An Application of Recurrent Neural Networks to Discriminative Keyword Spotting

Santiago Fernández[1], Alex Graves[1], and Jürgen Schmidhuber[1,2]

[1] IDSIA, Galleria 2, 6928 Manno-Lugano, Switzerland
[2] TU Munich, Boltzmannstr. 3, 85748 Garching, Munich, Germany

Abstract. The goal of keyword spotting is to detect the presence of specific spoken words in unconstrained speech. The majority of keyword spotting systems are based on generative hidden Markov models and lack discriminative capabilities. However, discriminative keyword spotting systems are currently based on frame-level posterior probabilities of sub-word units. This paper presents a discriminative keyword spotting system based on recurrent neural networks only, that uses information from long time spans to estimate word-level posterior probabilities. In a keyword spotting task on a large database of unconstrained speech the system achieved a keyword spotting accuracy of 84.5 %.

1 Introduction

The goal of keyword spotting is to detect the presence of specific spoken words in (typically) unconstrained speech. Applications of keyword spotting include audio indexing, detection of command words in interactive environments and spoken password verification. In general, it is most useful in domains where a full speech recogniser is cumbersome and unnecessary, partly because less than perfect detection rates are still very satisfactory.

Nonetheless, the same mathematical framework used in the majority of speech recognisers also forms the basis for keyword spotting systems. The typical keyword spotting system consists of a set of hidden Markov models (HMM), one for each keyword plus one or more filler models [1]. The filler models characterize non-keyword events in the speech signal, such as other words, background noises and silence. The approach is generative and therefore finds the sequence of models most likely to have produced the observations. The output of the system is post-processed before deciding on the presence or absence of keywords in the utterance [2]. First, the confidence of the predictions is estimated. This is typically done by computing the ratio of likelihoods between keyword model hypotheses and filler model hypotheses. Finally, a threshold level is applied to the confidence measure in order to achieve a compromise between the number of true and false positives predicted by the system.

In general, a discriminative approach to keyword spotting is more suitable, as it allows discrimination between keyword and non-keyword events, and also among similar keywords. In addition, posterior probabilities can directly be

J. Marques de Sá et al. (Eds.): ICANN 2007, Part II, LNCS 4669, pp. 220–229, 2007.

used as an effective confidence measure. Recently, discriminative keyword spotting systems have been explored in various papers. The most common approach [3,4,5] uses artificial neural networks (ANN) in the framework of hybrid ANN/HMM systems [6,7]. In these systems, keywords are modelled by concatenating phoneme models. A phoneme classifier (typically a multi-layer perceptron) estimates frame-level posteriors. These local posterior estimates are accumulated over the duration of a segment and, typically, the result is normalised by the length of the segment. Another approach to discriminative keyword spotting uses kernel machines and large margin classifiers with a set of models and feature functions consistent with hybrid ANN/HMM systems [8].

The approach presented in this paper attempts to model full keywords in the sequence data stream, while previous discriminative keyword spotting systems are based on sub-word units. It is based on recurrent neural networks (RNNs) trained with the connectionist temporal classification (CTC) objective function. This objective function allows artificial neural networks to map unsegmented sequential data onto a sequence of labels [9,10]. The proposed system uses information over long time spans, thereby providing non-local estimates of the a posteriori probability of the presence of either keyword or non-keyword events in the speech signal.

There is a plethora of different keyword spotting systems, often tailored to meet the requirements of particular tasks. However, there is little consensus on how to define benchmark tasks [11]. We have opted to tackle a realistic keyword spotting task in a large database of spontaneous and unconstrained speech where an HMM-based speech recogniser achieves a word accuracy of only 65 %.

The remainder of the paper is structured as follows. Section 2 provides a description of the system. Keyword spotting experiments are presented in section 3 and the results are shown in section 3.4. Section 4 offers a discussion on differences between previous approaches and the one presented in this paper, and gives directions for future work. Final conclusions are given in section 5.

2 Method

2.1 Outline

The network architecture selected for the keyword spotting task is the bidirectional long short-term memory recurrent neural network (BLSTM), which has shown good performance in a series of speech tasks [12,13]. The network is trained with the connectionist temporal classification (CTC) objective function [9,10].

The input to the recurrent network is the data sequence of a speech utterance. The network has a soft-max output layer with as many output units as keywords to be detected, plus one output unit associated with non-keyword events. The target for the training algorithm is a list (which may be empty) of keywords in the order in which they appear in the input speech utterance. No further constraints are applied to the system. In particular, the segmentation of the speech signal is not required and various pronunciation variants of the same keyword can appear in the data set.

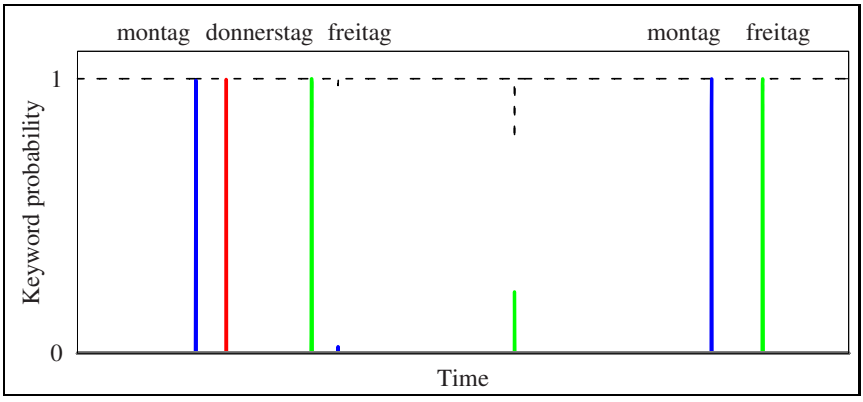

Fig. 1. Discriminant keyword spotting with BLSTM and CTC. When a keyword is detected the ANN produces a spike-like output with higher probability than the non-keyword output (dashed line).

Once the network has been trained, its output typically consists of a series of spikes corresponding to keyword events that have been detected in the input, separated by long periods of activation of the non-keyword output unit (see figure 1). The activation of every output unit at every time step is an estimate of the probability of detecting a particular keyword (or non-keyword event) at that time step.

2.2 BLSTM

The long short-term memory (LSTM) [14,15] is an RNN architecture designed to deal with long time-dependencies. It addresses the problem of the back-propagated error either blowing up or decaying exponentially for long time lags in conventional RNNs. The hidden layer of an LSTM network consists of a set of recurrently connected blocks containing one or more memory cells and three multiplicative units (the input, output and forget gates), which allow writing, reading or resetting of the information in the memory cell.

Bi-directional RNNs [16] address in an elegant way the need for delayed decisions in some sequential tasks such as speech processing. Data sequences are presented forwards and backwards to two separate recurrent networks, which are connected to the same output layer. Therefore, for every point in a given sequence, the network has complete sequential information about the points before and after it. An implementation of bi-directional LSTM (BLSTM) can be found in [12].

2.3 CTC

Connectionist temporal classification (CTC) [9,10] is an objective function to label unsegmented sequential data with RNNs. The basic idea behind CTC is to interpret the network outputs as a probability distribution over all possible label

sequences, conditioned on the input data sequence. Given this distribution, an objective function can be derived that directly maximises the probability of the correct labelling. Since the objective function is differentiable, the network can be trained with standard back-propagation through time.

3 Experiments

3.1 Material

The experiments were carried out on a set of dialogues in German from the Verbmobil database [17]. The dialogues deal with the scheduling of date appointments for meetings. The database consist in spontaneous and unconstrained speech, with long silences and noises. The results provided along with the database for a baseline automatic speech recogniser give an approximate idea of the difficulty of processing this database: the HMM-based speech recogniser achieves a word accuracy of 65 % in an automatic, full transcription, of the test set [17].

The material used in the experiments corresponds to version 2.3 (March 2004) of the Verbmobil database (except for CD-ROM number 53.1 of the training set which was not available) [17]. It includes separate training, validation and test sets. The database is speaker independent. Speakers were distributed equally across sexes in all sets and every speaker appears in only one of the sets. The training set includes 748 speakers and 23975 dialogue turns for a total of 45.6 hours of speech. The validation set includes 48 speakers and 1222 dialogue turns for a total of 2.9 hours of speech. The test set includes 46 speakers and 1223 dialogue turns for a total of 2.5 hours of speech.

Due to the nature of the dialogues included in the database, we decided to use dates and places as keywords for the detection task. This ensures a relatively good coverage of keywords in the training, validation and test data sets. The twelve keywords chosen were:

> april, august, donnerstag, februar, frankfurt, freitag, hannover, januar, juli, juni, mittwoch, montag

Note that the database includes various pronunciation variants of some of these keywords (e.g. “montag” can end either with a /g/ or with a /k/). In addition, several keywords appear as sub-words, e.g. in plural form such as “montags” or as part of another word such as “ostermontag” (Easter Monday).

The orthographic transcription provided along with the database was examined for the presence in every utterance of one or more of the twelve keywords selected. Once a keyword was found, the begin and end times for the keyword were saved for evaluating the performance of the keyword spotting system at a later stage. These times were provided along with the database and correspond to the segmentation given by an automatic speech recognition system.

This procedure gave a total of 10469 keywords on the training set with an average of 1.7 % keywords per non-empty utterance (73.6 % of the utterances did not have any keyword); 663 keywords on the validation set with an average

of 1.7 % keywords per non-empty utterance (68.7 % of the utterances did not have any keyword); and 620 keywords on the test set with an average of 1.8 % keywords per non-empty utterance (71.1 % of the utterances did not have any keyword).

The list of keywords (which may be empty) in the order in which they appear in every utterance, forms the target sequence for the network's training algorithm.

Finally, the speech data was characterised as a sequence of vectors of thirty nine coefficients, consisting of twelve Mel-frequency cepstral coefficients (MFCC) plus energy and first and second order derivatives of these magnitudes. The coefficients were computed every 10 ms over 25 ms-long windows. First, a Hamming window was applied; secondly, a mel-frequency filter bank of 26 channels was computed; and finally, MFCC coefficients were calculated with a 0.97 preemphasis coefficient. The input data was normalised to have zero mean and standard deviation one on the training set.

3.2 Difficulty with HMMs

We briefly illustrate the difficulty of performing keyword spotting with a generative HMM with as few constraints as those required by our system. In particular, no post-processing is applied. For these experiments every keyword was modelled as a left-to-right HMM with sixteen states and observation probabilities given by a mixture of Gaussians with diagonal covariance matrices. For the filler model we used an HMM of the same type with three states. The grammar allows one or more repetitions of any keyword or filler per utterance.

We experimented with: a) doubling the number of mixtures from two until a maximum of 128 and re-estimating the parameters twice after every step; b) the parameters of the HMMs with eight Gaussians were re-estimated for more than five hundred iterations, with an evaluation of the results carried out every ten iterations (henceforth, one step); and c) the first experiment was repeated with HMMs initialized using the reference segmentation and increasing the number of mixtures up to 256. The performance was evaluated on the validation set after every step with insertion penalties from zero to minus one hundred in steps of ten. For experiment b), the performance did not improve (it slightly decreased) after 350 iterations.

The results on the test set for all experiments, optimised on the validation set, showed negative accuracy due to the many false positives generated by the system: of the order of three thousand in all cases. Hence, the importance of a post-processing stage for HMMs that estimates the confidence of the predictions and a threshold level for balancing true and false positives.

3.3 Setup

The CTC-BLSTM network was trained with the input sequences and keyword targets from section 3.1. The network has, therefore, 39 input units and 13 units in the output soft-max layer. Both the forward and backward hidden recurrent

layers consist of 128 LSTM memory blocks with one memory cell per block, peephole connections and forget gates. The input and output cell activation functions are a hyperbolic tangent. The gates use a logistic sigmoid function in the range [0,1]. The input layer is fully connected to the hidden layer and the hidden layer is fully connected to itself and to the output layer. The total number of weights in the network is 176,141.

Training was carried out with a learning rate of 10^{-4} and a momentum coefficient of 0.9. Weights were initialised with a Gaussian random distribution with mean zero and standard deviation 0.1. In addition, Gaussian noise with a standard deviation of 0.5 was added to the inputs during training to improve generalization. The network was tested every five epochs on the validation set and training finished once the error on the validation set stopped decreasing.

3.4 Results

The typical sequence of outputs for a trained network can be seen in figure 1. It consists in a sequence of spikes associated with detected keywords and separated by long periods during which the non-keyword output unit is the most active. The CTC algorithm allows the network to keep any output active for as long as necessary or as little as one time step only. To evaluate the results, at every time step the output with the highest activation is chosen. If an output is the most active for a period of time, we choose the highest activation during that period as the probability of detecting the keyword, and the time at which the highest activation occurs as the location of the detected keyword.

Spikes whose identity match a keyword in the speech signal and that appear within the boundaries of the keyword in the speech signal are counted as true positives, unless more than one spike is output in that period, in which case only one of them counts as a true positive and the rest count as false positives. Those spikes that appear out of the boundaries of the keyword in the speech signal are considered false positives. The accuracy of the system is given by the number of true positives minus the number of false positives divided by the number of keywords in the data set.

Four networks were trained on the same task with different initial random weights. Average accuracy over four runs was 84.5 % with a standard error of 1.2 %. The average probability of true positives was 0.98 with a standard error of 0.004. The average probability of false positives was 0.80 with an standard error of 0.01. As expected, given the discriminative nature of the algorithm, setting a posteriori a threshold level in between this probability levels did not increase performance significantly (84.8 % accuracy, with an standard error of 1.2 %). Table 1 shows the number of true and false positives by keyword for the network with the best performance. As shown in the table, the system discriminates almost perfectly between similar keywords such as “juni” and “juli”.

As of now, the CTC training algorithm does not constrain the spikes to appear within the begin and end times of the speech pattern (it is assumed that a reference segmentation is not available). The algorithm trains the network to detect the correct sequence of keywords and in the right order for any utterance

Table 1. Results by keyword for the ANN with the best performance. Accuracy is the number of hits minus false positives divided by the actual number of keywords in the test set. Average accuracy over four runs is 84.5 % with a standard error of 1.2 %.

Keyword	# Hits	# FPs	# Actual	% Accuracy
april	27	2	32	78.12
august	29	1	34	82.35
donnerstag	55	6	56	87.50
februar	55	1	60	90.00
frankfurt	18	0	25	72.00
freitag	40	4	45	80.00
hannover	76	5	86	82.56
januar	35	4	38	81.58
juli	53	1	56	92.86
juni	63	2	66	92.42
mittwoch	36	1	39	89.74
montag	79	3	83	91.57
Overall	566	30	620	86.45

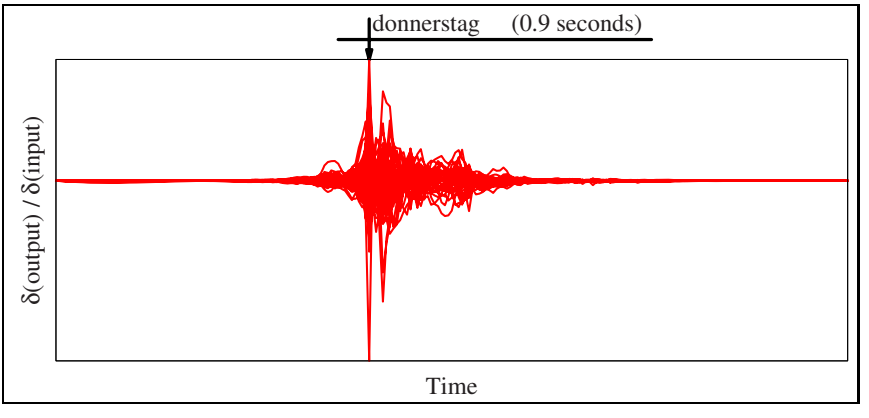

Fig. 2. The figure shows the dependency of the output unit associated with keyword "donnerstag" at the time step when this keyword is detected in figure 1, with respect to the network inputs at various time steps around the detection point (indicated by an arrow at the top of the figure). In addition, the extent (0.9 s) and location of the keyword in the speech signal is shown at the top of the figure. As can be seen, the probability of detecting the keyword depends on the inputs over a long time span, and decreases towards the end of the keyword which is the least discriminative part of it: "tag" (day).

in the training set. Probably this makes training slower, but it seems that it is not necessary to add this constraint to the training algorithm in order to achieve good performance: when the results are evaluated without using the segmentation, the average accuracy over four runs is 86.1 % with a standard

error of 0.7 %. Of those keyword candidates appearing outside the boundaries in the speech signal, only one was more than 0.5 s outside the reference segment. This level of precision should be enough for tasks where high detection accuracy is more important.

Finally, figure 2 shows, for a successful detection, the dependency of the network output at the time when the keyword is detected with respect to the inputs at some previous and following time steps. As can be seen, the probability of detecting the keyword depends on the inputs over a long time span (of about one second, or a hundred network time steps) and, for the example shown, the dependency decreases towards the end of the keyword, which is the least discriminative part of it.

4 Discussion and Future Work

The main difference between our approach to discriminative keyword spotting and other discriminative systems is the capability of computing non-local posteriors for the keywords, i.e. the system uses information over long time spans to compute the probability of a keyword appearing in the speech signal. In addition to this, the a posteriori estimation of a threshold level to balance true and false positives is not required.

The algorithm makes very few assumptions about the domain, which facilitates the development of systems for detection tasks. Nonetheless, if very high precision in the location of spots is required, the system might benefit from adding to the algorithm the constraint that keywords must be detected within the boundaries established by the reference segmentation. However, not having this constraint greatly simplifies the preparation of training data.

To make the system scale, one solution consists in implementing smaller networks. For example, one for each keyword, or one for each set of similar keywords among which the system must discriminate. New networks can be added to the system at any time. Under the assumptions that, for different networks, the training data has similar characteristics and that the non-keyword output unit is associated to a model of similar characteristics, which is reasonable, the estimates of the a posteriori probabilities given by different networks can be compared and used to determine the presence or absence of keywords in the input data stream.

Another option to improve scalability consists in detecting sub-word units in the signal and use these outputs as inputs for one of the existing systems that builds keywords from sub-word units. The detection of sub-word units with the system proposed in this paper will benefit from the same features and ease of use described in this paper (see [9] for results on phoneme recognition with CTC-BLSTM). Besides this, it is possible to feed the sub-word level predictions to another CTC-BLSTM network with keyword-level outputs [10].

The choice of sub-word or word models depends on the task. In general, sub-word units are more practical for audio indexing tasks because the system must respond to searches for unknown keywords. However, in tasks such as detection

of command words and spoken password verification the vocabulary is unlikely to change. Also, the benefits of word-level discrimination (e.g. decreased confusion among similar keywords) are most desirable in these cases, where security is paramount.

5 Conclusion

We have presented a word-level discriminative keyword spotting system that is fast, accurate and easy to use. The probability of detecting a keyword is computed using information from long time spans. These probabilities can be read directly from the outputs of a recurrent neural network. An a posteriori estimate of an acceptance threshold level is not required. Finally, the system makes few assumptions about the domain: only the input speech signal and a list of keywords (which may be empty) in the order in which they occur in the signal are necessary to train the system. The algorithm is general and can be used for any task requiring the detection of patterns in sequence data. In a keyword spotting task in a large database of unconstrained speech the system achieved a keyword spotting accuracy of 84.5 %.

Acknowledgments

This research was funded by SNF grant 200021-111968/1.

References

1. Wilpon, J.G., Rabiner, L.R., Lee, C., Goldman, E.R.: Automatic recognition of keywords in unconstrained speech using hidden Markov models. IEEE Transactions on Acoustics, Speech and Audio Processing 38(11), 1870–1878 (1990)
2. Benítez, M.C., Rubio, A., García, P., de la Torre, A.: Different confidence measures for word verification in speech recognition. Speech Communication 32, 79–94 (2000)
3. Silaghi, M.C., Bourlard, H.: Iterative posterior-based keyword spotting without filler models. In: Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop, Keystone - Colorado (U.S.A.) (1999)
4. Silaghi, M.: Spotting subsequences matching an HMM using the average observation probability criteria with application to keyword spotting. In: Proceedings of the 20th National Conference on Artificial Intelligence and the 17th Conference on Innovative Applications of Artificial Intelligence, pp. 1118–1123. Pittsburgh - Pennsylvania (U.S.A.) (2005)
5. Ketabdar, H., Vepa, J., Bengio, S., Bourlard, H.: Posterior based keyword spotting with a priori thresholds. In: Proceedings of the 9th International Conference on Spoken Language Processing, Pittsburgh - Pennsylvania (U.S.A.) (2006)
6. Robinson, A.J.: An application of recurrent nets to phone probability estimation. IEEE Transactions on Neural Networks 5(2), 298–305 (1994)
7. Bourlard, H.A., Morgan, N.: Connectionist speech recognition: a hybrid approach. Kluwer Academic Publishers, Dordrecht (1994)

8. Keshet, J., Grangier, D., Bengio, S.: Discriminative keyword spotting. In: Workshop on Non-Linear Speech Processing NOLISP, Paris (France) (2007)
9. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural nets. In: Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh - Pennsylvania (U.S.A.) (2006)
10. Fernández, S., Graves, A., Schmidhuber, J.: Sequence labelling in structured domains with hierarchical recurrent neural networks. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad (India) (2007)
11. Silaghi, M.C., Vargiya, R.: A new evaluation criteria for keyword spotting techniques and a new algorithm. In: Proceedings of InterSpeech (2005)
12. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks 18(5–6), 602–610 (2005)
13. Graves, A., Fernández, S., Schmidhuber, J.: Bidirectional LSTM networks for improved phoneme classification and recognition. In: Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S. (eds.) ICANN 2005. LNCS, vol. 3697, pp. 799–804. Springer, Heidelberg (2005)
14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation 9(8), 1735–1780 (1997)
15. Gers, F., Schraudolph, N., Schmidhuber, J.: Learning precise timing with LSTM recurrent networks. Journal of Machine Learning Research 3, 115–143 (2002)
16. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing 45, 2673–2681 (1997)
17. Verbmobil: Database version 2.3 (2004), `http://www.phonetik.uni-muenchen.de/Forschung/Verbmobil/Verbmobil.html`