End-To-End Generative Dialogue

Colton Gyulay
cgyulay@college.harvard.edu

Michael Farrell
michaelfarrell@college.harvard.edu

Kevin Yang
kyang01@college.harvard.edu

May 12, 2016

1 Introduction

Conversational interfaces offer smooth and natural exchanges between humans and devices. As such, dialogue systems (or chatbots) that enable these interactions are becoming evermore important. These dialogue systems have been employed in a number of applications, from virtual assistants to customer support to entertainment. Traditional approaches to developing dialogue systems are often quite narrow in scope and feature a significant amount of manual engineering. Recently, research into neural language modeling has yielded impressively flexible results in conversational contexts.

In this paper, we attempt to recreate cutting edge end-to-end generative probabilistic models for the purpose of conversation modeling. Neural approaches, especially those based on recurrent structures, lend themselves naturally to modeling text sequences and have been shown to achieve state-of-the-art results on non-goal-driven dialogue tasks [VL15; Ser+15], as well as statistical machine translation [SVL14; Cho+14]. We focus on two promising models — the sequence-to-sequence recurrent encoder-decoder and the hierarchical recurrent encoder-decoder — in order to generate a text response to a conversational context.

In order to develop a generalized conversational agent, we train these models on dialogue extracted from movie subtitles. The dialogue is comprised of many different topics, which enables our models to generate equally broad output. Because of a shortcoming of our specific training objective, the models learn to output "safe," vague responses, like "I don't know" in many different conversational contexts. We explore an alternative scoring criterion, maximum mutual information, which seeks to select context-specific responses over unconditionally probable responses. In total, we are able to achieve comparable results to current research, though many gains are still to be had in improving the quality and specificity of generated text.

2 Problem Description

Given a sequence of m utterances $U_1, \ldots, U_m$ between two interlocutors A and B, also known as a dialogue D, a conversational dialogue system should return the next utterance, U_{m+1}, that is,

the continuation of the conversation at hand [Ser+15]. For example, given the set of preceding utterances $\{U_1, U_2, U_3\}$, the model should generate outputs like U_4:

Input	Sample Output
U_1 : A: Hey there.	U_4 : B: Not bad.
U_2 : B: Hello.	U_4 : B: I'm having a rough time.
U_3 : A: How are you?	U_4 : B: Pretty good.
	...

Each utterance U_i, can be expressed as a sequence of N_i tokens.

$$U_i = \{w_{i,1}, \ldots, w_{i,N_i}\}$$

Each of these token $w_{i,j}$ can be thought of as a random variable that takes on some value in a vocabulary V. Since we are taking a generative approach to this task, we model the dialogue as a joint probability distribution, for a set of model parameters θ [Ser+15]:

$$P_\theta(U_1, \ldots, U_m) = \prod_{i=1}^{m} P_\theta(U_i|U_{<i}) = \prod_{i=1}^{m}\prod_{j=1}^{N_i} P_\theta(w_{i,j}|w_{i,<j}, U_{<i})$$

2.1 Datasets

For this task, we used two datasets:

2.1.1 MovieTriples

The main dataset that was used for this task was the *MovieTriples* dataset, a processed version of the *Movie-DiC* dataset from Banchs et al (2012) [Ser+15; BL12]. The *Movie-DiC* dataset contains subtitles from approximately 500 movies. Movie subtitles are a good source for dialogue since they contain a broad range of conversational topics, long uninterrupted interactions between speakers, and generally have minimal spelling errors [Ser+15].

The *MovieTriples* dataset was created by first breaking the subtitles into a set of dialogues $D_1, \ldots D_n$, and then each dialogue was expressed as a series of utterances $U_1, \ldots, U_m$. Therefore, $m - 2$ sets of triples were generated from each dialogue, i.e.: $\{U_1, U_2, U_3\}, \{U_2, U_3, U_4\}, \ldots, \{U_{m-2}, U_{m-1}, U_m\}$.

2.1.2 SubTle

The second dataset that was used was the non-dialogue corpus SubTle [Ame+14]. This corpus consisted of a set of 5.5 million Question-Answer pairs that were also generated from movie subtitles. This dataset was useful since it provided examples from a vast amount of topics, in contrast to the *MovieTriples* dataset that contained longer dialogues with less topical diversity.

2.2 Preprocessing

Named-entity recognition was performed on each of these datasets using the Python natural language toolkit [BK09]. All names and numbers were replaced with the <*person*> and <*number*> tokens respectively, and all letters were made lowercase in order to reduce the number of tokens in the vocabulary. Each piece of punctuation was also treated as a token. We then kept the 10,000 words that occurred the most and replaced the remainder of the words with the <*unk*> token. Each token was then mapped to an id in order to create a numerical representation of each utterance. Each sequence is concatenated with start (BOS) and end (EOS) of sentence tokens <*s*> and </*s*>, which allow for clean sentence initialization and completion during prediction.

The *MovieTriples* dataset was then formatted such that each input was composed of the concatenation of U_1 and U_2 separated by a special end token <*t*>, and an output utterance U_3. The SubTle dataset simply had an input U_1 and output U_2.

3 Models

As formalized in the *Problem Description*, a dialogue system's objective is to predict a conversational response given a prior discourse context. In this sense, it is essentially a language modeling problem. A generative language model produces a probability distribution P parametrized by parameters θ for dialogues of any length. At the word level, this model should produce a distribution over the next possible words conditioned on the previous context. Sampling is achieved one word at a time through greedy selection from this distribution.

Traditional approaches to language modeling, such as through n-gram count-based models which build probability distributions from word/context co-occurrence rates, suffer from crippling sparsity for larger vocabulary sizes and longer preceding context lengths n. Distributed representations of words as introduced by Bengio et al. (2003) [Ben+03] allow for shared information between tokens regardless of a token's position within an utterance. These representations, or *word embeddings*, are suited for use in neural language modeling [Ben+03] as well as recurrent neural language modeling [Mik+10]. In recent research, recurrent neural network language models have been shown to achieve strong language modeling results without the shortcomings of n-gram approaches, and thus will serve as the basis for our dialogue systems.

3.1 Recurrent Neural Network

A recurrent neural network (RNN) creates a representation for an utterance $U = \{w_1, \ldots, w_N\}$ by iteratively combining the next token w_n in the sequence with the hidden state h_{n-1} calculated for the preceding $n-1$ tokens:

$$h_n = f(h_{n-1}, w_n) = tanh(Hh_{n-1} + I_{w_n})$$

where the hidden state $h_n \in \mathbb{R}^{d_h}$ is a fixed-length representation of the tokens seen so far. For our purposes, the final hidden state h_N is then essentially a temporally-sensitive representation of all the tokens in U. In order to generate a normalized probability for a token v given a context

representation h_n, the *softmax* function is used:

$$P_\theta(w_{n+1} = v|w_1, \ldots, w_n) = \frac{exp(g(h_n, v))}{\sum_{v'} exp(g(h_n, v'))}$$

where the function g is defined as:

$$g(h_n, v) = O_{w_n}^T h_n$$

The matrix $I \in \mathbb{R}^{d_h \times |V|}$ is composed of input word embeddings for each v in the vocabulary V. Similarly, the matrix $O \in \mathbb{R}^{d_h \times |V|}$ contains output word embeddings for each v. Intuitively, the function g outputs higher values when the output representation of a token O_v is more similar to the context representation h_n. The probability of that token appearing conditional on the preceding context is in turn higher. Finally, $H \in \mathbb{R}^{d_h \times d_h}$ is a matrix of learned parameters which maps a preceding hidden state h_{n-1} to the following h_n. Though an RNN has the propensity to learn long-term dependencies from sequences by carrying information across hidden states, information tends to be lost as sequences increase in length. The RNN doesn't possess a very effective way to shield information contained in its hidden state over time as its hidden state is overwritten at each time step by a new representation. To give RNN models more "memory", two alternative recurrent architectures have been employed with success: the long short-term memory unit, and the gated recurrent unit.

3.1.1 Long Short-Term Memory Unit

The long short-term memory unit (LSTM), as first propsed by Hochreiter et al. (1997) [HS97], uses memory gates and a memory cell to control and store information across time steps. At each time step, the hidden state h_n is formed by combining an output gate and its memory cell c_n. The current memory cell c_n is combined with the previous memory cell c_{n-1} through learned input and forget gates. Essentially, the LSTM can preserve specific information across time steps by shielding its external memory cell from certain updates. As its internal state is not fully exposed to updates at each time step, the LSTM demonstrates stronger performance when modeling long-term dependencies. By holding constant certain information over time, it also allows for gradients to propagate error without vanishing as quickly [Hoc91].

3.1.2 Gated Recurrent Unit

The gated recurrent unit (GRU), as first proposed by Cho et al. (2014) [Cho+14], also uses memory gates in order to manipulate the flow of information into and out of its hidden state. Instead of combining previous hidden states and the current representation through the use of f, the current state of the model is a linear interpolation of the previous hidden state h_{n-1} and the newly calculated state $\tilde{h}_n$. The GRU learns an update gate which determines how much the hidden state is updated. Differently from a standard RNN, $\tilde{h}_n$ is calculated using a learned "reset" gate which can forget information held in the previous state. The GRU performs similarly to the LSTM on long-term dependency modeling, and it does so without the use of an external memory unit [Chu+14].

3.2 Sequence-To-Sequence Recurrent Encoder-Decoder

In order to predict a target sequence given an input sequence, we follow the lead of Sutskever et al. (2014) ([SVL14]), which first demonstrated the viability of the neural encoder-decoder structure. Though discussion will refer to model components as RNNs, any RNN can be replaced by an LSTM or GRU architecture. In the sequence-to-sequence recurrent encoder-decoder model, an encoder RNN maps m context utterances $U_1, \ldots, U_m$ to a fixed-length context representation for the previous dialogue. This representation is generated by taking the final hidden state h_N of the encoder model after forward propagating over the context. Then, a decoder RNN maps from this context representation to a probability distribution P_θ over the next possible tokens. A full output sequence is produced by greedily appending the most likely tokens to the context until an EOS token is generated. For the *MovieTriples* dataset, the encoder and decoder are trained end-to-end to maximize the probability of producing the output utterance U_3 given the two input utterances U_1, U_2.

3.3 Hierarchical Recurrent Encoder-Decoder

In the standard recurrent encoder-decoder model, all context utterances are concatenated into a single input sequence before being fed to the encoder. With this set up, the model may have a more difficult time incorporating earlier utterances in the input sequence as the length increases. In order to combat this, Sordoni et al. (2015) [Sor+15] proposed a new hierarchical recurrent encoder-decoder architecture (HRED).

The HRED employs a hierarchical encoder structure which tries to create a more stable representation for the combined input utterances. Similarly to the non-hierarchical model, an *utterance* RNN first creates a fixed-length representation for the input utterances. In this model, however, m separate representations are created for each utterance to form a sequence $\{h_N^{U_1}, \ldots, h_N^{U_m}\}$. Each representation is then sequentially fed through a *context* RNN which forwards this series of utterance representations in order, eventually generating a single overarching context representation, equivalent to h_N of the non-hierarchical model. The final hidden state of the context RNN represents a value of the all the preceding context dialogue. It is then used to initialize the decoder RNN in the same way as before, enabling the prediction of next-token probability distributions.

The HRED structure attempts to capture information from earlier utterances more directly by enabling a second encoder, the context RNN, to save helpful information from each utterance representation produced by the primary encoder, the utterance RNN. Unlike in a standard recurrent encoder-decoder model, dependencies from earlier utterances should be accounted for more effectively as the error from these utterances travels backwards over fewer time steps.

3.4 Bootstrapping Word Embeddings / SubTle

In order to train our model, we used *word embeddings* as introduced in Bengio et al. (2003) [Ben+03]. As discussed, rather than training our model with sparse input tokens, all tokens are first mapped to a dense representation vector of size d_h. This embedding space is continuous with the idea that words of similar representations will have close cosine similarities in the embedding space. Because of our large vocabulary size, this greatly improves the speed of training since the model

will share information between words that are similar in the embedding space.

Our embeddings were initialized using the Word2Vec embeddings from (Mikolov et al. 2013) [Mik+13]. These embeddings were trained on data from Google news which contained over 100 billion words, and have been demonstrated to contain powerful relationships over the vast set of words [1].

As indicated previously, the main dataset that was used for this assignment was *MovieTriples* since it is constructed from a set of dialogues. However, its usefulness is limited by its scope: there are only 179,792 examples. In order to further improve this generalized dialogue system, we pretrained our model on the SubTle dataset which is a Q-A corpus that consists of 5.5 million question and answer pairs. Before training on the MovieTriples dataset, we would pretrain the full model on this Q-A dataset for 3 epochs.

3.5 Evaluation Metrics

3.5.1 Perplexity

The main metric that we used to evaluate and train our models was perplexity defined as:

$$\exp\left\{-\frac{1}{N_W}\sum_{i=1}^{N}\log P_\theta(U_3^n|U_1^n,U_2^n)\right\}$$

for a model with parameters θ, N triples, and $N_W = \sum_{i=1}^{N}|U_3^i|$, where $|U_3^i|$ is the number of tokens in the *ith* triple's U_3 utterance [Ser+15]. We used perplexity as a metric since it measures a model's confusion when it predicts the tokens in U_3. This is a natural extension to the joint probability distribution for a generative model as described in the *Problem Statement* section.

3.5.2 Error rate

A second metric that we adopted was word error rate (WER), taken from Serban et al. (2015) [Ser+15]. Given the correct preceding dialogue context, the error rate is the percent of examples in which the most likely predicted word fails to match the gold target word.

4 Experiments

4.1 Sequence-To-Sequence Recurrent Encoder-Decoder

We experimented with three different types of recurrent encoder-decoder networks. Without pre-training the model on SubTle, the lowest perplexity we achieved was 38.24 using a vanilla 2-layer LSTM with hidden states and word embeddings of size 300. This closely matches the perplexity of 35.63 that Serban et al. achieved without the SubTLe dataset using a GRU [Ser+15]. In this model, word embeddings were preset to Word2Vec embeddings and held fixed [Mik+13].

[1] http://deeplearning4j.org/word2vec

Seeking to further improve this model, we pretrained our model on the SubTle dataset. From doing this, we generate an average reduction of perplexity by 3.52 across our different models. The best perplexity achieved was 34.00 using a GRU with unfixed word vector embeddings. This minor reduction in perplexity matches the reduction in perplexity from 35.63 to 27.09 that Serban et al achieved. The reduction was not as dramatic as Serban et al saw after training his model. A similarly minor reduction in word error rate also occurs between models trained with and without the SubTle dataset. This is primarily because he expanded his model size to a word vector size of 1200 and hidden state size of 400. To keep time for training reasonable, we did not expand our model to this size and likely did not benefit from the substantial amount of extra data.

4.2 Hierarchical Recurrent Encoder-Decoder

In order to fully reimplement Serban et al's paper, we reimplemented hierarchical recurrent encoder-decoder models. For this, we saw no improvement in perplexity from the standard recurrent encoder-decoder model. Our best perplexity achieved with the hierarchical recurrent-decoder model without pretraining on the SubTle dataset increased from a perplexity of 34.00 with the recurrent encoder-decoder to 37.68. This matches what Serban et al. saw where perplexity went from 36.53 to 36.59 in models without pretraining on the SubTle dataset. With pretraining on the SubTle, Serban et al. saw a marginal improvement in perplexity from 27.09 to 26.81. We did not see an improvement in our models between the standard and hierarchical recurrent encoder-decoder. Using the SubTle dataset, our models increased from a perplexity of with the recurrent encoder-decoder of 34.00 to 34.76 with 2-layer bidirectional LSTMs in the hierarchical recurrent encoder-decoder. The word error rate was also comparable between the standard and hierarchical recurrent encoder-decoder models.All models demonstrated comparable word error rates to each other, as well as those from Serban et al.

4.3 Bootstrapping Word Embeddings / SubTle

We generally noticed that using unfixed word embeddings while training with both the SubTle and MovieDialogue datasets seemed to reduce perplexity. Initially, the model without word embeddings and without the SubTle dataset would converge to a perplexity below results produced with fixed embeddings. However, the models trained without fixed word embeddings produced qualitatively lackluster outputs. Word error rates for models with unfixed embeddings were generally lower than their counterparts with fixed embeddings.Visually comparing the predictions generated by models without fixed word embeddings seemed to further confirm this result.

4.4 Generated Dialogue

When generating output, two primary approaches were employed: greedy search, and beam search. Greedy search takes the most likely token given a prior context and appends it to the context iteratively until a model predicts an EOS token. Beam search is similar to greedy search, but instead maintains a series of width k of most likely next tokens. As a series of tokens is predicted at each time step, the beam is consolidated to the k most likely sequences up to that point. A

beam width of $k = |V|$ is an exhaustive search over the generative state space, which is intractable for any significant vocabulary size.

As a consequence of the perplexity metric used for training loss, models tended to predict "safe," vague utterances that were highly likely but not necessarily context-specific. This same behavior has been seen in a number of neural language model approaches [VL15; Ser+15; LJO16]. Li et al. (2015) [Li+15] propose an alternative objective function, the maximum mutual information criterion (MMI) which in our context can be formalized as follows:

$$log(\frac{P_\theta(U_1, U_2, U_3)}{P_\theta(U_1, U_2)P_\theta(U_3)})$$

MMI seeks to maximize the mutual information between the context utterances and the target utterance. Theoretically, this criterion should encourage generative output more specific to the context rather than output that is unconditionally more likely.

Though MMI has not yet been adapted to the encoder-decoder framework, it can be used with models trained for perplexity to score context-specific generated output as more likely at test time. Without MMI, the probability associated with a given context and output is simply the summed likelihoods of each word in the output, given the encoded context. With MMI, the probability associated with each token is reduced by its likelihood as predicted by an RNN language model. Sequence scores are also normalized by length to encourage different length outputs. Qualitatively, the MMI scoring modification results in generated dialogue that strongly outperforms original dialogue. Lexical diversity visibly increases, and the outputs tend to be less safe and more context-specific.

Reference (U_1, U_2)	Predicted	Predicted (+MMI)	Target (U_3)
U_1 : you wish ! U_2 : you knew she was pregnant	1) what ? 2) i ' m sorry . 3) no !	1) what did she say ? 2) what was she doing ? 3) she loved you .	what makes you so sure it was me ? she coulda had lots !
U_1 : oh , no . we ' ll never get it out now . U_2 : so certain are you . always with you it can not be done . hear you nothing that i say ?	1) yes , sir . 2) of course . 3) no !	1) certainly not . 2) no , i don ' t believe you . 3) forget it ..	<person> , moving stones around is one thing . this is totally different .

Table 1: Generated outputs for LSTM bootstrapped from SubTle corpus. The first column shows the input utterances U_1 and U_2, the second column shows the most likely output prediction, the third column shows the most likely output prediction modified by MMI, and the fourth column shows the target utterance U_3.

4.5 Summary

All results are based off of the validation set:

Model	Perplexity	Error rate
GRU (w.o. SubTle)	38.37	0.647
LSTM (w.o SubTle)	38.24	0.642
Bi-LSTM (w.o SubTle)	37.84	0.647
HRED LSTM (w.o SubTle)	38.97	0.648
HRED GRU (w.o SubTle)	38.74	0.647
HRED Bi-LSTM (w.o. SubTle)	37.68	0.649
GRU (unfixed)	34.00	0.643
LSTM (unfixed)	**33.94**	0.641
GRU (fixed)	34.34	0.640
LSTM (fixed)	36.59	**0.637**
Bi-LSTM (fixed)	35.54	0.639
HRED LSTM (fixed)	34.94	**0.637**
HRED GRU (fixed)	35.15	0.639
HRED Bi-LSTM (fixed)	35.75	0.638

5 Conclusion

Overall, we were able to achieve relatively comparable results to current research in conversational modeling, adjusting for model size. Our models and experiments largely followed the design of Serban et al. (2015) [Ser+15], and we were able to improve upon the quality and context-specificity of their models' generated text using MMI, as proposed by Li et al. (2015) [Li+15]. Due to limitations on training time, our models were trained with significantly fewer parameters (smaller word embedding and hidden state sizes) than those we compare against. These smaller models still achieved very promising results in MovieTriples perplexity and word error rate, though they were unable to achieve the same boost in performance from SubTle bootstrapping. Our best model that was not boostrapped from SubTle, the bidirectional HRED LSTM, achieved a perplexity competitive with equivalent models on this dataset.

Our sequence-to-sequence recurrent encoder-decoder and hierarchical recurrent encoder-decoder models were trained using the perplexity objective, which is inherently limited in the task of varied language generation. Future work should seek to employ the maximum mutual information objective during training or find alternative objectives to perplexity which encourage less generic output. Further future work should seek to incorporate longer dialogue contexts beyond the triples found in the MovieTriples dataset.

6 Acknowledgments

The authors acknowledge Harvard University and the professors and teaching staff of CS 287 for support and resources. The authors thank Alexander Rush and Yoon Kim for constructive feedback. The authors also thank Iulian Serban for providing the *MovieTriples* and *SubTle* datasets, and Rafael Banchs for providing the *Movie-DiC* dataset.

7 Code

All code (excluding datasets) can be found at the repository here:
https://github.com/michaelfarrell76/End-To-End-Generative-Dialogue.

References

[Hoc91] S. Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München*. 1991.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[Ben+03] Yoshua Bengio et al. "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3 (Mar. 2003), pp. 1137–1155. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=944919.944966.

[BK09] Edward Loper Bird Steven and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media Inc. 2009.

[Mik+10] Tomas Mikolov et al. "Recurrent neural network based language model". In: *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*. 2010, pp. 1045–1048. URL: http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html.

[BL12] Rafael E. Banchs and Haizhou Li. "IRIS: A Chat-oriented Dialogue System Based on the Vector Space Model". In: *Proceedings of the ACL 2012 System Demonstrations*. ACL '12. Jeju Island, Korea: Association for Computational Linguistics, 2012, pp. 37–42. URL: http://dl.acm.org/citation.cfm?id=2390470.2390477.

[Mik+13] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* abs/1301.3781 (2013). URL: http://arxiv.org/abs/1301.3781.

[Ame+14] David Ameixa et al. "Intelligent Virtual Agents: 14th International Conference, IVA 2014, Boston, MA, USA, August 27-29, 2014. Proceedings". In: ed. by Timothy Bickmore, Stacy Marsella, and Candace Sidner. Cham: Springer International Publishing, 2014. Chap. Luke, I am Your Father: Dealing with Out-of-Domain Requests by Using Movies Subtitles, pp. 13–21. ISBN: 978-3-319-09767-1. DOI: 10.1007/978-3-319-09767-1_2. URL: http://dx.doi.org/10.1007/978-3-319-09767-1_2.

[Cho+14] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078 (2014). URL: http://arxiv.org/abs/1406.1078.

[Chu+14] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3555 (2014). URL: http://arxiv.org/abs/1412.3555.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks". In: *CoRR* abs/1409.3215 (2014). URL: http://arxiv.org/abs/1409.3215.

[Li+15] Jiwei Li et al. “A Diversity-Promoting Objective Function for Neural Conversation Models”. In: *CoRR* abs/1510.03055 (2015). URL: http://arxiv.org/abs/1510.03055.

[Ser+15] Iulian Vlad Serban et al. “Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Networks”. In: *CoRR* abs/1507.04808 (2015). URL: http://arxiv.org/abs/1507.04808.

[Sor+15] Alessandro Sordoni et al. “A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion”. In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. CIKM ’15. Melbourne, Australia: ACM, 2015, pp. 553–562. ISBN: 978-1-4503-3794-6. DOI: 10.1145/2806416.2806493. URL: http://doi.acm.org/10.1145/2806416.2806493.

[VL15] Oriol Vinyals and Quoc V. Le. “A Neural Conversational Model”. In: *CoRR* abs/1506.05869 (2015). URL: http://arxiv.org/abs/1506.05869.

[LJO16] Yi Luan, Yangfeng Ji, and Mari Ostendorf. “LSTM based Conversation Models”. In: *CoRR* abs/1603.09457 (2016). URL: http://arxiv.org/abs/1603.09457.