Comparing Support Vector Machines, Recurrent Networks and Finite State Transducers for Classifying Spoken Utterances *

Sheila Garfield and Stefan Wermter

University of Sunderland, School of Computing and Technology,
Centre for Hybrid Intelligent Systems, Sunderland, SR6 0DD, United Kingdom
{stefan.wermter,sheila.garfield}@sunderland.ac.uk
http://www.his.sunderland.ac.uk/

1 Abstract

This paper describes new experiments for the classification of recorded operator assistance telephone utterances. The experimental work focused on three techniques: support vector machines (SVM), simple recurrent networks (SRN) and finite-state transducers (FST) using a large, unique telecommunication corpus of spontaneous spoken language. A comparison is made of the performance of these classification techniques which indicates that a simple recurrent network performed best for learning classification of spontaneous spoken language in a robust manner which should lead to their use in helpdesk call routing.

2 Introduction

Spontaneous spoken language is disfluent [1] and as a result errors in recognition of speech input can arise due to background noise, speaking style, disfluencies, out-of-vocabulary words, parsing coverage or understanding gaps [2]. Spontaneous speech also includes filled pauses and partial words. Spoken dialogue systems must be able to deal with these as well as other discourse phenomena such as anaphora and ellipsis, and ungrammatical queries [2, 3].

A general research question arises whether certain symbolic, statistical or neural network methods can be used effectively to classify faulty telephone utterances. In order to address this question we have chosen symbolic finite state transducers, statistical support vector machines and simple recurrent neural networks. So far there has been little work on comparing and evaluating such techniques on substantial real world corpora although we believe that such a study is crucial to suggest more effective techniques and architectures for the future.

In this paper we describe an approach to the classification of recorded operator assistance telephone utterances. We describe experiments in a real-world scenario utilising a large, unique corpus with 4 000 utterances of spontaneous spoken

* This research has been partially supported under grant ML846657 to Professor S. Wermter. The authors thank Mark Farrell and David Attwater of BTexact Technologies for their helpful discussions.

language. The paper is structured as follows: Section 3 provides an overview of the corpus used; Sections 4 and 5 outline the experimental work using SVMs and FSTs and Section 6 is an evaluation and comparison of these techniques with the SRN; and finally Section 7 draws together conclusions.

3 Description of the Helpdesk Corpus

Our task is to learn to classify real incoming telephone utterances into a set of service level classes. For this task a corpus from transcriptions of recorded operator assistance telephone calls was used [4]. Examination of the utterances reveals that the callers use a wide range of language to express their problem, enquiry or to request assistance [5].

1. "*could I book a wake up call please*"
2. "*could you check for conversation please*"
3. "*I'm in on my own and I've just got a phone call and I picked it up and it was just gone and it happened yesterday as well at the same time*"

3.1 Call Transcription

A corpus based on transcriptions of the first utterances of callers to the operator service is the focus of the investigation. A number of service levels or call classes, primary move types and request types [4] were identified after analysis of the utterances. As shown by the example in Table 1, the *call class* is the classification of the utterance and is associated with the service that the caller requires. The *primary move* is a subset of the first utterance, it is like a dialogue act and gives an indication of which dialogue act is likely to follow the current utterance. The *request* type identifies whether the request is explicit or not [6].

Table 1. Example call from the corpus

Call	*Call Class*	*Primary Move Type*	*Request Type*
yeah could I book a wake up call please	alarm call	action	explicit

Four separate call sets of 1 000 utterances each were used in this study. The call sets are split so that 80% of utterances are used for training and 20% of utterances used for testing. The average length of an utterance in the training set is 16.05 words and in the test set the average length of an utterance is 15.52 words. An illustrative example of the breakdown of utterances is given in Table 2, however for illustration not all call classes are shown. The part of the utterance identified as the primary move was used for both the training and test sets.

Table 2. Breakdown of utterances in training and test sets from call set 1. Note: For illustration purposes not all classes are shown

1 000 utterances
Total of 712 utterances in Training set
Total of 205 utterances in Test set
Total of 83 utterances not used since they did not contain a primary move

Classes:	class 1	class 2	class 3	class 4	class 5	class 6	class 7	class 8	class 9	class n
in train set:	261	11	41	3	85	32	6	16	28	...
in test set:	59	3	21	1	29	11	2	4	7	...

3.2 Semantic Vector Representation

The experiments use a semantic vector representation of the words in a lexicon [7]. These vectors represent the frequency of a particular word occurring in a call class and are independent of the number of examples observed in each class. The number of calls in a class can vary substantially. Therefore we *normalise* the frequency of a word w in class c_i according to the number of calls in c_i (2). A *value* $v(w, c_i)$ is computed for each element of the semantic vector as the *normalised* frequency of occurrences of word w in semantic class c_i, divided by the *normalised* frequency of occurrences of word w in all classes. That is:

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

where:

$$v(w, c_i) = \frac{Normalised\ frequency\ of\ w\ in\ c_i}{\sum\limits_j Normalised\ frequency\ for\ w\ in\ c_j},\ j \in \{1, \cdots n\}\ . \tag{2}$$

4 Support Vector Machines

A support vector machine is a binary classifier which divides the problem space into two regions via a *dividing line* or *hyperplane*. This hyperplane separates positive examples from negative examples with a maximum margin [8, 9, 10] between it and the nearest data point of each class. SVMs can deal with non-linearly separable classes and can handle noisy data. SVMs can learn or generalise from labelled training data and consequently can be automatically constructed.

Classifiers were constructed [11] with different kernel functions: polynomial, RBF and sigmoid [9, 12]. For each of these kernel functions we performed two series of experiments. In the first series the value of tuning parameter C, the error penalty, was chosen randomly; in the second series this parameter was selected automatically. Several values for C were determined empirically; the aim was to select a value that reduced the test error to a minimum while at the same time constructing a classifier that was able to perform successfully on individual classes. Other parameters were kept constant, that is, the classifiers were constructed using the same values except the error penalty C.

The input sequences generated are composed of feature-value pairs, for example, 1:0.106232, where the 'feature', in this case '1', is the position of the word in the utterance and the 'value', '0.106232', is the frequency of occurrence of that word for the particular class under investigation, (see also Section 3.2).

A classifier was constructed [11] for each class in a *one-versus-all* approach, that is, the training and test sets used positive examples for the particular class in question and negative examples of all the other classes. The classifier in each case was trained and then tested on the utterance sequences generated from four call sets used in [13]. Table 3 shows the overall F-score performance (combined recall and precision [14]) of each of the trained classifiers on the four call sets. We will later analyse and compare this performance in Section 6.

Table 3. Training and test set. F-score results of SVM on 4 000 utterances

	Training Set			*Test Set*		
	RBF	Sigmoid	Polynomial	RBF	Sigmoid	Polynomial
Call set 1	89.31	75.70	77.00	56.71	57.82	64.36
Call set 2	89.89	81.06	81.77	58.46	60.51	63.82
Call set 3	88.92	79.78	78.50	57.88	59.58	66.66
Call set 4	91.23	84.09	83.92	57.57	60.51	65.26

5 Finite-State Machines

A finite-state machine is a formal machine that consists of states, transitions, one start state, any number of final states and an alphabet [15, 16]. While their representation and their output is both easily understood and interpreted the foundation of a finite-state machine is the regular expression and these must be manually pre-specified to encode an utterance classification.

In order to create the regular expressions, the utterances in the call set are symbolically tagged to enable identification of possible patterns or strings and substrings in the utterances. This means that each word in the utterance is tagged with a symbolic representation of the class associated with that word based on the highest frequency of occurrence as shown by the example utterance "*how do I cancel it or can't I*" in Table 4. Thus in the lexicon, the word "how" is most frequently associated with the "class 4" class based on the frequency of occurrence. These symbolic tags represent the sequence for a specific utterance. The higher frequency of class 4 would indicate that this utterance is classified as class 4.

The same 4 000 utterances were used as in the previous experiment and transducers were constructed for each class. Transducers were constructed for

Table 4. Example sequence

how	do	I	cancel	it	or	can't	I
class 4	class 4	class 8	class 4	class 1	class 4	class 4	class 8

each class using regular expressions which were hand-crafted. The tagged training sets were used as reference sets whilst creating the regular expressions based on sequences of classes identified in the symbolic representation of the utterances. Once the transducers were constructed each was tested against the data sets for the other classes. The transducer produces an output for each sequence in the input file. A successful output is the semantic label of the class for which the transducer has been constructed while an unsuccessful output is '0'.

The overall F-score performance of the transducers on each of the four call sets is shown in Table 5.

Table 5. Training and test set. F-score results of FST on 4 000 utterances

	Training Set	*Test Set*
Call set 1	38.75	40.59
Call set 2	36.31	35.91
Call set 3	41.43	37.21
Call set 4	34.58	33.59

6 Evaluation and Comparison with Simple Recurrent Network

The overall test recall and precision rates for each of the SVM classifiers and the FST are shown in Figure 1 for each of the four call sets.

In an initial comparison of the three techniques SRN, SVM and FST the average recall and precision performance on the test set across three classes for each technique is shown in Figure 2. The SRN network and its performance has been described in more detail in [13] which is why we concentrated here on the description of SVM and FST in this paper. The performance of the SRN, SVM and the FST on individual call classes for recall and precision on the unseen test data for call set 1 is shown in Figure 3.

From the results on the test set, shown in Figure 3, the performance is varied across each technique and it is evident that no one technique performs well on every class and, in some cases, both the SRN and the SVM produce exactly the same recall result, for example, class 3 recall percentage. Generally, the performance of the SRN is better than that of the SVM and the FST. One possible reason why the performance of the SRN is better than that of the other

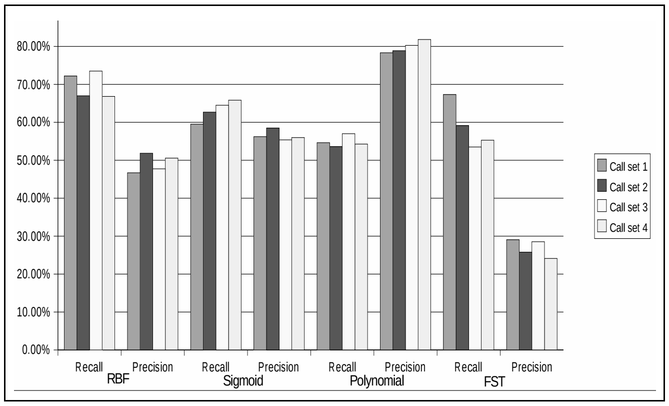

Fig. 1. Test results of SVM and FST

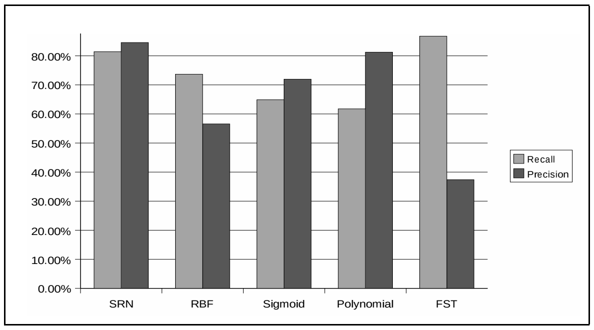

Fig. 2. Test set. Comparison of average performance on three classes for the SRN, SVM and FST

techniques is that the SRN is able to make use of sequential memory in the form of its context layer. That is, the information stored in the context layer does assist the network in assigning the correct class to an utterance.

However, in the case of class 3 the precision rate for both the SVM and the FST is below 50%. This is a possible indication that the method of representation of the semantic content of the utterances in this particular class is inadequate and does not allow either technique to consistently classify the utterances.

On the other hand the recall performance of the FST on this class is significantly better than that of the SVM. It can be concluded that the means by which the semantic content is presented to the FST generally enables this technique to perform more effectively with regard to recall than the SVM but in most cases not as well with regard to precision performance. It is evident that the hand-crafted regular expressions do not capture a fine enough level of granularity in the semantics of this type of utterance for the FST to accurately classify the utterances consistently when precision performance is considered.

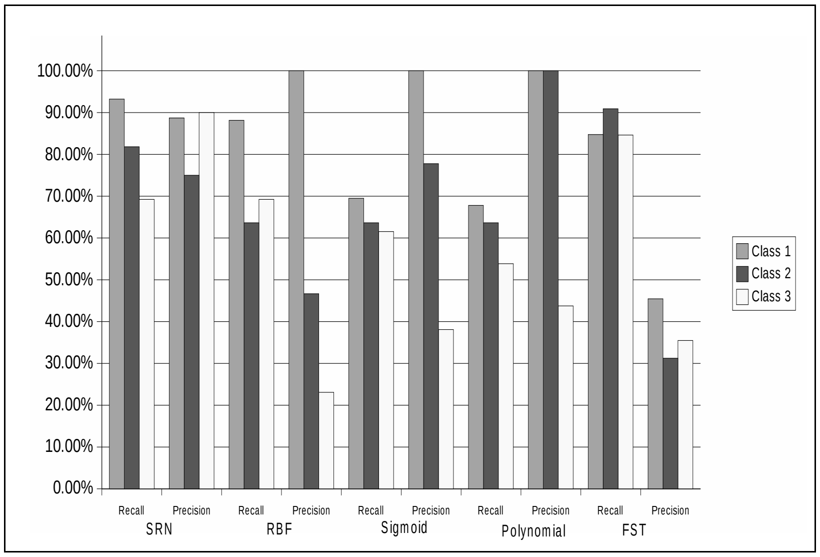

Fig. 3. Test set. Comparison of SRN, SVM and FST. Note: For illustration purposes three classes are shown

One factor to consider is the number of examples in each of the classes. The number of examples in class 1 is over 5 times greater than that of class 3. Consequently, if one or two examples are misclassified or not classified in a small number of examples the impact on the result is much greater and produces a greater difference to the result but the same is not true for a larger set of examples.

SVMs are a promising method for learning classification from examples. Classifiers can be automatically constructed and, depending on the task, achieve a smooth trade-off between recall and precision [17]. Finite-state machines can handle strings of any length and this is of particular significance in this task because the length of the caller's utterance is variable. However, the SRN overall outperformed both the SVM and FST and provided better overall results due to its context layer.

7 Conclusion

The SRN network was compared to the other approaches identified and the main result from this work is that the performance of the simple recurrent network is better when factors such as noise in the utterance and the number of classes are taken into consideration. However, a combination of techniques might yield even improved performance. The SVM allows a trade-off between recall and precision and as a result for classes that are problematic to the SRN the SVM might be a better alternative. Whilst the FST does not allow a trade-off between recall and precision again it could be used for those classes with very few and problematic examples because it can achieve almost perfect performance, that is 100%, on both recall and precision for very few examples. The drawback to this approach is that it requires hand-crafted regular expressions which are time-consuming to construct. One additional factor that must be considered is the number of examples available for training and testing as this does influence performance. Based on this detailed study of 4 000 utterances we conclude that an SRN-based architecture combined with SVM models is the most promising candidate to improve classification performance even further.

References

[1] Abney, S.: Statistical Methods and Linguistics. In: Klavans, J. and P. (eds): The Balancing Act. MIT Press, Cambridge, MA (1996)

[2] Glass, J.R.: Challenges for Spoken Dialogue Systems. Proceedings of IEEE ASRU Workshop. Keystone, CO (1999)

[3] Clark, H.: Speaking in Time. Speech Communication **36** (2002) 5–13

[4] Durston, P.J. and Kuo, J.J.K. et al.: OASIS Natural Language Call Steering Trial. Proceedings of Eurospeech, Vol2 (2001) 1323–1326

[5] Edgington, M., Attwater, D. and Durston, P.: OASIS - A Framework for Spoken Language Call Steering. Proceedings of Eurospeech '99 (1999)

[6] Attwater, D.: OASIS First Utterance Version 2.10 Release Note, 100CS:QUAL:REL:005 Version. (2000)

[7] Wermter, S., Panchev, C., Arevian, G.: Hybrid Neural Plausibility Networks for News Agents. Proceedings of the National Conference on Artificial Intelligence. Orlando, USA (1999) 93–98

[8] Chapelle, O. and Vapnik, V.: 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 (2000)

[9] Opper, M. and Urbanczik, R.: Universal Learning Curves of Support Vector Machines. Physical Review Letters **86**,19 (2001) 4410–4413

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

[11] Joachims, T.: 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 (1999)

[12] Moghaddam, B. and Yang, M-H.: 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 (2001) 960–966

[13] Garfield, S. and Wermter, S.: Recurrent Neural Learning for Helpdesk Call Routing. Proceedings of the International Conference on Artificial Neural Networks (2002) 296–301

[14] Van Rijsbergen, C.J.: Information Retrieval. 2nd edition. Butterworths, London. (1979)

[15] Roche, E. and Schabes, Y.: Finite-state Language Processing. MIT, London (1997)

[16] Jurafsky, D. and Martin, J.H.: Speech and Language Processing. Prentice Hall, New Jersey (2000)

[17] Dumais, S.: Using SVMs for text categorization. IEEE Intelligent Systems (1998) 21–23