Recurrent Neural Network and LSTM Models for Lexical Utterance Classification

Suman Ravuri^{1,3} *Andreas Stolcke*^{2,1}

^{1}International Computer Science Institute, ^{3} University of California, Berkeley, CA, USA
^{2}Microsoft Research, Mountain View, CA, USA

ravuri@icsi.berkeley.edu, anstolck@microsoft.com

Abstract

Utterance classification is a critical pre-processing step for many speech understanding and dialog systems. In multi-user settings, one needs to first identify if an utterance is even directed at the system, followed by another level of classification to determine the intent of the user's input. In this work, we propose RNN and LSTM models for both these tasks. We show how both models outperform baselines based on ngram-based language models (LMs), feedforward neural network LMs, and boosting classifiers. To deal with the high rate of singleton and out-of-vocabulary words in the data, we also investigate a word input encoding based on character ngrams, and show how this representation beats the standard one-hot vector word encoding. Overall, these proposed approaches achieve over 30% relative reduction in equal error rate compared to boosting classifier baseline on an ATIS utterance intent classification task, and over 3.9% absolute reduction in equal error rate compared to a the maximum entropy LM baseline of 27.0% on an addressee detection task. We find that RNNs work best when utterances are short, while LSTMs are best when utterances are longer.

Index Terms: utterance classification, recurrent neural net language model, distributed word representations, LSTM.

1. Introduction

Utterance classification is an important pre-processing step for many dialog systems that interpret speech input. For example, a user asking Siri or Cortana to "tell me about the weather" should have her utterance classified as *weather-query* so that the query can be routed to the correct natural understanding subsystem. In a more challenging scenario, in which multiple speakers may be issuing commands to a machine or talking to one another, the system must first correctly ignore human-human and identify human-computer interactions. In past work we have studied addressee detection leveraging several speech-based knowledge sources, such as what was said (lexical content) [1] and how it was said (speaking style) [2]. Other researchers have also incorporated multi-modal cues to help solve this task [3, 4, 5, 6].

In the present paper we continue our exploration of neural network (NN) models for lexical addressee detection [1], in which we showed how to modify a standard Neural Network Language Model (NNLM) [7] to perform utterance-level classification. We now explore potentially more powerful NN-based models, while also expanding the scope of our investigation to a different classification task, i.e., intent classification. There is a vast literature on domain and intent classification for purposes of speech understanding; for prior work see [8] and references therein.

All previous ngram-based classification approaches (standard LMs, NNLMs, boosting) suffer from two fundamental and competing problems: the limited temporal scope of ngrams, and their sparseness, requiring large amounts of training data for good generalization. The longer ngrams one chooses to model, the more the sparseness issue is exacerbated. To address sparseness of data, one can try to enlist outside training data for ngram LMs [9] or to train word embeddings [1] for NN-based classifiers, but these approaches are ultimately limited by the domain-specific nature of ngram distributions (i.e., models trained from outside data often do not generalize). The limited temporal scope of ngrams matters for utterance classification as results may depend on long-term dependencies ("Tell me about your day" and "how was the Mexican restaurant" are likely directed at other humans while "Tell me about the weather" and "Tell me about Mexican restaurants" are likely directed at the system). Since recurrent neural models hold the promise of encoding long-term dependencies through its hidden state, we propose models based on Recurrent Neural Networks and Long Short-Term Memory [10] Units. As our results show, the recurrent neural architectures investigated here achieve better results than the simple feedforward NNs used previously, even without using outside data to train word embeddings as proposed in [1]. In this work, we focus on tasks with binary decisions (which can be treated as detection tasks), although the proposed systems can easily be extended to multi-class situations.

2. Proposed Systems

2.1. RNN-based utterance classifier

Recent work in Recurrent Neural Network Language modeling [11] suggests that temporal modeling of an entire sentence through a series of hidden units can outperform models based on the Markov assumption. Much like the original Neural Network Language Model [7], the RNNLM maps a one-hot $|V|$-dimensional vector w – in which only one dimension of w is 1 and the rest are 0 and $|V|$ is the size of the vocabulary – to a dense n-dimensional word embedding v through the function $P_v w$. The hidden state h_t is a function of the current embedding, the previous hidden state, and a bias – $h_t = \sigma(W_t h_{t-1} + v_t + b_h)$ – and the model attempts to predict the next word w_{t+1} given h_t. A trivial adaptation of the RNNLM for this task would be to train separate models for each class, and at test time take the log likelihood ratio (as is done for LM-based classifiers [1]). This approach, however, requires different vocabulary sets for each model, and unknown word probabilities need to be significantly tuned. Moreover, such an approach would also not share statistical strength between the models, as word representations are trained independently on different classes of data.

Instead, we train a single model on utterance class labels, shown in Figure 1. The RNN model attempts to classify the utterance based on the information stored thus far in h_t. At test time, the probability of an utterance label is calculated as:

$$P(L|\mathbf{w}) \approx P(L_1, \ldots, L_n|\mathbf{w}) = \prod_{i=1}^{n} P(L_i|\mathbf{w})$$
$$\approx \prod_{i=1}^{n} P(L_i|w_i, h_{i-1}) = \prod_{i=1}^{n} P(L_i|h_i)$$

where the last equality is embodied in the final softmax function.

2.2. LSTM-based utterance classifier

A desideratum for a model performing utterance classification is to have a single label per utterance. In preliminary experiments, we did not obtain competitive performance with RNN models predicting a single label at the end of an utterance, likely due to the vanishing gradient problem. As a result, we investigated the use of long short-term memory (LSTM) units for utterance classification.

The LSTM, first described in [10], attempts to circumvent the vanishing gradient problem by separating the memory and output representation, and having each dimension of the current memory unit depending linearly on the memory unit of the previous timestep. A popular modification of the LSTM uses three gates – input, forget, and output – to modulate how much of the current, the previous, and output representation should be included in the current timestep. Mathematically, it is specified by the equations:

$$i_t = \sigma(W_i v_t + U_i h_{t-1} + b_i)$$
$$f_t = \sigma(W_f v_t + U_f h_{t-1} + b_f)$$
$$o_t = \sigma(W_o v_t + U_o h_{t-1} + V_o m_t + b_o)$$
$$m_t = i_t \circ tanh(W_c x_t + U_c h_{t-1} + b_c) + f_t \circ m_{t-1}$$
$$h_t = o_t \circ tanh(m_t)$$

where i_t, f_t, and o_t denote the input, forget, and output gates respectively, m_t, the memory unit, and h_t, the hidden state, and is shown in Figure 2. We found that, unlike for RNNs, a model making a single prediction at utterance end does achieve good performance.

One question is whether the one-hot vector w should input directly to the LSTM, as proposed by [12] for a slot-filling task. We found it better to use a separate linear embedding, and use the embedding v_t as input to the LSTM. Figure 2 shows this model.

2.3. Word hashing

One issue we noticed was that recurrent models are somewhat more sensitive to the occurrence of unknown words than standard feedforward networks. This is best explained by example. Consider an utterance which begins with an unknown word. In a trigram model, only the first two samples are affected by the unknown word, while in recurrent NNs all future hidden states are affected by the unknown word. For corpora with a high percentage of singletons in the training set, this problem is particularly acute, as the standard practice is to map all such words to an unknown token. In fact, one corpus on which we tested our models contained 60% singletons. Moreover, such unknown words may be informative, such as "Kleaners" in "Can you show me the address of Happy Kleaners?". To combat issues with unknown word modeling, we investigate word representations based on sets of character ngrams, as proposed in [13]. A word such as "Kat" is transformed into a set of character ngrams, each of which is associated with a bit in the input encoding. In the case of character trigrams, this hash is the set "#Ka", "Kat", "at#", and the probability of a collision in the hash is less than 0.01%.

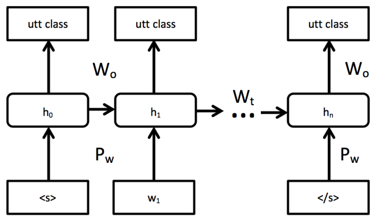

Figure 1: *Proposed RNN classifier model.*

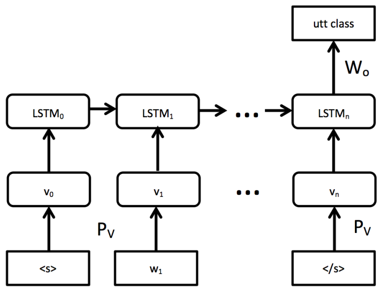

Figure 2: *Proposed LSTM classifier model.*

3. Method

For our experiments, we chose two corpora – ATIS and the Conversational Browser – as the corpora exhibit large differences, and we wanted to determine whether our proposed models can handle vastly different conditions. The ATIS corpus generally has higher-accuracy transcripts, fewer singletons, and longer utterances than the Conversational Browser corpus.

3.1. ATIS Intent Classification Task

We follow the ATIS corpus setup used in [14, 15] in this paper. The training set comprises 4,978 utterances taken from the Class A (context independent) portions of ATIS-2 and ATIS-3, and 893 test utterances from the ATIS-3 Nov93 and Dec94 datasets. The corpus has 17 different intents, which we mapped to a binary "flight" versus "other" classification task (70% of the utterances are classified as "flight", though our metric is insensitive to prior distribution, as explained below). Training was based on reference transcripts, but testing used ASR output as described in [8], with a word error rate of around 14%. There are two versions of the input: one uses only the original transcript words, the other—known as "autotagged"—replaces entities by phrase labels such as CITY and AIRLINE, obtained from a tagger [12].

3.2. Conversational Browser addressee classification

The "Conversational Browser" (CB) is a corpus in which two users interact with a dialog system using spoken input. Subjects were brought into a room and seated about 5 feet away from a large TV screen and roughly 3 feet away from each other, told basic capabilities and a small set of short commands, but are

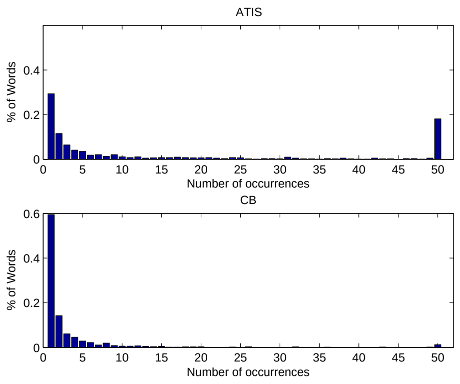

Figure 3: *Distribution of word occurrences for ATIS and Conversational Browser datasets.*

otherwise told to use unrestricted natural language. More information about the dialog system itself and its spoken language understanding approach can be found in [16].

The resulting corpus comprises 6.3 hours of recordings over 38 sessions with 2 speakers each from a set of 36 unique speakers. Session durations ranged from 5 to 40 minutes, as determined by users. Speech was captured by a Kinect microphone-array; endpointing and recognition used an off-the-shelf recognizer. Note that training and testing is based on recognizer output, with a word error rate of about 20%. More information on the addressee task for this corpus can be found in [1].

3.3. NN Training

As noted by other authors, parameter estimation of RNNs is substantially more difficult to train than for feedforward networks. Well-trained systems typically use a combination of momentum, truncated back-propagation through time (BPTT), regularization, and gradient clipping. Since utterance lengths for the corpora investigated were typically under 20, we found no improvement employing gradient clipping or truncated BPTT. Moreover, regularization had either minimal or deleterious effect. While simple momentum did help, more advanced modifications such as Nesterov momentum yielded no improvement.

Despite the relative ease of the problem, we did find that final results were sensitive to initial parameters, which, for RNN and non-gate LSTM weights were drawn from a $\mathcal{N}(0.0, .04)$ distribution, while LSTM gate weights were drawn from the same distribution, except that the gate biases were a large positive value (around 5) to ensure those values started at approximately 1.0. The variance of performance is investigated in Section 5. One heuristic that worked well in practice: prior to training, we calculated the cross-entropy on a held-out set across ten random seeds and picked the one which produced the lowest cross-entropy.

The initial learning rate for each of the systems is 0.01, with a momentum of 3E-4 for recurrent neural networks, and 3E-5 for LSTM models. The learning rate is halved once the cross-entropy on a held-out set decreases less than 0.01 per example, continues at the same rate until the same stopping point is achieved, and then halved each epoch until cross-entropy does not decrease. As stated earlier, the best initial cross-entropy across 10 different initializations is used.

3.4. Experimental Setup

Both the RNN and LSTM models use a 200-dimensional word embeddings for one-hot and word hashing on both corpora, as those parameters experimentally produced the best results. In addition, the LSTM included a layer of 15-dimensional hidden and memory units. Since the LSTM per hidden unit uses 11 times the number of parameters as a standard feedforward network, the LSTM model has roughly 150% more parameters than the RNN, but this structure produced the best results. For word hashes, we use a concurrent trigram and bigram representations. For example, the set describing “cat” is “#c”, “ca”, “at”, “t#”, “#ca”, “cat”, and “at#’.

For model combination and evaluation, we use linear logistic regression (LLR) to calibrate all model scores or to combine multiple scores where applicable [17]. To estimate LLR parameters, we jack-knife over all sessions in the test data, training on all but one session in turn, and cycling through all sessions.[1] Scores are then pooled over the entire test set and evaluated using equal error rate (EER). The EER is obtained by choosing a decision threshold that equates false alarm and miss error probabilities. EER is thus independent of class priors.

4. Results and Discussion

Results for both proposed systems are shown in Tables 1 and 2 and highlight the importance of using multiple corpora and tasks for assessing model performance. For the ATIS corpus, both the RNN and LSTM models beat a maximum entropy language model, and the LSTM beats a boosting model. In particular, the LSTM model is 45.2% better relative to the boosted word-ngram system on the word setting, and 40.1% better when the certain classes, such as digits and named entities, are automatically tagged. RNN models do not beat the boosted ngram baseline system when using only words, but do in the auto-tagged setting. Finally, in no instance does word hashing beat the standard word embedding baseline.

These results are reversed for the CB data, as shown in Table 2. Here, the LSTM model is modestly worse than the baseline. The RNN model, however, is 0.58% better absolute than the ngram baseline for standard words. Moreover, hashing provides a substantial gain for both the LSTM and RNN model. The RNN word hash is 1.73% better absolute over the ngram baseline, and in combination is 3.92% better.[2] All models outperform previous feedforward neural network language models.

What accounts for the LSTM to perform better than RNN and baseline models on ATIS while worse on CB, and word hashing to perform better than word embedding on CB while worse on ATIS? The answer to the first question is likely that the average utterance length of ATIS is much higher than on CB. As shown in Figure 4, most utterance lengths for the CB data are under 5 words, while for ATIS the median is above 10 words. Differences in datasets also account for the disparate performance in word hashing. As shown in Figure 3, the number of singletons – which get mapped to unknown words in the training set – for ATIS is roughly around 30%, while for CB, the number is 60%.

Tables 3 and 4 show the mean performance and standard deviation across 10 different random seeds. On average, the results are better than the baseline, but the mean result is generally a bit worse than picking the best initial cross-entropy. The notable counterexample are the word-based RNN systems on the ATIS dataset. Moreover, it looks as if the LSTM generally has

[1]For ATIS we did not have session information and paritioned the data sequentially into nine equal-sized portions.

[2]Combination with baseline systems did not, on average improve performance

higher variance than the RNN counterparts. Since the datasets are relatively small, this effect could be diminished when using more data.

Finally, one can modestly improve performance by picking the best neural network based on held-out set performance, at the cost of ten times the training time. Using those results, can, in some cases, achieve the oracle results for both sets.

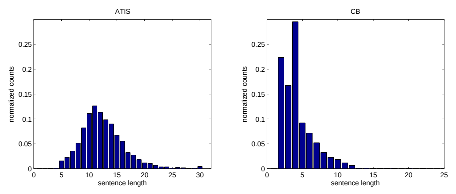

Figure 4: *Histograms of sentence length for ATIS (left) and Conversational Browser (right) data.*

Table 1: *ATIS intent classification results. The column labeled autotagged refers to the condition in which certain named entities are marked via lookup table.*

System	EER (%)	Autotag EER (%)
word-ngram (MaxEnt LM)	9.37	6.05
word-ngram-boost	4.47	3.24
RNN-word	5.26	2.45
RNN-hash	5.33	2.81
LSTM-word	**2.45**	**1.94**
LSTM-hash	2.88	2.81

Table 2: *CB addressee classification results. The column labeled "Combo" refers to system performance when scores are combined with baseline word-ngram system.*

System	EER (%)	Combo EER (%)
word-ngram (MaxEnt LM)	27.00	–
NNLM-addressee	30.01	–
NNLM-ngram	29.46	–
RNN-word	26.42	24.72
RNN-hash	**24.27**	**23.08**
LSTM-word	27.17	25.69
LSTM-hash	26.65	25.28

Table 3: *Average, Best Held Out, and Oracle Errors on ATIS intent classification. Best held out refers to the hypothetical performance when the model with the lowest cross-entropy on a held-out set is chosen (among 10 random seeds) after training.*

System	EER (%)	Autotag EER (%)
Average Error		
RNN-word	$4.86 \pm .919$	$3.50 \pm .775$
RNN-hash	$4.32 \pm .917$	$2.64 \pm .324$
LSTM-word	$3.38 \pm .986$	2.22 ± 1.01
LSTM-hash	4.05 ± 1.13	2.64 ± 1.02
Best Held-Out X-ent (Oracle Error)		
RNN-word	3.95 (3.95)	2.45 (2.45)
RNN-hash	3.59 (3.24)	2.45 (2.09)
LSTM-word	2.81 (2.45)	2.02 (1.30)
LSTM-hash	3.24 (2.88)	2.02 (1.22)

5. Conclusions

In this work, we investigated recurrent models for utterance classification. LSTM and RNN models in general outperformed strong baselines based on ngram boosting or maximum entropy LMs. LSTM models seemed to work quite well when the utterances were long on average, while the RNN seemed to work better for shorter utterances. Moreover, word hashing provided a good improvement for the corpus with many singleton words.

For future work, we would like to better quantify at what utterance length is the LSTM a better model. Moreover, we would like to analyze in what scenarios word hashing makes sense, and how to best combine word hashing and standard word embedding approaches. Finally, we are in the process of using these models on much larger corpora to determine if similar improvements are available.

Table 4: *Average, Best Held Out, and Oracle Errors on CB addressee detection. Best held out refers to the hypothetical performance when the model with the lowest cross-entropy on a held-out set is chosen (among 10 random seeds) after training.*

System	EER (%)	Combo EER (%)
Average Error		
RNNLM-word	$26.20 \pm .304$	$25.13 \pm .319$
RNNLM-hash	$24.66 \pm .482$	$23.81 \pm .395$
LSTM-word	$27.67 \pm .822$	$25.86 \pm .686$
LSTM-hash	$26.69 \pm .802$	$25.64 \pm .574$
Best Held-Out X-ent (Oracle Error)		
RNNLM-word	25.75 (25.65)	25.20 (24.09)
RNNLM-hash	23.88 (23.88)	23.43 (23.08)
LSTM-word	27.47 (26.65)	27.25 (24.82)
LSTM-hash	25.65 (25.65)	25.20 (24.88)

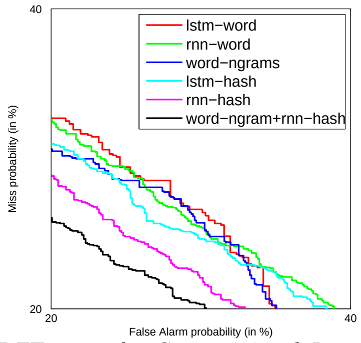

Figure 5: *DET curve for Conversational Browser addressee classification.*

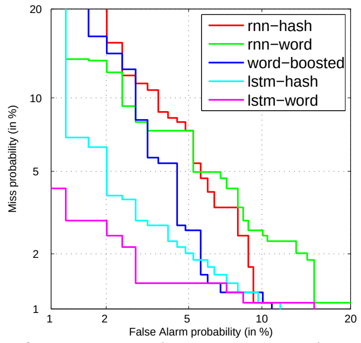

Figure 6: *DET curve for ATIS intent classification.*

6. Acknowledgements

We gratefully acknowledge Dilek Hakkani-Tur and Kaisheng Yao for their help with setting up the ATIS corpus and discussion on LSTM use for this task.

7. References

[1] S. Ravuri and A. Stolcke, "Neural network models for lexical addressee detection", *in Proc. Interspeech*. ISCA - International Speech Communication Association, September 2014.

[2] E. Shriberg, A. Stolcke, and S. Ravuri, "Addressee detection for dialog systems using temporal and spectral dimensions of speaking style", *in Proc. Interspeech*, Lyon, Aug. 2013.

[3] M. Katzenmaier, R. Stiefelhagen, and T. Schultz, "Identifying the addressee in human-human-robot interactions based on head pose and speech", *in Proceedings of the 6th international conference on Multimodal interfaces*, pp. 144–151, State College, PA, USA, 2004. ACM.

[4] R. op den Akker and D. Traum, "A comparison of addressee detection methods for multiparty conversations", *in Proceedings of Diaholmia*, pp. 99–106, 2009.

[5] D. Bohus and E. Horvitz, "Multiparty turn taking in situated dialog: Study, lessons, and directions", *in Proceedings ACL SIGDIAL*, pp. 98–109, Portland, OR, June 2011.

[6] N. Baba, H.-H. Huang, and Y. I. Nakano, "Addressee identification for human-human-agent multiparty conversations in different proxemics", *in Proceedings 4th Workshop on Eye Gaze in Intelligent Human Machine Interaction*. ACM, Oct. 2012, Article no. 6.

[7] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model", Technical Report 1178, Department of Computer Science and Operations Research, Centre de Recherche Mathématiques, University of Montreal, Montreal, 2000.

[8] G. Tur, D. Hakkani-Tür, L. Heck, and S. Parthasarathy, "Sentence simplification for spoken language understanding", *in IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE SPS, May 2011.

[9] H. Lee, A. Stolcke, and E. Shriberg, "Using out-of-domain data for lexical addressee detection in human-human-computer dialog", *in Proceedings North American ACL/Human Language Technology Conference*, pp. 221–229, Atlanta, GA, June 2013.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[11] T. Mikolov, M. Karafiát, L. Burget, J. H. Černocký, and S. Khudanpur, "Recurrent neural network based language model", *in Proc. Interspeech*, pp. 1045–1048, Makuhari, Japan, Sep. 2010.

[12] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, "Spoken language understanding using long short-term memory neural networks", *in IEEE SLT*, 2014.

[13] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for web search using clickthrough data", *in Proceedings of the 22Nd ACM International Conference on Conference on Information &#38; Knowledge Management*, CIKM '13, pp. 2333–2338, New York, NY, USA, 2013. ACM.

[14] Y. He and S. Young, "A data-driven spoken language understanding system.", *in Proceedings IEEE Workshop Automatic Speech Recognition and Understanding*, 2003.

[15] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding.", *in INTERSPEECH*, pp. 1605–1608. ISCA, 2007.

[16] L. Heck, D. Hakkani-Tür, M. Chinthakunta, G. Tur, R. Iyer, P. Parthasarathy, L. Stifelman, A. Fidler, and E. Shriberg, "Multimodal conversational search and browse", *in Proceedings IEEE Workshop on Speech, Language and Audio in Multimedia*, Marseille, Aug. 2013.

[17] S. Pigeon, P. Druyts, and P. Verlinde, "Applying logistic regression to the fusion of the NIST'99 1-speaker submissions", *Digital Signal Processing*, vol. 10, pp. 237–248, Jan. 2000.