A CONVERSATIONAL NEURAL LANGUAGE MODEL FOR SPEECH RECOGNITION IN DIGITAL ASSISTANTS

Eunjoon Cho, Shankar Kumar

Google Inc., USA
{ejcho,shankarkumar}@google.com

ABSTRACT

Speech recognition in digital assistants such as *Google Assistant* can potentially benefit from the use of conversational context consisting of user queries and responses from the agent. We explore the use of recurrent, Long Short-Term Memory (LSTM), neural language models (LMs) to model the conversations in a digital assistant. Our proposed methods effectively capture the context of previous utterances in a conversation without modifying the underlying LSTM architecture. We demonstrate a 4% relative improvement in recognition performance on *Google Assistant* queries when using the LSTM LMs to rescore recognition lattices.

Index Terms— speech recognition, recurrent language model, digital assistants, conversation

1. INTRODUCTION

Automatic speech recognition of conversations has a wide range of applications such as telephony, call-center and meeting transcriptions as well as digital assistants. Unlike other spoken corpora such as news broadcasts, voice-search queries or short message services, conversations are typically centered around common themes or topics, which persist through the duration of the conversation. Thus, a conversation-aware language model can potentially make use of such long range context to resolve ambiguities in speech recognition.

Prior work has explored the modeling of long term contextual information using language models (LMs) in the context of automatic speech recognition [1, 2], machine translation [3] and spoken language understanding [4, 5, 6].

Similar to prior work, we use recurrent, Long Short Term Memory (LSTM), neural language models to capture the long term context within a spoken conversation. A common approach in these contextual systems is to model the speaker turn or context as an additional input to the network layers [7, 5, 8] or the LSTM cells [9, 10].

As an alternative to modifying the network architecture to capture dependencies specific to conversations, we explore multiple techniques to modify the input data for training a standard LSTM language model on conversations. This has the advantage of using an identical and reliable architecture for generic ASR language models while still being able to provide previous utterances as context for training the LM.

The human-machine dialog in a digital assistant has a word distribution which is quite different from that seen in normal human-human interactions. The user of the assistant usually provides short command-like queries whereas the assistant is likely to respond with a more verbose response that may provide detailed information back to the user. We experiment with strategies that take these assistant-specific characteristics into account while training the LSTM LM.

In Section 2, we describe the architecture and model training for the LSTM LM. In Section 3, we present a variety of approaches to train the LSTM LM by modifying the input text consisting of the previous queries and assistant responses. We describe our lattice rescoring setup in Section 4 and present results of our experiments in Section 5 followed by a discussion in Section 6.

2. LSTM LANGUAGE MODEL

A language model (LM) [11] assigns a probability to a sequence of words, w_1^T:

$$P(w_1^T) = \prod_{i=1}^{T} P(w_i|w_1, w_2, \ldots, w_{i-1}). \quad (1)$$

While an N-gram LM uses the previous $N-1$ words as context, a recurrent neural network language model (RNNLM) [12] can make use of the entire history of words thus far, while assigning a probability to the next word. This makes RNN LMs, and specifically the LSTM variants [13] effective in tasks where long-term history is important, as is the case for speech recognition of videos [14].

We use the LSTM LM architecture described in [15, 14]

(Figure 1). Each LSTM cell is specified as:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{ri}r_{t-1} + D_{wi}c_{t-1} + b_i) \\
f_t &= 1.0 - i_t \\
o_t &= \sigma(W_{xo}x_t + W_{ro}r_{t-1} + D_{wo}c_{t-1} + b_o) \\
c_t &= c_{t-1} \odot f_t + i_t \odot \tanh(W_{xc}x_t + W_{rc}r_{t-1} + b_c) \\
m_t &= \tanh(c_t) \odot o_t \\
r_t &= W_{rm}m_t
\end{aligned} \tag{2}
$$

where $x_t = Ew_t$ is a continuous representation of the word w_t obtained using the embedding matrix E. σ is the logistic sigmoid function. i_t, f_t and o_t are the *input*, *forget* and the *output* gates. The *forget* and *input* gates are coupled to limit the range of the internal state of the cell. The tuple (c_t, r_t) represents the internal cell states or the *LSTM state*. W, D and b denote full weight matrices, diagonal weight matrices and biases, respectively. $\odot$ represents element-wise multiplication. The architecture uses *peep-hole* connections from the internal cells (c_t) to the gates to learn the precise timings of the outputs [16, 17]. We use a recurrent projection layer (r_t) [18] that reduces the dimension of a big recurrent state (m_t), thus allowing the matrices to remain relatively small while retaining a large memory capacity.

A speech recognition system for voice-search or a digital assistant typically uses a vocabulary of a few million words [19]. With such a large vocabulary, the softmax layer is computationally expensive for both training and testing. To reduce the computation, our RNNLM models the top 100K words from the training corpus. Any word not in this vocabulary is mapped to an unknown token, which is considered a word in the RNNLM vocabulary. To further reduce the computation at training time, we employ a *sampled softmax* [15] over 3% of the vocabulary, where the negative samples are drawn from a log-uniform distribution after ensuring that the words are sorted in decreasing order of their unigram frequencies in the training data.

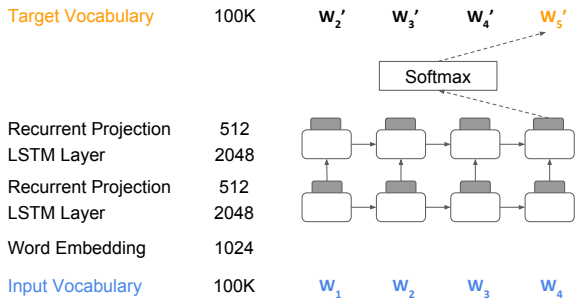

Fig. 1. Architecture of the LSTM LM: State of the network while predicting the final target word w_5, is shown.

3. MODELING CONVERSATION

Our goal is to incorporate conversation context in the standard LSTM LM. For training such a model, we create sentences by concatenating previous queries either with or without the responses generated from the agent. We experiment with the following strategies for generating the input data.

3.1. Queries only

In this scheme, each user query e.g. *What is the weather today?* is treated as a separate sentence, i.e. there is no conversational context. This serves as the baseline for our experiments.

3.2. Contextual Queries

We prepend each query with the previous two queries, collected within an interval of 5 minutes. In this setup, we require that each training sentence consist of a total of three queries. We separate each pair of queries using a turn boundary marker, which is a token in the LSTM LM vocabulary.

3.3. Contextual Queries and Assistant responses

We augment the training data in 3.2 with assistant responses e.g. *It is cloudy and 70 degrees Fahrenheit*. Each training sentence consists of a sequence of three queries, and their corresponding assistant responses, separated using turn boundary markers. We experiment with four variants. In the first variant, we omit the turn boundary markers. For the second, we do not require a maximum of three queries per sentence i.e. if we extract fewer than 3 queries in the 5 minute window, we also keep those queries in our training data. This strategy makes it possible to include additional training data which were discarded because they did not consist of three queries. The next two variants are based on the hypothesis that for modeling a query, the previous query history is more influential compared to the previous assistant response. Our third variant derives the vocabulary from queries alone, while training on both queries and agent responses. In the fourth variant, we vary the order by presenting the assistant responses first followed by the queries.

4. LATTICE RESCORING

A speech recognizer selects the hypothesis with the highest posterior probability given the acoustic waveform O:

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, P(W|O) = \underset{W}{\operatorname{argmax}}\, P(O|W)P(W), \tag{3}$$

where $P(O|W)$ and $P(W)$ denote the probabilities assigned by the acoustic and the language model, respectively. In this work, we integrate the LSTM LM into the recognizer by first generating recognition lattices using an acoustic model and an

N-gram LM. The lattices are then rescored by the LSTM LM using the push-forward algorithm [20, 14], where we keep the single-best LSTM state at each lattice node. This strategy is based on prior work [14] that has shown that keeping a larger number of LSTM hypotheses at each lattice node gives a very small improvement in recognition performance at the cost of much higher computation. On each lattice arc, we interpolate the log probabilities of the LSTM LM with that of the first pass, n-gram LM, using a weight of 0.5:

$$\log P(W) = 0.5 \log P_{\text{LSTM LM}}(W) + 0.5 \log P_{\text{n-gram}}(W) \quad (4)$$

For rescoring the lattice corresponding to a given query, we first present the LSTM LM with the word tokens in the conversational context for the utterance. This context consists of the previous queries and/or assistant responses depending on the scheme used (Section 3). We then rescore the lattice using the LSTM LM. By initializing the LSTM LM with the tokens in the conversation, we expect the internal state of the LSTM LM to capture the long-range context in the conversation.

5. EXPERIMENTS

We next report speech recognition results on an US English, conversational assistant task. The underlying recognizer uses a low frame rate neural acoustic model (AM) as described in [21]. A 5-gram LM with Katz backoff, with a total of 100M n-grams and 4M vocabulary, was trained on $180B$ sentences. The initial word lattice was generated using this AM and LM, and the lattices were rescored with our conversational LSTM LM.

The conversational LSTM LMs were trained on anonymized queries and responses for the *Google Assistant*. Each data set was extracted and formatted differently according to the specific model that was being trained. However, all of them were extracted on the same date range such that the results across models are comparable. Each sequence consists of at most 3 query and response pairs that occur within a 5 minute time frame. The data was approximately 200M sequences in total, with 16.9B words for sequences with responses and 6.3B words for sequences with only previous queries.

For training the LSTM LM, we used hyperparameters from [15] and trained the model until convergence using an AdaGrad optimizer [22] using a learning rate of 0.2 without *dropout* [23]. We unrolled the RNNs for 20 steps, used a batch size of 128, and clipped the gradients of the LSTM weights so that their norm was bounded above by 1.0 [24]. The training used 32 GPU workers and asynchronous gradient updates. Training was performed using TensorFlow [25].

We measure the Word Error Rate (WER) on two test sets. The data was sampled differently and collected from a different region compared to the training data used for training our LSTM LMs. Testset A consists of 4,511 transcribed utterances (16,081 words) sampled from the same *Google Assistant* traffic. This contains utterances that might or might not have previous queries (and responses) within a 5 minute window. Testset B is a subset of Testset A and has 3,774 transcribed utterances (12,672 words). Each of the utterances in Testset B has exactly 2 previous query and response pairs that the model can pass on as context. Although each of the utterances in the test set are transcribed, the preceding queries that are input as context for any given utterance are not transcribed. i.e., we rely on the ASR output for preceding queries in order to simulate what would happen in production.

5.1. Using previous queries as context

Models	Testset A	Testset B
No context	11.9	12.5
Context	11.6	12.2

Table 1. Evaluation using previous queries as context

Table 1 reports WERs for the models that were trained with and without previous queries as context. The model with previous queries was trained on sequences that each had three consecutive queries occurring within a 5 minute window. The model trained with previous queries as context had gains over the model that did not. A big portion of the gains were queries from users engaged in a question answering feature of the *Google Assistant*. Acoustically confusable words such as *two* vs. *too* were commonly corrected with the contextual model. This is mostly likely because of the fact that if the previous queries were numbers, the contextual model would likely hypothesize the current query to be a number as well.

5.2. Using Assistant responses as context

In this section, we explore the benefit of including the responses of the digital assistant as input to the model. Model 1 which includes responses in the context outperformed the baseline model as shown in Table 2. The reason this helps is similar to the reason for gains achieved in Section 5.1. The additional context that the previous agent response provides is helpful in avoiding mis-recognitions on the current user query. For example, wins were achieved when the user said *no* to answer a question asked by the agent. In this case, having reference to the agent's previous response would help recognize the query given the strong correlation between the two. However, the majority of the questions are actually usually said by the users. Words that were often correctly recognized in the baseline model but not in Model 1 were question words (e.g. *what*, *who*, etc.). These are common words that appear mostly as part of a user query and less so in a response by the agent. The fact that Model 1 was trained on both user

Model #	Model Details	Testset A	Testset B
Baseline	Context (queries only)	11.6	12.2
Model 1	Context (queries + responses)	11.5	12.1
Model 2	Context (queries + responses) with vocabulary from queries	11.6	12.3
Model 3	Context (queries + responses) with priority on queries	11.4	12.1
Model 4	Context (queries + responses) without turn makers	11.5	12.2
Model 5	Context (queries + responses) with partial sequences	11.6	12.4

Table 2. Model variations for using assistant responses as context

queries and agent responses resulted in a distribution that was different from our target, which is to predict on queries only.

Without directly modifying the model architecture, we explored two approaches of modifying the input to compensate for this difference. The first approach, Model 2, was to restrict the vocabulary of the model with words that were from the queries only. This was to constrain the model to only output what was previously observed as a query and not words unique to the responses. The second approach, Model 3, was to reorder the sequence of previous queries and responses such that all the assistant responses were presented at the beginning of the sequence despite the chronological ordering. For example, if the chronological order or queries (Q_i) and responses (R_i) were

$$Q_1, R_1, Q_2, R_2, Q_3,$$

we would re-order the training sequence to be

$$R_1, R_2, Q_1, Q_2, Q_3.$$

This was motivated by the fact that a lot of the responses were long utterances. Although LSTM models have the ability to memorize long sequences, given the fact that the sequences including assistant responses had on average 3.2 times more words, (with an average of 53 words per sequence) the reordering was done to put a recency bias on words that were most relevant to the task. Table 2 shows that with Model 3, we were able to achieve a slight gain on top of Model 1.

We also explored two other approaches of modeling the query and responses. Model 4 attempts to model the sequence without the presence of turn markers, which are used to separate the consecutive queries and responses. We observe a slight quality loss without these markers. Model 5 trains the model on sequences that are not constrained to have 3 consecutive query, response pairs. Naturally, some queries to the assistant are not part of a longer interaction, hence modeling this would make sense. However, this didn't necessarily help Testset A which had all range of sequences, whereas it predictably hurt Testset B, which was limited to utterances that had 2 previous query, response pairs. We believe this can be partially explained by the fact that a mixture of sequence lengths in the input breaks the regularity that is otherwise present in a model that is consistently trained with sequences of 3 query, response pairs.

6. DISCUSSION

In this paper, we explored strategies for training a standard LSTM LM on conversational data from a digital assistant. Rather than modifying the structure of the model, we kept the model architecture fixed and experimented with alternative inputs for training the model. We obtained a 4% relative in accuracy on a speech recognition task for *Google Assistant*. This demonstrates that it is possible to obtain improvements in speech recognition for conversational agents without using architectures which are tailored to the problem. Our training strategies investigate the asymmetric nature of turns in a digital assistant, where the user queries are short and the agent responses are generally longer.

7. REFERENCES

[1] Songfang Huang and Steve Renals, "Modeling topic and role information in meetings using the hierarchical dirichlet process," in *International Workshop on Machine Learning for Multimodal Interaction*. Springer, 2008, pp. 214–225.

[2] Holger Schwenk and Jean-Luc Gauvain, "Neural network language models for conversational speech recognition," in *Eighth International Conference on Spoken Language Processing*, 2004.

[3] Alan Ritter, Colin Cherry, and William B Dolan, "Data-driven response generation in social media," in *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2011, pp. 583–593.

[4] Yi Luan, Shinji Watanabe, and Bret Harsham, "Efficient learning for spoken language understanding tasks with word embedding based pre-training.," in *INTERSPEECH*, 2015, pp. 1398–1402.

[5] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan, "A neural network approach to context-sensitive generation of conversational responses," *arXiv preprint arXiv:1506.06714*, 2015.

[6] Orio Vinyals and Quoc V. Le, "A neural conversational model," in *ICML*, 2015.

[7] Yi Luan, Yangfeng Ji, and Mari Ostendorf, "Lstm based conversation models," *arXiv preprint arXiv:1603.09457*, 2016.

[8] Bing Liu and Ian Lane, "Dialog context language modeling with recurrent neural networks," *arXiv preprint arXiv:1701.04056*, 2017.

[9] Tomas Mikolov and Geoffrey Zweig, "Context dependent recurrent neural network language model.," *SLT*, vol. 12, pp. 234–239, 2012.

[10] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young, "Semantically conditioned lstm-based natural language generation for spoken dialogue systems," *arXiv preprint arXiv:1508.01745*, 2015.

[11] F. Jelinek, *Statistical Methods for Speech Recognition*, The MIT Press, Cambridge, MA, USA, 1997.

[12] Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH*, 2010.

[13] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] Shankar Kumar, Michael Nirschl, Daniel Holtmann-Rice, Hank Liao, Ananda Theertha Suresh, and Felix Yu, "Lattice rescoring strategies for long short term memory language models in speech recognition," in *ASRU*, 2017.

[15] Rafal Josefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Wu Yonghui, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410v2*, 2016.

[16] Alex Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[17] Gers A, Schraudolph Nicol N., and Jürgen Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of Machine Learning Research*, vol. 3, pp. 115–143, 2003.

[18] Hasim Sak, Andrew W. Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *INTERSPEECH*, 2014.

[19] Babak Damavandi, Shankar Kumar, Noam Shazeer, and Antoine Bruguier, "NN-grams: Unifying neural network and n-gram language models for speech recognition," in *INTERSPEECH*, 2016.

[20] Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig, "Joint language and translation modeling with recurrent neural networks," in *EMNLP*, 2011.

[21] G. Pundak and T. N. Sainath, "Lower Frame Rate Neural Network Acoustic Models," in *Proc. Interspeech*, 2016.

[22] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[23] Nitish Srivastava, *Improving neural networks with dropout*, Ph.D. thesis, University of Toronto, 2013.

[24] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks.," in *ICML*, 2013, pp. 1310–1318.

[25] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.