of a number of call classes by detecting the service required.

The remainder of the paper continues as follows: Section 2 provides a description of the Helpdesk corpus, Section 3 presents a brief overview of the two individual classification techniques adopted, Section 4 outlines the architecture of the hybrid combinations, Section 5 discusses the combination of the classifiers, Section 6 presents an analysis and discussion of the results, and finally Section 7 presents the conclusions.

2 Description of the Helpdesk Corpus and its Representation

2.1 Objective

The objective of this work is to address the problem of helpdesk automation and call routing using natural conversational language. In this work the textual transcriptions of spoken language input from customers is used. The task is to classify utterances into one of a number of call classes or categories. Consider the following example of an actual caller utterance:

*The authors thank Peter Durston of BTexact Technologies for helpful discussions and comments.

"*yeah yeah I wonder if you could please um I've been getting some a... anonymous calls and n... the they they're not er there's no number stored um*"

After human analysis of this example of spoken language input the caller would be referred to the nuisance calls bureau. Although the caller has not stated explicitly which service is required the operator has determined what he believes is the caller's intention from what the caller has said. This allows the speaker to utter sentences appropriate to an infinite number of situations (Burton-Roberts 1986). As a result the problem of automatically understanding fluent speech is difficult for a machine.

2.2 Description and Analysis of the Helpdesk Corpus

For this task a corpus from transcriptions of 8,441 recorded operator assistance telephone calls was used (Durston et al. 2001). The corpus covers calls received at all times of the day and therefore gives a representative selection of call traffic to the operator assistance service. The utterances range from simple direct requests for services to more descriptive narrative requests for help as shown by the examples below:

1. "*can I reverse the charges please*"
2. "*hi ya um me me next door neighbour's just informed me that our phone's um not working*"
3. "*oh I wonder if you can help me I've been trying to phone my grandson in Aberdeen I've got his phone number and he also has er you know one that receives messages I've used my memo thing twice and also by dialling the numbers and I'm getting the same thing which is a strange woman answering and just sort of saying nobody's at home would you leave your message and it's complete er I can't so I've also used as I say the ordinary digital thing and I still get this reply thing*"

These utterances demonstrate quite clearly the problems associated with automatically interpreting spoken language. The utterances are interspersed with filled pauses such as '*er*' and '*um*' which add nothing to the semantic content of the utterance. In the first example the caller makes a direct request for a service. In the second example the caller does not state directly whether he wants to be connected to, or wants the number for, the engineers' service to report a fault. The operator has to deduce what the intention of the caller is - that he wants to be connected to this service, before re-directing the call. In the third example the request for help is preceded by a lengthy description of the problem situation. The operator has to listen to the problem description and then propose some solution to the problem.

In the above examples the length of the utterances ranges from six words in the shortest utterance to 94 words in the longest utterance; although utterances in excess of 100 words have been recorded.

2.3 Call Transcription

The investigation focuses on a corpus from transcriptions of the first utterances of callers to the operator service. The first phrase uttered by the caller in response to the operator greeting is the first utterance. The transcription of each utterance is divided up into four segments: *call, call class, primary move* type and *request* type. The first utterance only of the caller is transcribed unless it is a greeting or phrase such as *'I'm sorry to trouble you'* or *'is that the operator?'*. In these cases the caller's second utterance is also transcribed. In the case of the last example in Table 1, the second part of the utterance of the caller would also be transcribed beginning at "I'm I'm trying to trace a call ....".

The transcribed recorded utterances were associated with one of the call classes and a primary move type by our telecommunication collaborator (Durston et al. 2001). There are 19 call classes in this task and these will be referred to throughout as "class 1", "class 2", etc. Eight separate call sets each containing 1,000 utterances and one call set containing 441 utterances were used in this study. To create the training and test sets the nine call sets were combined using a rotational approach. This approach produced nine overlapping training sets constructed by dropping one of the nine call sets each time to produce a 9-fold cross-validation (Bishop 1995) so that all examples in the corpus are at some time included in the test set. As a result at each rotation one call set was excluded and this set was used for the test set, for example, call sets 2-9 were combined to create a training set containing 7,441 utterances and call set 1 was used for the test set containing 1,000 utterances. The average length of an utterance in both the training and the test set is 22.12 words. An illustrative example is given in Table 2; however, not all call classes are shown.

2.4 Calls and Semantic Vector Representation

A semantic vector representation (Wermter et al. 1999) is generated for each word in a word list. Each vector represents the frequency of a particular word occurring in a call class and is independent of the number of examples observed in each call class. The number of calls in a call class can vary substantially. Therefore the frequency of a word w in call class c_i is *normalized* according to the number of calls in c_i, (1).

A *value* $v(w, c_i)$, (2), is computed for each element of the vector as the *normalized* frequency of occurrences of word w in call class c_i, divided by the sum of the *normalized* frequency of occurrences of word w in all call classes. That is:

$$Normalized\ frequency\ of\ w\ in\ c_i = \frac{Frequency\ of\ w\ in\ c_i}{Number\ of\ calls\ in\ c_i} \tag{1}$$

where:

$$v(w, c_i) = \frac{Normalized\ frequency\ of\ w\ in\ c_i}{\sum_j Normalized\ frequency\ for\ w\ in\ c_j},\ j \in \{1, \cdots n\} \tag{2}$$

Each call class is represented in the vector. An illustrative example is given in Table 3; however, not all call classes are shown.

As can be seen, in the illustrative example, words like *could* and *I* have fairly even distributions, while words like *transfer* and *charge* have more specific preferences. The example vector representation presented in Table 3 shows the numeric values associated with the utterance "could I make a transfer charge call please" for each word in a particular call class. The utterances in each call set are pre-processed to identify and mark the separate sections of each utterance, see example utterance, Table 4. The labels correspond to the sections of the utterance as outlined in Section 2.3.

The labels are only used in the processing of the call set for the purpose of generating the training and test vector representations used as input to the classification components described in Section 3.

2.5 Performance Evaluation Measures

The performance of the hybrid combinations was evaluated using the metrics recall and precision (Salton and McGill 1983) and the F-measure (Van Rijsbergen 1979). Recall and precision are standard measures of information retrieval performance. The F-measure is a combination of the precision and recall rates and is a method for calculating a value without bias, in other words, without favouring either recall or precision. Recall refers to the percentage of the total relevant utterances in the call set that are retrieved, while precision refers to the percentage of relevant utterances retrieved in relation to the number of utterances retrieved. Recall and precision are inversely related therefore as recall increases there is a tendency for precision to decrease and vice versa. Recall and precision can be calculated using the following equations:

$$Recall = \frac{Number\ of\ relevant\ utterances\ retrieved}{Total\ Number\ of\ relevant\ utterances\ in\ collection}$$

$$Precision = \frac{Number\ of\ relevant\ utterances\ retrieved}{Total\ Number\ of\ utterances\ retrieved}$$

The F-measure metric is a measure that provides the '*harmonic*' mean value, which does not favour either recall or precision, and is calculated using the recall and precision values:

$$F = \frac{2PR}{(P\ +\ R)}$$

3 Simple Recurrent Networks and Support Vector Machines

3.1 Simple Recurrent Network

The simple recurrent network (Elman et al. 1996, Wermter 2000, Wermter 1995), Figure 1, uses a set of context units which copy their activations (values) from the hidden units. The hidden units also receive input

from the context units. The hidden units are updated first before the context units copy the hidden units' values and provide input to the hidden units. Therefore the context layer enables the output values of the network's hidden units to be stored and then re-used in the network by providing the hidden layer with the pattern of the previous activation state (Elman 1991).

A simple recurrent network (SRN) with input, output, hidden, and context layers was used for our experiments. Training was carried out using supervised learning techniques (Elman et al. 1996, Wermter 1995). In general, the input to a hidden layer L_n is constrained by the underlying layer L_{n-1} as well as the incremental context layer C_n. The activation of a unit $L_{ni}(t)$ at time t is computed on the basis of the weighted activation of the units in the previous layer $L_{(n-1)i}(t)$ and the units in the current context of this layer $C_{ni}(t)$ limited by the logistic function f.

$$L_{ni}(t) = f(\sum_{k} w_{ki} L_{(n-1)i}(t) + \sum_{l} w_{li} C_{ni}(t)) \tag{3}$$

This provides a simple form of recurrence that can be used to train networks to perform sequential tasks over time which means the output of the network not only depends on the input, but also on the state of the network at the previous time step. This means events from the past can be retained and used in current computations, that is, the output at time *t* is re-used as the input at time *t + 1*. This allows the network to produce complex time-varying outputs in response to simple static input which is important when generating complex behaviour. As a result the addition of recurrent connections can improve the performance of a network and provide the facility for temporal processing (Wermter et al. 1999).

3.1.1 Construction of Training and Test Sets for Neural Networks

To create the input for the SRN the word list and tagged utterances in each call set were used together. For each utterance in the training and the test set the format of the generated sequences was the same as the example in Table 5.

The utterance was processed word by word and each word was associated with two sets of values; one set represented the call class of the utterance - the category output representation, and the other set represented the frequency of occurrence of the word - the word input representation. Each word and its associated values were stored in the training or test set one word per line.

In the example, in Table 5, the first set of values following each word, '1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0', indicate the call class associated with the utterance (the category output representation). The '1' in the first position denoted that the utterance was connected with call class 1. The second set of values were the frequency values associated with a particular word (the word input representations), in the example '0.07', '0.03', etc are related to the word 'COULD'; there is one value per call class. The values were extracted from the word list as each word in the utterances in the tagged call set was processed.

3.1.2 Training Environment

In one epoch, or cycle of training through all training samples, the network is presented with all utterances from the training set and the weights are adjusted at the end of each utterance because initially the output may not be the correct or desired output. A training algorithm is used that can compare the actual output to the desired output and calculate the difference so that the output produced comes a little closer to the desired output (LeCun et al. 1998).

The input layer has one input for each call class. During training and testing, utterances are presented sequentially to the network one word at a time. Each input receives the value of $v(w, c_i)$, where c_i denotes the particular call class of a word *w* which the input is associated with. Utterances are presented to the network as a sequence of word input and category output representations, one pair for each word. At the beginning of each new sequence the context layer is cleared and initialised with 0 values. Each unit in the output layer corresponds to a particular call class and the output unit that represents the desired call class is set to 1 and all other output units are set to 0.

3.1.3 Test Environment

After completion of the training phase the network was tested by presenting the test utterances to the network, again sequentially, one word at a time as a combination of word input and category output values. As the unknown test sample utterance was processed each output unit was checked for the activation value and compared with the target value to identify the desired output unit in the classification process.

3.1.4 Evaluation of Overall Simple Recurrent Network Performance

The performance of the SRN was evaluated using the metrics recall and precision (Salton and McGill 1983) and the F-measure (Van Rijsbergen 1979). In order to use these measures, in the testing phase, at the end of the sequence an utterance was labelled as *classified* to a particular call class if the activation value for the desired output unit was higher than 0.5. That is the utterance was counted as both *retrieved and relevant*. An activation value that did not exceed 0.5 for the desired output unit produced no classification result and the utterance was counted as not retrieved and relevant, termed *relevant*. An activation value on an output unit that was higher than 0.5, although the unit was not the desired output unit, produced no classification result and the utterance was counted as retrieved and not relevant, termed *retrieved*. These classification values were used to compute the recall and precision rates and the F-measure for each call class.

The test results for recall, precision, and F-measure are provided in Figure 2. The results presented are the results for the three networks trained and tested on each cross-validated call set averaged over the nine test sets.

3.2 Support Vector Machines

A support vector machine (SVM) is a binary classifier and the aim of support vector classification is to produce a classifier that will perform well on unseen examples which means it is able to generalise well (Opper 2001, Cooley 1999, Moghaddam and Yang 2001). This approach divides the problem space into two regions via a *dividing line* or *hyperplane* with a classifier referred to as the *optimal separating hyperplane*, Figure 3. This hyperplane separates positive examples from negative examples with a maximum margin (Dumais et al. 1998, Chapelle and Vapnik 2000, Opper 2001, Platt 1998, Feng and Williams 2001) between it and the nearest data point of each class.

SVMs make use of kernels (Moghaddam and Yang 2001, Opper 2001) solving a series of optimisation problems by calculating the inner products between vectors. All computations are performed directly in input space and depending on the kernel function used, for example, polynomial (4), radial basis function (RBF) (5), and sigmoid (6), different types of classifier can be constructed (Stitson et al. 1996, Moghaddam and Yang 2001, Schölkopf et al. 1995). *K* is a function that accepts two vectors as input and returns a scalar that represents the inner product of the vectors. The kernel function must obey Mercer's condition to be a valid kernel. Here, γ, *d*, *s*, and *c* are kernel parameters.

$$K(d_1, d_2) = (d_1 * d_2 + l)^d \tag{4}$$

$$K(d_1, d_2) = exp(-\gamma(d_1 - d_2)^d) \tag{5}$$

$$K(d_1, d_2) = tanh(s(d_1 * d_2) + c) \tag{6}$$

However, one potentially problematic area is the optimisation of parameters and choosing the right combination of parameters.

3.2.1 Construction of Training and Test Sets for Support Vector Machines

To create the input for the SVM the word list and tagged utterances in each call set were used together. An example of the input and output is presented in Figure 4.

If the input was an example of call class 1, denoted by the number '1' in the first position in the category output representation '1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0', the values written to the training or test set were those in the first column of the word input representation for each word in the utterance. The sequence was preceded by '+1' to indicate the sequence was an example of the call class under investigation. If the utterance was not an example of call class 1 because the number in the first position in the category output representation was '0' the values written to the training or test set were still those in the first column in the word input representation but the sequence was preceded by '-1' to indicate the sequence was not an example of the call class under investigation.

The training and test sequences were composed of feature-value pairs separated by a colon as appears in the example for the utterance '*can you try a number for me please*', the first example in Figure 4. In that example the first feature-value pair is '1:0.04'. The *feature*, in this case '1', is the position of the word in the utterance and the *value*, '0.04', is the frequency of occurrence of the word 'can' in any position for the

particular call class under investigation. In the first example '0.04' is the frequency of the word 'can' for the classifier associated with call class 1.

3.2.2 Support Vector Machine Experiments

We constructed a support vector machine classifier based on (Joachims 1999) for each call class. The training and test sets were generated from the same utterance sequences from the same nine call sets. The approach adopted was *one-versus-all*, that is to say the training and test sets used positive examples for the particular call class in question and negative examples of all the other call classes. The classifier in each case was trained and then tested on these sets.

On completion of the training phase a model is generated which is then used in the testing phase. In its basic form the decision function that determines whether a test example is classified can be described by a hyperplane *h*, attribute vector *x*, weight vector *w*, and a threshold *b* and takes the form of:

$$h(x) = sign(w * x + b) \tag{7}$$

where the output is:

+1 if w * x + b $>$ 0, and
-1 otherwise.

3.2.3 Evaluation of Overall Support Vector Machine Performance

At the end of both the training and the testing phases three additional values were obtained for each classifier: (1) the number of utterances retrieved that were relevant, termed *retrieved and relevant*, (2) the number of utterances retrieved that were not relevant, termed *retrieved*, and (3) the number of relevant utterances not retrieved, termed *relevant*. These values from each classifier were combined and used to calculate an overall performance figure for recall and precision (Salton and McGill 1983) and F-measure (Van Rijsbergen 1979).

The overall test recall, precision, and F-measure results for each type of kernel (RBF, sigmoid, and polynomial) are presented in Figure 5.

The overall result means the results of each of the 19 individual call class classifiers were added together to produce a result for each of the three types of kernel, for each test set. The results from each test set were then added together and averaged over the nine test sets. The results presented are the average of the nine test sets for each type of classifier.

4 Overview of the Hybrid Classifier Architecture

There are many types of classifier available in the field of pattern recognition. However deciding on the best classifier to use for a particular task can be problematic. Over the last few years there have been many approaches advocated for the combination of classifiers appearing under different names (Abbott 1999, Ho 2002). Whatever name is used for a particular approach the general idea is the same; multiple classifiers are trained and the results combined to provide a final output (Abbott 1999, Dietterich 2000, Demirekler and Altinçay 2002, Oh 2003). The general view (Abbott 1999, Dietterich 1998, Giacinto et al. 2000, Dietterich 2000, Duin 2002) is that ensembles of classifiers can prove to be better at classifying examples than individual classifiers but only if they each produce different errors (Hansen and Salmon 1990, Giacinto and Roli 2001, Kuncheva 2002). This relies on the notion that classifiers that are diverse will produce different errors on new examples. Therefore when the results are combined the correct classification is given. So if classifier '*A*' is wrong on example one but classifiers '*B*' and '*C*' may be correct then after combination, say by majority vote, example one will be classified correctly (Dietterich 1998). A key reason for combining classifiers is that if individual classifiers are created and then only the best performing classifier is selected it may be possible that useful information will be lost that is contained in the other classifiers (Tumer and Ghosh 1996). Therefore to reduce this loss the decision of each classifier is used by combining them before a final decision is made (Tumer and Ghosh 1996).

A block diagram, representing a top-level view of the general structure of our architecture is shown in Figure 6, which combines different components.

The input, that is a representation of the caller's first utterance, is presented to the classifiers: a support vector machine (SVM) and a simple recurrent network (SRN). The outputs from the classifiers are processed by a *combiner*, that is a combination rule, and the hybrid classifiers operate as a cross-validated committee as a result of training using 9-fold cross-validation (Dietterich 1997). The outputs are combined in parallel; the

outputs of the particular SRN and SVM combination are combined example by example at the same time. The combination of the outputs from each of the classifiers produces a final result for each example. The final result is the call class assigned to the example by the combined votes of the classifiers. The next section discusses the first approach used for classifier combination.

5 Combination of the Classifiers

Experimental work was undertaken to test the hypothesis that creating a hybrid combination of classifiers could produce an improvement in classification performance over that of the classifiers individually. The outputs of each of the classifiers were used to generate a vote for each example in the test set and the votes were combined to give an overall decision. The method chosen to create the combination was *majority vote*.

5.1 Development of the Majority Vote Combination Rule

5.1.1 The Majority Vote Rule

If the total vote for a particular call class was greater than a threshold the example was deemed to have been classified to that particular call class. The threshold for the number of votes required to decide whether or not an example was classified to a particular call class was governed by the number of classifiers being combined. The approach was to set a threshold based on the number of classifiers being used in the combination. If an even number of classifiers k are being combined the threshold t should be greater than half the number of classifiers. Where an odd number of classifiers k are being combined the threshold t should be greater than or equal to half the number of classifiers: $t = (k + 1)/2$ (Sebastiani 2002). If the total vote of the classifiers exceeds the threshold the example was deemed to have been classified to the particular call class voted for by the majority of the classifiers.

5.1.2 The Majority Vote Combination Method

For the SRN each output was assigned a call class label based on the unit in the output layer with the highest activation value and this was counted as a vote for the call class label associated with that unit. The SVM is a binary classifier and produces its output result as the distance from the margin to the hyperplane (the dividing line between the classes) using positive and negative values to determine on which side of the hyperplane an example lies. A positive value indicates the example is classified to the call class in question and a negative value indicates the example is not classified to that call class. One positive or negative value is produced for each example and the positive or negative value is converted into a vote for each example. A positive value is converted to '1' (one) and is a vote for the call class in question, while a negative value is converted to '0' (zero) and indicates no vote for the example.

Three different SVM classifiers were created for each of the 19 call classes, RBF, sigmoid, and polynomial and each classifier produced a vote for each example presented in the test set for a particular call class. The vote from the SRN was then combined with the vote from the SVM for each call class for the classifier type being combined, for example RBF. If there was a majority vote for a call class the example was counted as an example of that particular call class. A comparison was made to the actual call class for the example under examination and if the actual call class matched the call class voted for the example was counted as *retrieved and relevant* for that call class, if not the example was counted as *retrieved* only.

There were 19 individual SVM classifiers (one for each call class) for each of the three kernel types. During voting for an example in the test set the SVM classifiers for the kernel type being investigated, for example RBF, were combined on an individual call class basis with the SRN.[1] For each example in the test set the SRN was combined with the RBF call class 1 classifier, then the SRN was combined with the RBF call class 2 classifier and so on. The classifiers were combined in this way because, as indicated by the example of the voting process in Figure 7, generally only one SVM classifier made a vote for each example enabling the use of a *majority vote* rule.

In example one, in Figure 7, both SRNs and the SVM have voted for the example as being an example of call class 1. The SRN vote is indicated by the number '1' in the 'SRN Vote' column and the SVM vote is indicated by the number '1' in the 'Call Class Number 1' column. The number of votes for this example

[1] In practice, combinations involved one, two, and three SRNs and one, two, and three SVMs but for the purpose of description and illustration, the discussion is confined to two SRNs and one SVM.

was three therefore the threshold for the number of classifiers voting was exceeded and the example goes into the classification process for the purposes of recall and precision. For the calculation of recall and precision the example was counted as *retrieved and relevant* because the actual call class of the example was call class 1 indicated by the number '1' in the 'Actual Class' column and the majority of classifiers had voted for the example as belonging to that call class. When the actual call class of the example is different to the call class voted for by the majority of the classifiers then the example is counted as only *retrieved*.

Three approaches to parallel classifier combination were investigated: (1) combination after parallel training of the classifiers, (2) combination after training the classifiers in sequence, and (3) classifier combination using a casting vote. In total 21 hybrid combinations of SRNs and SVMs were investigated for each of the approaches. To enable a direct comparison between individual classifiers and combined classifiers the F-measure was used as the performance measure indicator. The F-measure is chosen because it provides a mean value with no bias towards either recall or precision and therefore a direct comparison can be made.

5.2 Classifier Combination after Training in Parallel

The first approach to classifier combination was combination after training the classifiers in parallel. In this first approach the classifiers processed the same data before the results were combined for an overall result. Each classifier performed the same task and was considered equal. The classifiers were independently trained and tested on the inputs. The outputs from each classifier were then combined using the method described below to produce a final classification result.

The classifiers were combined in the following manner. On each occasion three SRNs were trained and the combinations were: network one with the RBF kernel SVM, network one with the sigmoid kernel SVM, and network one with the polynomial kernel SVM, and the results were averaged. This was repeated for networks two and three and the average of the three results was calculated for each kernel type. For the combinations involving two networks, networks one and two were combined with the RBF, then the polynomial, then the sigmoid kernel SVM. The next combination was networks one and three with the three SVMs, then networks two and three with the three SVMs and the results averaged. Therefore the results presented are the averages of the SRN and SVM kernel combinations.

5.2.1 Evaluation of Combined Classifier Performance: Parallel

In this first approach to classifier combination there were five hybrid classifiers that produced an improvement in performance over the best performing individual SRN; the F-measure results are provided in Table 6.

The results in Table 6 suggest that no significant improvement in performance was gained by combining more than two SRNs and one SVM because combinations involving three SRNs and two SVMs, hence more classifiers, produced the same results as did a combination of three SRNs (81.4 F-measure). Based on these results there appears to be only a small advantage to creating hybrid classifiers, possibly because the performance of the SRN is better than that of the SVM. Therefore the voting of the hybrid classifier that comprised two SRNs and one SVM (polynomial) that had performed better than an individual SRN was analysed to try to establish a possible reason for only a small number of hybrid classifiers performing better than the individual SRNs and this is discussed next.

5.2.2 Investigation of Voting Performance: Parallel Training

An analysis of the voting by a combination of two SRNs and one SVM (polynomial) on a randomly selected test set provided a possible reason for such a small improvement in performance namely that the classifiers being combined were similar. This means the individual classifiers had classified the same examples correctly and the same examples incorrectly before voting. In the sample set 55% of the examples had been classified correctly by the individual classifiers. Therefore the classifiers when voting, for some examples, were all voting for the same example and the benefit of the majority vote was potentially ineffective.

The results of five of the hybrid classifiers were an improvement in performance compared to the performance of the best performing SRN and based on the analysis of the randomly selected test set there was a possibility that a further improvement in performance was achievable. In an attempt to create classifiers that were not as similar, the SVM classifiers were trained using a different sequential approach and this is described in Section 5.3. While the new training of the SVM classifiers and the testing of the hybrid classifiers was an attempt to examine in particular the performance of the five hybrid classifiers that had outperformed an individual SRN, all hybrid classifiers were trained and tested using the approach described in the next section.

5.3 Classifier Combination after Training in Sequence

In the second approach to classifier combination the classifiers were trained in sequence and then combined in parallel as previously. Again each classifier performed the same task and was considered equal, but, in this approach the classifiers were trained using a linear mode. In this second approach the first component acted as a pre-processor for the second component and the results were combined in parallel; the outputs from the first component were also used in the combination as well as being converted into inputs for the second component.

The SRNs did not undergo any further training; the same network outputs were used as those used in the previous section. The SVM classifiers were trained using the outputs produced by the SRN training; the outputs produced by the SRN were converted into inputs for the SVM.[2] The outputs generated by the SRN on completion of the training phase were recorded in a log file. The log file contained the outputs for each sequence, that is each caller utterance, word by word. The outputs for the sequence (each word) were used to create the input representation for the SVM and the SVM was trained on these inputs. After completion of the SVM training the outputs from the SVM were combined with the outputs from the SRN.

5.3.1 Evaluation of Combined Classifier Performance: Sequence

In this second approach to classifier combination the same hybrid classifiers were investigated as previously. The results presented are the average results produced by the hybrid classifiers on the nine test sets. After training the classifiers in sequence, six hybrid classifiers outperformed the best performing individual SRN. The F-measure results of the six hybrid classifiers are provided in Table 7.

Examining the results presented in Table 7 there is little difference between the performance of the hybrid classifiers and the individual SRN, that is, it makes little difference which SVM classifier is combined with the SRN. The reason for this is that the SRN has produced good results generally for most call classes whereas the performance of the SVM is very mixed across the call classes. Therefore the performance of either the SRN or the SVM on particular call classes has not been exploited to improve classification performance.

There was a small improvement in the performance of the hybrid classifiers using the approach described in this section compared with the hybrid classifiers using the approach described in Section 5.2. A possible reason for the improvement was that the classifiers were now less similar and were not classifying as many of the same examples correctly or incorrectly before voting. To determine whether the classifiers were less similar the voting process was again investigated and this is discussed in the next section.

5.3.2 Investigation of Voting Performance: Sequence Training

An examination of the voting of the combination of two SRNs and a polynomial SVM on the test set corresponding to the test set used in Section 5.2.2 revealed that now only 38% of the examples were correctly classified by the individual classifiers before voting. This meant that the majority vote had the potential to be of benefit because fewer examples had in effect been classified correctly before voting. As a result if two out of the three classifiers had voted for the same example, either two SRNs or an SRN and an SVM, there was still a majority vote for that example.

Unlike the previous hybrid classifiers where the addition of an extra SRN produced no extra improvement in performance, for the hybrid classifiers discussed in this section, the addition of an extra SRN did, in most cases, produce an improvement. Again, as reported in Section 5.2.1, the hybrid classifier of two SRNs and an RBF SVM produced the highest F-measure result suggesting that this is the optimum classifier combination. The results for the hybrid classifiers indicated an improvement in performance although there was a possibility that the strengths and weaknesses of particular classifiers were not being utilised.

The original idea was to consider the classifiers as a whole rather than focus on individual call class performances. The hybrid classifiers described in the previous sections had produced only a small improvement in performance over the individual SRN. As there was only a small improvement in performance a decision was made to examine the training sets of the SRNs and SVMs to establish whether either classifier had achieved a higher performance compared to the other on particular call classes; the idea was to use this knowledge to influence the vote on particular call classes.

The training results of the SRNs and those of the SVMs generated from training the classifiers both in parallel and in sequence were analysed; the F-measure was used as the comparison measure. From the analysis the SRNs were unable to classify examples associated with call class 18. The SVMs trained in parallel and in

[2]The SVM outputs cannot be used as inputs to re-train the SRN because the SVM produces only one output for each sequence so the result is the result for the whole sequence not for each word in the sequence as is the case for the SRN and the SRN needs an input for each unit in the input layer.

sequence outperformed the SRNs on call class 18 with the exception of the sigmoid SVM classifier (trained in parallel) on three cross-validated call sets. Based on this analysis call class 18 was selected as a candidate for improving the performance of the combined classifiers.

The analysis of the SVM classifier results, for the classifiers trained in sequence, identified a second possible candidate call class for influencing the performance of the combined classifiers; this was call class 15. The performance of the SVMs on call class 15 was varied and suggested diversity among the classifiers. Even with the possibility of diversity among the SVM classifiers this candidate call class may only produce additional improvement in performance on particular combinations. A decision was made to select call class 15 as a candidate for both sets of classifier combinations so that the results were directly comparable. The introduction of a casting vote for the SVM classifiers on two particular call classes was an attempt to further improve the performance of the six hybrid classifiers that had outperformed the individual SRN and again all combinations were re-tested using the approach described in Section 5.4.

5.4 Classifier Combination Using a Casting Vote

For this third approach to classifier combination the voting process was amended to allow the SVM classifiers to have a *casting vote* on decisions involving two particular call classes, *class 15* and *class 18*. The idea of a casting vote is similar to the idea of dynamic classifier selection where the classifier that was most effective on validation samples is selected and its judgment adopted. In this task the most effective classifier was selected based on performance on the training set because there was no separate validation set. The training and test sets were created using 9-fold cross-validation and therefore the hybrid classifiers could be referred to as a cross-validated committee (Dietterich 1997). Here the term casting vote is used to indicate that the vote of the selected classifier is used only when the majority of classifiers have failed to reach a majority decision on class 15 and class 18 call classes, as illustrated in Figure 8.

5.4.1 Evaluation of Combined Classifier Performance: Casting Vote

The 21 hybrid classifiers investigated were the same as in the previous sections. The results reported are the average results on the nine test sets. An examination of the F-measure results produced by the hybrid classifiers after the SVM had a casting vote on class 15 and class 18 call classes revealed that the number of hybrid classifiers producing an increase in performance on the individual SRN had increased. For both the classifiers trained in parallel and in sequence nine hybrid classifiers produced higher F-measure results, reported in Table 8.

The hybrid classifier of two SRNs and one SVM (sigmoid) trained in sequence produced the highest F-measure test results. Until the introduction of the casting vote this hybrid classifier had not outperformed the individual SRNs using either of the two previous approaches. These results indicate that the largest increase in performance was achieved by what appeared to be the poorest performing SVM suggesting that a poor performance in training created a more diverse classifier for combination. The combination now producing the highest result was two SRNs and a sigmoid SVM classifier; previously it was two SRNs and an RBF SVM classifier. The change to the combination using a sigmoid SVM classifier probably happened as a result of the implied diversity of the sigmoid SVM classifier exploited by the use of a casting vote.

Based on the analysis presented in Section 5.3.2, for both call classes 15 and 18, knowledge about the performance of the individual call classes on the training sets was transferred to the test sets. The analysis suggested that the SVMs were able to classify examples associated with call class 18 unlike the SRNs. Therefore it is probable that all the combinations benefited from the SVMs' ability to classify these examples exploited by the use of a casting vote. However, on call class 15 the overall performance of the individual SVMs was lower than that of the SRNs. As a result the benefit of the casting vote on call class 15 may have varied depending on the combination.

6 Analysis and Discussion

This paper has described an approach for spoken language classification, using SRNs and SVMs to create hybrid classifiers demonstrating their potential for use in the task of call classification. Based on the demonstrated potential and with further development the hybrid classifiers could be used for other classification tasks where currently individual classifiers are used. Three approaches to combining the classifiers were discussed: (1) classifier combination after training the classifiers in parallel, (2) classifier combination after training the

classifiers in sequence, and (3) classifier combination using a casting vote where the classifiers had previously been trained in parallel and in sequence.

For the classifiers trained in parallel only a small improvement in performance was achieved by the hybrid classifier on the performance of the best individual SRN until the casting vote was introduced; a combination of three SRNs performed equally well. For the classifiers trained in sequence a hybrid classifier of two SRNs and an RBF SVM produced the largest improvement in performance over the individual SRN suggesting diversity in the classifiers created by training the SVMs using a different strategy. This hybrid classifier, two SRNs and an RBF SVM, also outperformed the combination of three SRNs. For the classifiers trained in parallel and the associated hybrid classifiers, exploited by a casting vote, the largest improvement in performance was achieved by the combination of three SRNs, an RBF SVM, and a polynomial SVM. Finally, for the classifiers trained in sequence and the associated hybrid classifiers, exploited by a casting vote, the combination of only two SRNs and a sigmoid SVM was required to produce an improvement in performance over the individual SRNs and all other hybrid classifiers.

The results reported were evidence that a hybrid classifier of two SRNs and one SVM outperformed other hybrid classifiers and that the inclusion of additional classifiers provided no additional benefit to performance. The results also revealed that combining the classifiers after training in sequence and using a casting vote produced the highest improvement in performance. This means that the SRN was trained on the corpus first and then the SVM was trained on the outputs of the SRN which were used as inputs for the SVM. The reason why these combinations appear to be more successful is that the SRN acts as a pre-processor for the SVM. So, it is possible that knowledge learned by the SRN during training that is stored in the outputs helps the SVM during training and therefore what is learned by the SVM increases its ability to classify unseen examples. As a result when the outputs from both classifiers are combined there is an improved likelihood of the classifiers being able to classify more examples because the SVM could have additional knowledge to improve its performance on examples that were previously problematic for either or both of the classifiers individually.

The SRN and SVM hybrid combinations are effective because the weakness of one classifier on some call classes is compensated for by the stronger performance of the other classifier on these call classes. This is demonstrated by the introduction of the casting vote for the SVM classifier on call classes 15 and 18. These particular call classes were problematic for the SRN and it was unable to classify examples belonging to these call classes whereas this was not the case for the SVM. By introducing the casting vote for the SVM on these particular call classes the SVM was able to improve the classification performance on them and as a result improve the overall classification performance. This strategy of training the classifiers using different inputs, that is training the SVM classifiers on the outputs of the SRN, appears to create diverse classifiers because the classifiers are training on different representations of the corpus. This potential diversity can be exploited by using a casting vote. The casting vote provides the strongest performing classifier, on particular call classes, with the opportunity to use its “expertise” to influence the voting on these call classes thereby improving the overall classification performance.

7 Conclusions

This work has demonstrated that call classification can be achieved using a hybrid combination of simple recurrent networks and support vector machines and makes a contribution to the field of classification using a large, unique corpus of spontaneous spoken language. From the perspective of connectionist networks it has been demonstrated that an SRN can be used for spoken language classification. This may be more broadly applicable to language processing tasks in general because it highlights issues that may impact on how language processing tasks could be addressed in the future. For example, the use of continuous utterances which could be equated to longer representations for other language processing tasks or domains, the idea that it may not always be necessary to manipulate the data, and using a flat rather than a deep interpretation may prove beneficial to other language processing tasks. From the viewpoint of SVMs it has been demonstrated that there is potential in support vector machines for spoken language analysis. It has also been demonstrated that the improvement in performance produced by combining classifiers supports the claim that a combination of techniques can provide an improvement in performance in a difficult spoken language classification task. Therefore considering hybrid techniques to address a problem may also be beneficial and useful for future developments in the fields of classification and language processing.

References

Abbott, D.W. (1999), "Combining Models to Improve Classifier Accuracy and Robustness", *Proceedings of International Conference on Information Fusion - Fusion 1999*, Sunnyvale, CA, pp. 1–7.

Abney, S. (1996), "Statistical Methods and Linguistics", In: Klavans, J. and Resnik, P. (eds) *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, MIT Press, Cambridge, MA, pp. 1–27.

Bishop, C.M. (1995), *Neural Networks for Pattern Recognition*, Oxford University Press, UK.

Burton-Roberts, N. (1986), *Analysing Sentences. An Introduction to English Syntax*, Longman Group UK Ltd, England.

Chapelle, O. and Vapnik, V. (2000), "Model Selection for Support Vector Machines", In: Solla, S., Leen, T., and Muller, K-R. (eds) *Advances in Neural Information Processing Systems 12*, MIT Press, Cambridge, MA, pp. 230–236.

Clark, H.H. (1996), *Using Language*, Cambridge University Press, UK.

Cooley, R. (1999), "Classification of News Stories Using Support Vector Machines", *IJCAI '99 Workshop on Text Mining*, Stockholm, Sweden.

Core, M.G. and Schubert, L.K. (1999a), "Speech Repairs: A Parsing Perspective", *Proceedings of the ICPhS Satellite Meeting on Disfluency in Spontaneous Speech*, Berkeley, CA, pp. 47–50.

Core, M.G. and Schubert, L.K. (1999b), "A Model of Speech Repairs and Other Disruptions", *AAAI Symposium on Psychological Models of Communication in Collaborative Systems*, Cape Cod, MA, pp. 48–53.

Demirekler, M. and Altinçay, H. (2002), "Plurality voting-based multiple classifier systems: statistically independent with respect to dependent classifier sets", *Pattern Recognition*, 35(11):2365–2379.

Dietterich, T.G. (2000), "Ensemble Methods in Machine Learning", *Lecture Notes in Computer Science 1857*, Springer-Verlag, New York, pp. 1-15.

Dietterich, T.G. (1998), "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms", *Neural Computation* 10, pp. 1895–1923.

Dietterich, T.G. (1997), "Machine Learning Research: Four Current Directions", *AI Magazine*, 18(4):97–136.

Duin, R.P.W. (2002), "The Combining Classifier: To Train or Not to Train", In: Kasturi, R., Laurendeau, D., and Suen, C. (eds) *Proceedings of 16th International Conference on Pattern Recognition Vol. II*, Quebec, pp. 765-770.

Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998), "Inductive Learning Algorithms and Representations for Text Categorization", *Proceedings of ACM-CIKM98*, pp. 148–155.

Durston, P.J., Farrell, M., Attwater, D., Allen, J., Kuo, H-K.J., Afify, M., Fosler-Lussier, E., and Lee, C-H. (2001), "OASIS Natural Language Call Steering Trial", *Proceedings of Eurospeech* Vol. 2, pp. 1323–1326.

Elman, J.L., Bates, E.A., Johnson, M.H., Karmiloff-Smith, A., Parisi, D., and Plunkett, K. (1996), *Rethinking Innateness*, MIT Press, Cambridge, MA.

Elman, J.L. (1991) "Distributed representations, simple recurrent networks, and grammatical structure", *Machine Learning*, vol. 7, pp. 195–225.

Feng, J. and Williams, P. (2001), "The Generalization Error of the Symmetric and Scaled Support Vector Machines", *IEEE Transactions on Neural Networks*, 12(5):1255–1260.

Giacinto, G. and Roli, F. (2001), "An Approach to the Automatic Design of Multiple Classifier Systems", *Pattern Recognition Letters*, 22(1):25–33.

Giacinto, G., Roli, F., and Fumera, G. (2000), "Design of Effective Multiple Classifier Systems by Clustering of Classifiers", *Proceedings of ICPR 2000 15th International Conference on Pattern Recognition*, Barcelona, Spain, pp. 160–163.

Hansen, L.K. and Salmon, P. (1990), "Neural Network Ensembles", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001.

Heeman, P.A. and Allen, J.F. (1999), "Speech Repairs Intonational Phrases and Discourse Markers: Modeling Speakers' Utterances in Spoken Dialogue", *Computational Linguistics*, 25(4):527–571.

Ho, T.K. (2002), "Multiple Classifier Combination: Lessons and Next Steps", In: Kandel, A. and Bunke, H. (eds) *Hybrid Methods in Pattern Recognition*, World Scientific, pp. 171–198.

Joachims, T. (1999), "Making Large-Scale SVM Learning Practical", In: Schölkopf, B., Burges, C., and Smola, A. (eds) *Advances in Kernel Methods: Support Vector Learning*, MIT Press, Cambridge, MA, pp. 169–184.

Kuncheva, L.I. (2002), "A Theoretical Study on Six Classifier Fusion Strategies", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):281–286.

LeCun, Y., Bottou, L., Orr, G., and Muller, K-R. (1998), "Efficient BackProp", In: Orr, G. and Muller, K. (eds) *Neural Networks: Tricks of the Trade*, Springer, pp. 9–50.

Moghaddam, B. and Yang, M-H. (2001), "Sex with Support Vector Machines", In: Leen, T.K., Dietterich,

T.G., and Tresp, V. (eds) *Advances in Neural Information Processing Systems 13*, MIT Press, Cambridge, MA, pp. 960–966.

Oh, S-B. (2003), "On the Relationship Between Majority Vote Accuracy and Dependency in Multiple Classifier Systems", *Pattern Recognition Letters*, 24(1–3):359–363.

Opper, M. and Urbanczik, R. (2001), "Universal learning curves of support vector machines", *Physical Review Letters*, 86(19):4410–4413.

Platt, J.C. (1998), "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines", *Technical Report MSR-TR-98-14*.

Salton, G. and McGill, M. (1983), *Introduction to Modern Information Retrieval*, McGraw Hill, New York.

Schölkopf, B., Burges, C., and Vapnik, V. (1995), "Extracting Support Data for a Given Task", In: Fayyad, U.M. and Uthurusamy, R. (eds) *Proceedings First International Conference on Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA, pp. 252–257.

Sebastiani, F. (2002), "Machine Learning in Automated Text Categorization", *ACM Computing Surveys*, 34(1):1–47.

Stitson, M.O., Weston, J.A.E., Gammerman, A., Vovk, V., and Vapnik, V. (1996), "Theory of Support Vector Machines", *Technical Report CSD-TR-96-17*.

Tumer, K. and Ghosh, J. (1996), "Analysis of Decision Boundaries in Linearly Combined Neural Classifiers", *Pattern Recognition*, 29(2):341–348.

Van Rijsbergen, C.J. (1979), *Information Retrieval. 2nd edition*, Butterworths, London.

Wermter, S. (2000), "Neural Fuzzy Preference Integration using Neural Preference Moore Machines", *International Journal of Neural Systems*, 10(4):287–309.

Wermter, S., Panchev, C., and Arevian, G. (1999), "Hybrid Neural Plausibility Networks for News Agents", *Proceedings of the National Conference on Artificial Intelligence*, Orlando, USA, pp. 93–98.

Wermter, S. and Weber, V. (1997), "SCREEN: Learning a Flat Syntactic and Semantic Spoken Language Analysis", *Journal of Artificial Intelligence Research*, 6(1):35–85.

Wermter, S. (1995), *Hybrid Connectionist Natural Language Processing*, Chapman and Hall Thomson International, London, UK.

Table 1. Example calls from the corpus. Note: @@ indicates primary move boundaries.

Call
oh hi @@ could I book a um an alarm call please @@
hello @@ i'd like to make a chargecard call please @@
I'm sorry to trouble you @@ I'm I'm trying to trace a call which came through about ten minutes ago @@

Table 2. Breakdown of utterances in training and test set. Note: For illustration purposes not all call classes are shown.

8,441 utterances Total of 7,441 utterances in Training set: Call sets 2-9 Total of 1,000 utterances in Test set: Call set 1								
Call class	class 1	class 2	class 3	class 4	class 5	class 6	class 7	class *n*
in train set	2325	126	502	39	775	317	64	...
in test set	320	14	62	4	114	43	8	...

Table 3. Example of semantic vectors. Note: For illustration purposes not all call classes are shown. There are 19 call classes used in this study.

Word	Call Class							
	class 1	class 2	class 3	class 4	class 5	class 6	class 7	class *n*
COULD	0.06	0.05	0.06	0.03	0.06	0.05	0.07	...
I	0.05	0.05	0.05	0.04	0.06	0.07	0.09	...
MAKE	0.01	0.51	0.02	0.00	0.02	0.01	0.02	...
A	0.12	0.07	0.03	0.04	0.04	0.09	0.08	...
TRANSFER	0.00	**0.87**	0.00	0.00	0.00	0.04	0.00	...
CHARGE	0.00	**0.86**	0.00	0.00	0.00	0.00	0.00	...
CALL	0.01	0.18	0.25	0.19	0.02	0.01	0.08	...
PLEASE	0.03	0.11	0.11	0.04	0.05	0.04	0.02	...

Table 4. Example tagged call.

(CALL hi could you tell me what erm code 6 0 9 6 is please)CALL
(CLASS @@, class 6)CLASS
(MOVE type 2)MOVE
)END

Table 5. Example neural network input sequences. Note: For illustration purposes not all call classes are provided.

$\overbrace{Sequence\ number}$						
(*SEQ 20*	*could you check the line for me*					
COULD	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0.07	0.03	0.04	0.05	0.04 ...
YOU	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0.06	0.01	0.01	0.04	0.05 ...
CHECK	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0.55	0.00	0.00	0.00	0.01 ...
THE	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0.05	0.00	0.02	0.00	0.00 ...
LINE	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0.21	0.00	0.00	0.00	0.00 ...
FOR	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0.08	0.00	0.05	0.00	0.00 ...
ME	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0.07	0.01	0.01	0.02	0.01 ...
)						
	$\underbrace{category\ output\ representation}$	$\underbrace{word\ input\ representation}$				

Table 6. Comparison of average F-measure test results on the nine test sets of combinations of the simple recurrent network (SRN) and support vector machines (SVM) after training the classifiers in parallel.

F-measure		
Hybrid Classifiers		SRN
2 SRNs & RBF SVM	81.4	80.9
2 SRNs & Polynomial SVM	81.2	80.9
3 SRNs & RBF SVM	81.0	80.9
3 SRNs & RBF & Sigmoid SVM	81.2	80.9
3 SRNs & RBF & Polynomial SVM	81.4	80.9

Table 7. Comparison of average F-measure test results on the nine test sets of combinations of the simple recurrent network (SRN) and support vector machines (SVM) after training the classifiers in sequence.

F-measure		
Hybrid Classifiers		SRN
2 SRNs & RBF SVM	82.0	80.9
2 SRNs & Polynomial SVM	81.5	80.9
3 SRNs & RBF SVM	81.2	80.9
3 SRNs & RBF & Sigmoid SVM	81.7	80.9
3 SRNs & RBF & Polynomial SVM	81.9	80.9
3 SRNs & Sigmoid & Polynomial SVM	81.2	80.9

Table 8. Comparison of average F-measure test results on the nine test sets of combinations of the simple recurrent network (SRN) and support vector machines (SVM) after training the classifiers in parallel and in sequence and using a casting vote.

F-measure			
Hybrid Classifiers	Trained in Parallel	Trained in Sequence	SRN
2 SRNs & RBF SVM	82.9	85.3	80.9
2 SRNs & Sigmoid SVM	82.5	85.6	80.9
2 SRNs & Polynomial SVM	82.7	85.3	80.9
3 SRNs & RBF SVM	82.5	85.0	80.9
3 SRNs & Sigmoid SVM	82.5	85.4	80.9
3 SRNs & Polynomial SVM	82.3	85.2	80.9
3 SRNs & RBF & Sigmoid SVM	83.8	85.4	80.9
3 SRNs & RBF & Polynomial SVM	84.0	85.4	80.9
3 SRNs & Sigmoid & Polynomial SVM	83.4	84.7	80.9

Figure 1. Representation of a simple recurrent network.

Figure 2. Average of the test results for the simple recurrent networks on nine test sets.

Figure 3. Representation of the optimal separating hyperplane. (After Dumais et al. (1998); Platt (1998)).

Figure 4. Example of the input and the output for the support vector machine.

Figure 5. Average results of support vector machines on nine test sets.

Figure 6. Top-level representation of the overall architecture of the system (SRN - simple recurrent network, SVM - support vector machine).

Figure 7. Instance of the voting process indicating *retrieved and relevant* example.

Figure 8. Representation of the combination process (SRN - simple recurrent network, SVM - support vector machine).

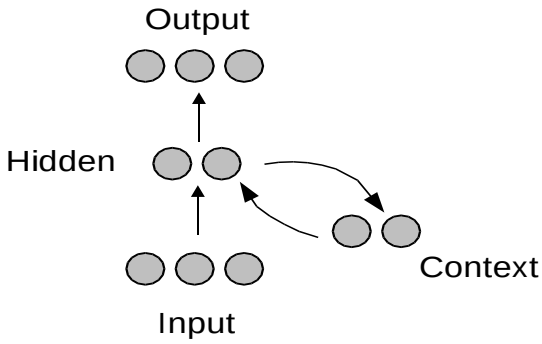

Figure 1.

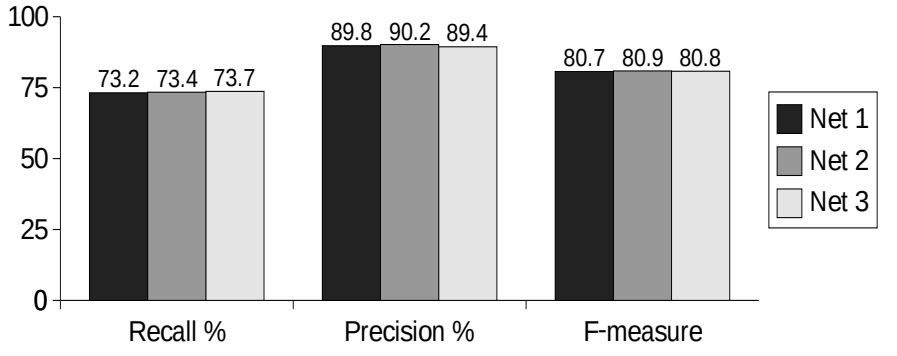
100
75
50
25
0
73.2
73.4
73.7
89.8
90.2
89.4
80.7
80.9
80.8
Net 1
Net 2
Net 3
Recall %
Precision %
F-measure

Figure 2.

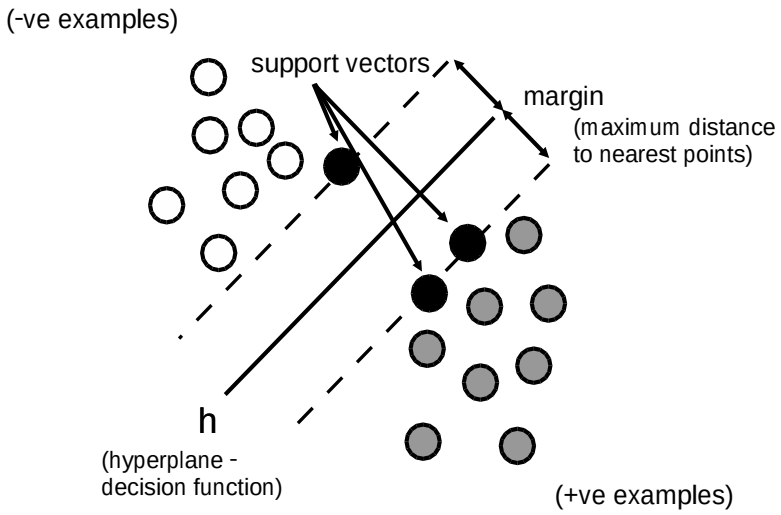
(-ve examples)
support vectors
margin
(maximum distance
to nearest points)
h
(hyperplane -
decision function)
(+ve examples)

Figure 3.

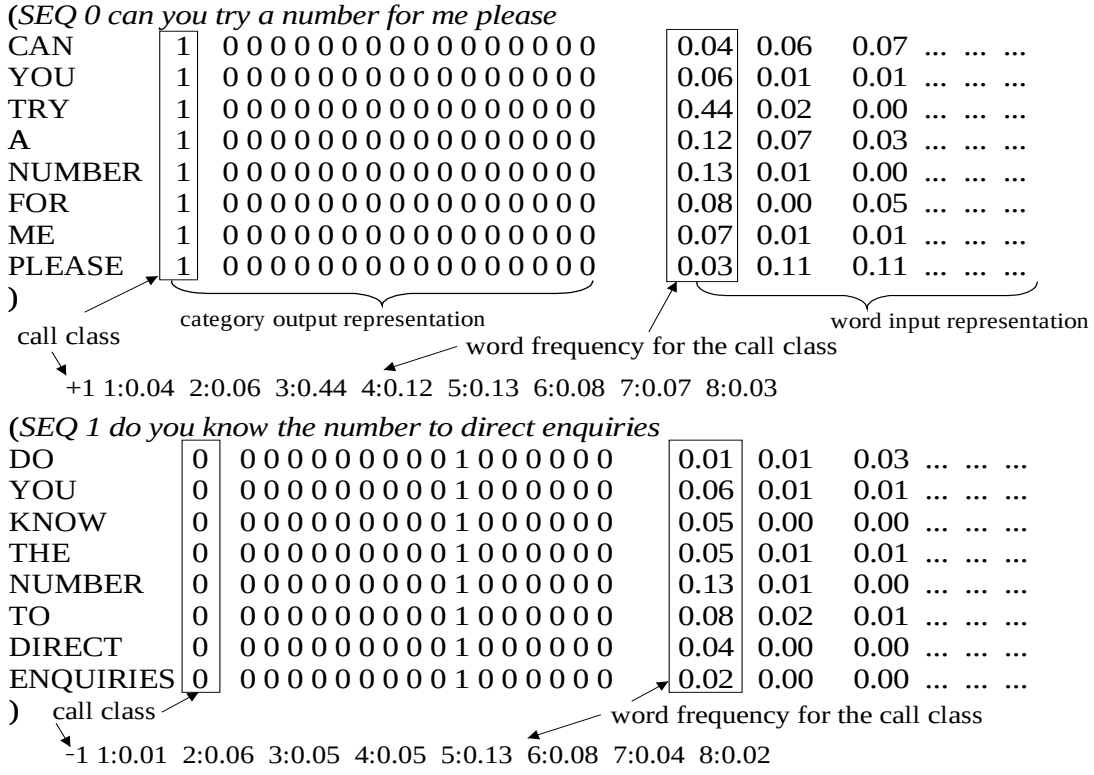
(SEQ 0 can you try a number for me please
CAN 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.04 0.06 0.07 ... ...
YOU 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.06 0.01 0.01 ... ...
TRY 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.44 0.02 0.00 ... ...
A 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.12 0.07 0.03 ... ...
NUMBER 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.13 0.01 0.00 ... ...
FOR 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.08 0.00 0.05 ... ...
ME 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.07 0.01 0.01 ... ...
PLEASE 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.03 0.11 0.11 ... ...
)
call class
category output representation
word input representation
word frequency for the call class
+1 1:0.04 2:0.06 3:0.44 4:0.12 5:0.13 6:0.08 7:0.07 8:0.03
(SEQ 1 do you know the number to direct enquiries
DO 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0.01 0.01 0.03 ... ...
YOU 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0.06 0.01 0.01 ... ...
KNOW 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0.05 0.00 0.00 ... ...
THE 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0.05 0.01 0.01 ... ...
NUMBER 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0.13 0.01 0.00 ... ...
TO 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0.08 0.02 0.01 ... ...
DIRECT 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0.04 0.00 0.00 ... ...
ENQUIRIES 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0.02 0.00 0.00 ... ...
)
call class
word frequency for the call class
-1 1:0.01 2:0.06 3:0.05 4:0.05 5:0.13 6:0.08 7:0.04 8:0.02

Figure 4.

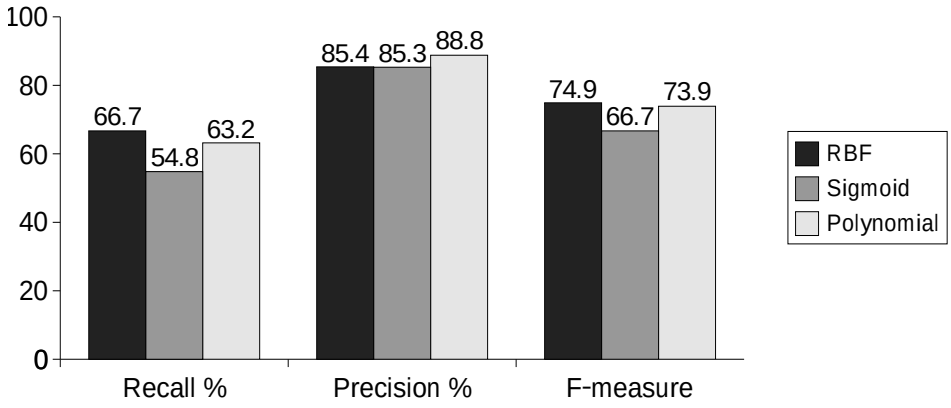
100
80
60
40
20
0
66.7
54.8
63.2
85.4
85.3
88.8
74.9
66.7
73.9
Recall %
Precision %
F-measure
RBF
Sigmoid
Polynomial

Figure 5.

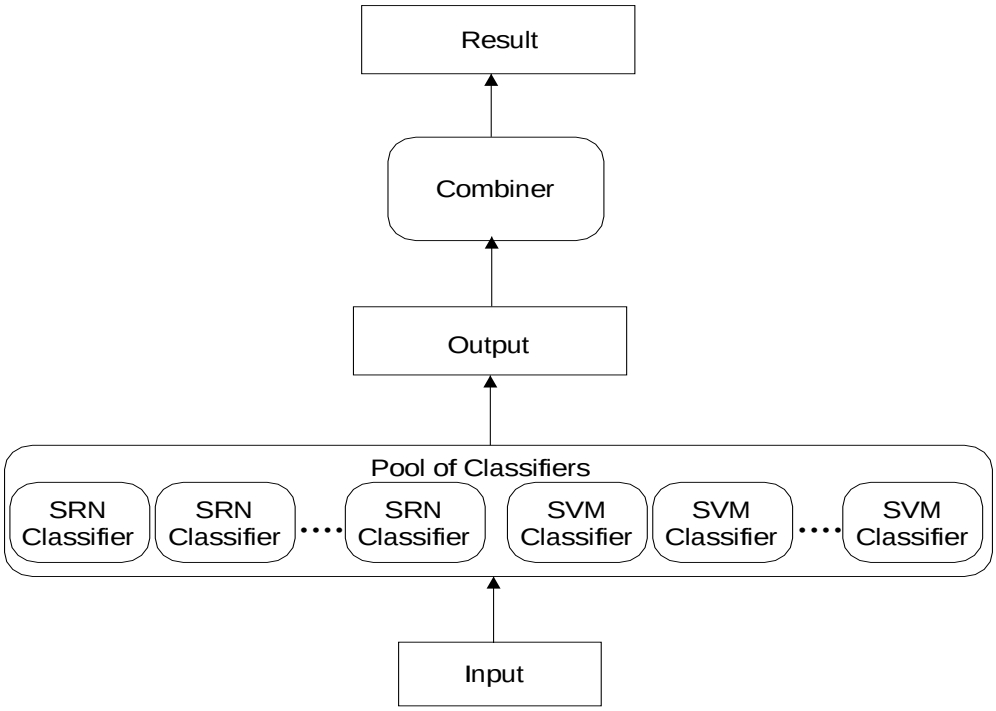
Result
Combiner
Output
Pool of Classifiers
SRN Classifier
SRN Classifier
SRN Classifier
SVM Classifier
SVM Classifier
SVM Classifier
Input

Figure 6.

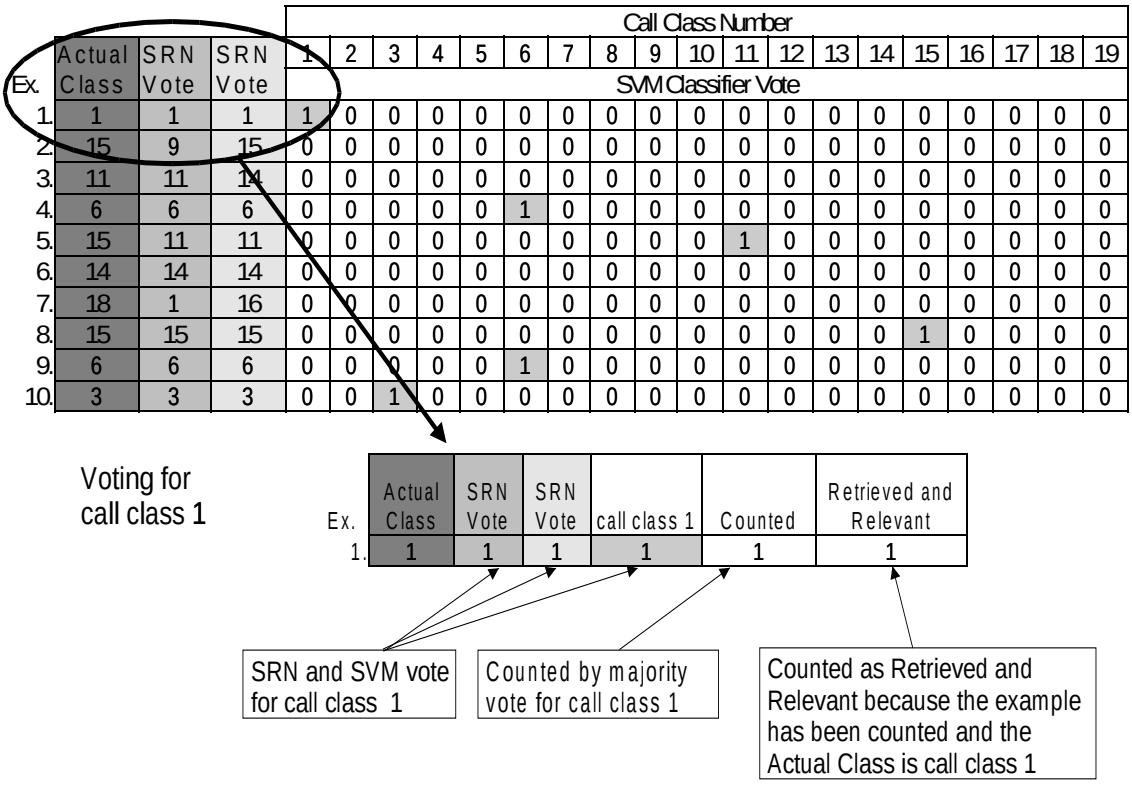
Call Class Number
Ex.
Actual Class
SRN Vote
SRN Vote
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
SVM Classifier Vote
1. 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2. 15 9 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3. 11 11 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4. 6 6 6 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
5. 15 11 11 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
6. 14 14 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7. 18 1 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
8. 15 15 15 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
9. 6 6 6 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
10. 3 3 3 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Voting for call class 1
Ex.
Actual Class
SRN Vote
SRN Vote
call class 1
Counted
Retrieved and Relevant
1. 1 1 1 1 1 1
SRN and SVM vote for call class 1
Counted by majority vote for call class 1
Counted as Retrieved and Relevant because the example has been counted and the Actual Class is call class 1

Figure 7.

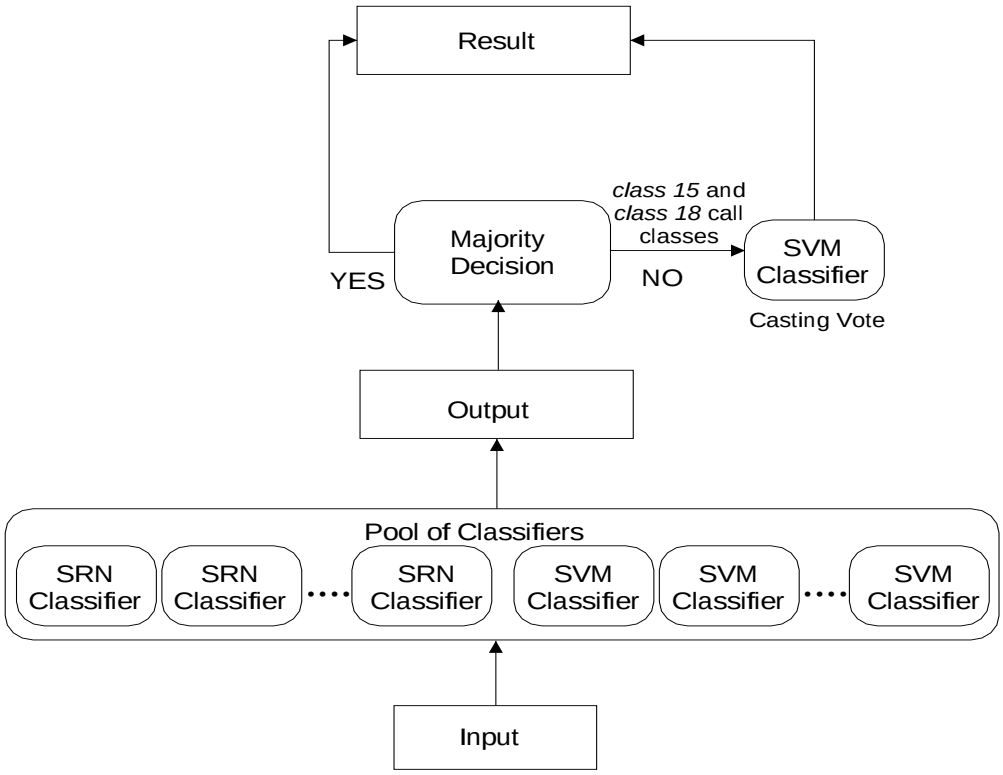
Result
class 15 and
class 18 call
classes
Majority
Decision
YES
NO
SVM
Classifier
Casting Vote
Output
Pool of Classifiers
SRN
Classifier
SRN
Classifier
SRN
Classifier
SVM
Classifier
SVM
Classifier
SVM
Classifier
Input

Figure 8.