TrumpBot: Seq2Seq with Pointer Sentinel Model

Filip Zivkovic
Department of Computer Science
Stanford University
zivkovic@stanford.edu

Derek Chen
Department of Computer Science
Stanford University
derekchen14@stanford.edu

Abstract

Recurrent neural networks have become prominent across many natural language processing tasks, from named entity recognition to machine translation. We apply a Seq2Seq model to the goal of creating a chatbot able to return self-generated text responses based on arbitrary inputs. In doing so, we explore the benefits of encoder-decoder networks, trade-offs between lacking data versus cross-training, various attention scoring functions, and finally, an attempt at adapting Pointer Sentinel Mixture Model to a dynamic length decoder. In order to create a chatbot of specific persona, we also scraped Twitter and the general web for public data of speech in the style of Donald Trump.

1 Introduction

Advancements in conversational modeling have had, and continue to have, broad impact across academic research and global industry. Automated language comprehension in conversation form could become a scalable and a highly regarded method of interface for various data retrieval tasks. Recent advances in machine translation [15, 3], context-aware query suggestions [20] and describing multimedia content [2] show sequence-to-sequence models as a promising foundation towards solving previously intractable natural language processing tasks. To this end, we employ a Seq2Seq model with attention as the core network in creating a chatbot capable of ingesting variable length queries and returning variable outputs similar to those in [24], [18] and [19].

Past researchers have expressed difficulty in training a generalized open-domain chatbot, thus it was elected to restrict our subject matter to conversations of a political nature. Furthermore, we proceed to further limit scope by attempting to embed the chatbot with a persona around the newly elected President Donald Trump, using ideas gleaned from [11]. Trump was chosen as the target persona for his uniquely identifiable speech patterns and large online presence, providing a good foundation for data acquisition.

Pointer Sentinel Mixture Model (PSMM) [16] solves a seemingly simple task, nevertheless, the motivation and methods are far more intricate. PSMM allows words used within the encoder to be directly copied forward to the decoder. Upon analyzing the frequency of words, we find that most corpus's have a long right tail of interesting words that simply don't arise very often. Taking advantage of the temporal nature of RNN's, PSMM reduces the perplexity scores associated lacking data haunting rare vocabulary words.

Nonetheless, PSMM has been designed for a fixed length encoder/decoder, which comes with two strong drawbacks. First being that there is no clean implementation for training the PSMM concurrently with a Seq2Seq model. Second, having a decoder length of one, yet backpropagating 100 timesteps with RNN's is not computationally efficient. Forward-propagation and back-propagation with RNN's is sequential, requiring previous timesteps to complete before concurrent timesteps can begin. We used spare time at the end of this project to investigate two simple methods to extend PSMM to dynamic length, and in doing so, iterate over the dataset in a more computationally efficient manner.

2 Related Work

2.1 Encoder Decoder Models

Inspired by the work from a Neural Conversational Model [24], we saw great potential in being able to create a working chatbot that was capable of generating relevant and meaningful responses to arbitrary input queries. In their work, Vinyals and Le were able to train a bot to respond accurately to IT Helpdesk queries using a relatively straightforward seq2seq model. Findings from Serban et. al. [18] were able to extend this model to non-goal-driven systems. From our understanding, this approach differs from previous work since its main target aims to emulate the dialogues of real humans rather than simply return a correct answer to a specific query.

The best baselines we could find on dialog corpora was published by Kaldec et. al. [9] who performed their studies around ranking responses given an input query. Their best models used a Bi-LSTM, which served as a good starting point for our goals as well. To get closer to target of generating responses we used the encoder-decoder network from [4], which has proven quite popular.

2.2 Pointer Models

Pointer Networks [23] learn the conditional probability of an output sequence with elements that are discrete tokens corresponding to positions in an input sequence, which is able to refer back to a variable length input for predicting an output. Whereas a traditional classifier require the size of the output vocabulary to be fixed beforehand, a pointer network creates a vocabulary on-the-fly for each new example by using the input sequence as the dictionary of possible words to choose from. This method is especially useful when the correct answer is a specific name or unique slang found in the input, but is too rare to be found in most corpora. However, there is the drawback that the model now fails to take advantage of the large vocabulary that is actually available, often as dense word representations which can offer further training benefit or sparsely encoded discrete words.

Pointer Sentinel Mixture Models [16] is a recent, clever, and simple method which produces state-of-the-art perplexity scores on Penn Tree Bank and Wiki Text datasets with only a medium size model. One key design trait is the inclusion of the learned mixture gate within the sentinel softmax. The Pointer Sentinel Mixture Model distinguishes when to use the default RNN softmax for predicting the next word against the pointer network, as regulated by:

$$p(y_i|x_i) = g\ p_{vocab}(y_i|x_i) + (1-g)\ p_{ptr}(y_i|x_i) \tag{1}$$

The trainable parameters added by PSMM are the affine tranformation used to generate q, and the sentinel s, where both s and q are of the same dimensions as h for the LSTM's. Specifically, the q below represents a query used to derive g for the model.

$$q = \tanh(Wh_{N-1} + b) \tag{2}$$

s in turn represents an additional input that is appended to the encoder inputs. In effect, s allows the gate of the pointer network to choose between the encoder inputs or the RNN. Concretely, when g from Eq(1) is high, the pointer distribution will be disregarded, ultimately using the Softmax RNN to generate the next word. Alternatively, when g is low, the pointer network will lean towards selecting an item among the inputs, going back L words. For more information and detailed explanation, please refer to the original paper [16].

3 Model

3.1 Encoder Decoder Network

The core of our conversational model was built using a Seq2Seq architecture that featured two RNNs, an encoder used to build up a hidden state representation of an input query, and a decoder used to generate an output response. Given that our maximum bucket size was 60 words, we used GRU [5] cells for both the encoder and decoder in order to minimize the vanishing gradient problem.

$$z = \sigma(x_t U^z + s_{t-1} W^z)$$

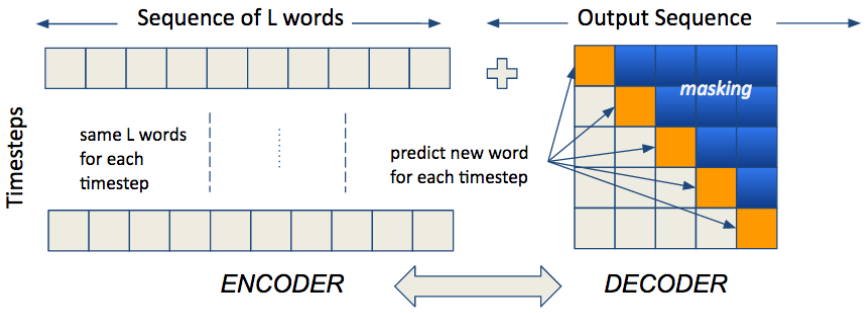

Figure 1: Extending the pointer sentinel mixture model to be able to predict dynamic length outputs required updating the backpropagation method such that gradients from the predicted words (orange) were passed back, while the gradients from empty padding (dark blue) were not used for weight updates. In this case, five predictions would be made, the last of which would use L+4 words in it's context.

$$
\begin{aligned}
r &= \sigma(x_t U^r + s_{t-1} W^r) \\
h &= \tanh(x_t U^h + (s_{t-1} \circ r) W^h) \\
s_t &= (1 - z) \circ h + z \circ s_{t-1}
\end{aligned}
$$

Ideally the hidden state of each cell would allow the gradients to flow through smoothly until the gates felt it was time to forget that particular signal. Conversely, we also used gradient clipping top to handle the exploding gradient problem.

In order to allow the final hidden state of the encoder to store the most amount of information, we reverse the inputs on the encoder following the style of [22]. Given the amount of padding added to certain sentences, we noticed that flipping the inputs caused noticeable increase in performance. Similarly, we also considered using a bi-LSTM [7] in order to encode even more information into the network. Given the limited time, we opted instead for increasing layer size of the encoder RNN, which was much more straightforward to implement. Lastly, the chatbot utilized GloVE vectors as distributed word embeddings in order to gain semantic understanding of the text [17]. Given our relatively limited data size, we opted not to let the word embeddings be trainable. Words that were not found in the GloVE vectors were initialized randomly.

3.2 Pointer Sentinel Mixture Model

Our tenorflow code of the Pointer Sentinel model was built off work from Zaremba et al. [27]. This model also served as the foundation by Merity et. al., indicating that Zaremba et al. is the most appropriate model to compare against. Additionally, in order to get a more direct comparison to the work of Merity, our implementation also used LSTM cells [8] rather than the GRUs mentioned earlier, which we'd used for the Seq2Seq model.

3.3 Pointer Sentinel for Variable Length Generation

In an effort to expand the Seq2Seq model, we took a cue from Luong et.al. [15] who varied the amount of decoders and encoders on such a network to perform a variety of tasks. Concretely, we attempted to take the results of a single encoder to feed into both a traditional RNN encoder, as well as a Pointer Model, effectively yielding a Pointer Sentinel Mixture Model capable of predicting variable length outputs.

A standard PSMM would take the L words from the encoder to predict a single output word in the decoder section, represented by an orange square. As seen in Figure 1 above, we extend this model for dynamic length outputs. Please refer to the caption for a complete description.

The most straight forward method for applying PSMM to a Seq2Seq model is without making any changes in the architecture, with the only novelty being the changes in training as outlined in Figure 1. A second adaptation, the next logical choice, is the expand s to be of dimensions hidden-state by

decoder length. A third choice, which we would implement next had we had more time, would be to keep s the same, but increase the size of the affine transformation. In either of these cases, the goal is to produce a q that is unaltered based on changes to the input sequence length.

4 Experiments and Results

This type of issue is readily apparent through an example. Suppose we ask a person, "How was your day?". This seemingly innocuous question could generate a wide variety of answers including:

1. "I had a wonderful time at the park today playing on the swing and on the slides. We had a picnic afterwards with sandwiches and chips."
2. "Meh."
3. "Don't want to talk about it."

From a person's perspective, all of these answers are not only valid, but carry a weight of information uncorrelated with the length of response or location of vectors in a word embedding space. Given such varied accurate answers, it is quite difficult to train a conversational model to predict with a high degree of confidence which one is the "most accurate". In fact, papers such as [12] suggest that most traditional measures of NLP tasks, such as BLEU score, ROGUE and METEOR are unsuitable measures of dialog systems because most systems assumes only one ground truth response is available given each context, whereas the reality is that multiple plausible responses are possible when talking to a person. Consequently, we ended up turning heavily towards human evaluation when determining the strength of the conversational model.

4.1 Data

A custom dataset was created from speeches, interviews, and presidential debates, resulting in half of the collected 16,876 QA pairs. A strong effort put into parsing data into appropriate Q/A pairs that had adequate length and content.

The other half of the dataset came from twitter. The content was filtered based on keyword, and the keywords were iteratively and interactively determined. Proceeding this, the keywords were reused to help rank each tweet with it's most similar tweets. The similarity measurement was simple: first ranking based on keywords, proceeded by secondary words if the count of common keywords was a tie. The most similar tweet in the corpus became the question, and each tweet was gauranteed to be used once and only once as an answer. This strategy gave us fairly clean data for minimal effort, saving an estimated 80 hours of manual cleaning that would have been necessary otherwise.

The final split amongst $(train, validation, test)$ was (12800, 2560, 1516) QA pairs each. With the addition of cross training, the train set grew by 220,000 QA pairs from Cornell Movie Dialogue Corpus. None of the Movie Dialogue QA pairs were added to the validation set, as we still wished to minimize the perplexity of the original trump data solely.

Since PSMM tests needed quantitative results against a known baseline, these tests were performed on Penn Tree Bank database.

4.2 Attention

We tested three different types of attention. From the tensorflow library, the default option available was bahdanau [1]. From class, we learned that luong [14] might be better because the weight matrix causes a direct interaction between the encoder hidden state and the decoder hidden state, rather than a concatenation of the two. Additionally, we also looked at using the attention mechanism from Vinyals et. al. [25] because it seemed possible that the extra parameters associated with the scoring function could give a boost in performance.

$$score(h_t, \bar{h}_s) = \begin{cases} h_t^\top \bar{h}_s & \text{if Bahdanau} \\ h_t^\top W_a \bar{h}_s & \text{if Luong} \\ v_a^\top \tanh(W_a[h_t; \bar{h}_s]) & \text{if Vinyals} \end{cases}$$

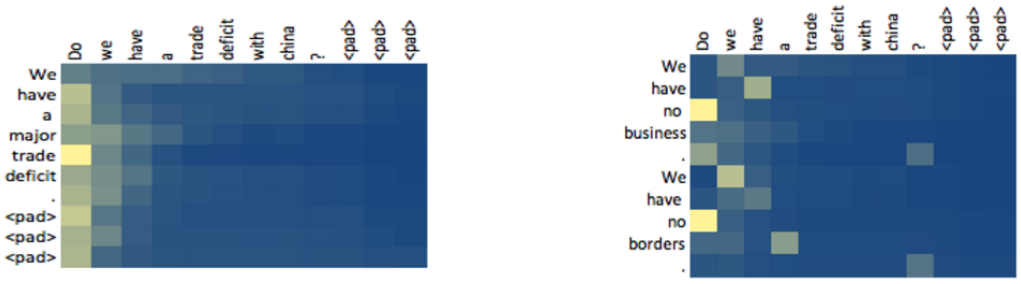

Figure 2: On the left, the chart displays the results of the attention scoring using the method from Vinyals et al. [25]. Conversely, the chart on the right displays the results of attention scoring using the method from Luong et. al. [14]

Our interpretation of visualizations 2 is that as the model is predicting each word in the decoder, it simply looks back at the first word in the encoder. Since the encoder inputs were flipped, this first word represents the last hidden state of the encoder, which is disappointing since the goal of using attention is to allow the model to look beyond that last hidden state for more contextual information.

Figure 2 above does spark another hypotheses. Perplexity scores on the validation set determined when to stop training, and it seems that the Vinyals attention model is under-trained. However, the Luong model has less hyperparameters, and thus actually trains faster. It is suitable to say, based on these visualizations and reasoning, Luong attention is better suited when smaller data sets are available.

4.3 Cross Training and Qualitative Results

The core concept to cross-training was by modifying the cost function directly, using the following equations.

$$J = \sum_{t=1}^{M} m^{(t)} \, CE(y^{(t)}, \hat{y}^{(t)}) \tag{3}$$

$$m = \begin{cases} 0.03 & \text{if Movie data} \\ 1.0 & \text{if Trump data} \end{cases}$$

We used the movie from from Cornell Movie Dialogue Corpus [6]. We chose this corpus specifically because Serban et al. produced successful results using it[18]. We also considered using the Ubuntu Data Corpus [13], which has nearly a million Q/A pairs; however, we had already mitigated the challenge of finding additional cross-training data sufficiently.

These are qualitative results we have generated from three separate models:

Query	Vinyals - No Cross Training	Luong - No cross training	Luong - w/ Cross Training
Do we have a trade deficit with China?	we have a trade deficit .	we owe japan .	no .
Nuclear weapons, iran, and war have what in common?	It 's a mess .	iran is taking our economic dollars dollars .	the army .
The president is terrible	it is time to be president .	i beat hillary clinton	elitism
Crooked hillary is running for president, will she win?	crookedhillary is unfit to serve . bigleaguetruth debate	hillary will change the voter debate . bigleaguetruth debate	hillary clinton wants to makeamericagreatagain

Especially when we heavily weighted movie data, we noticed the generation of conversational responses tend to generate generic, safe responses regardless of the input, such as "i don't know." As we shifted the importance over to just using the custom Trump Data, common phrases from our data, such as makeamericagreat" and "competition" would continually show up. To combat this issue, we looked into work from [10] which suggested using Maximum Mutual Information (MMI) as the objective function in neural models, rather than the traditional objective function of lowering perplexity. We did not have time to implement such a cost function, but it is certainly something to explore in the future to produce more diversity, perhaps allowing us to take advantage of even larger datasets, such as the Ubuntu Dialog Corpus [13].

We found that things that are political in nature tend to have better outcomes, which makes sense because we specifically trained on political sentences and purposely removed sentences related to his business, hotels, and other investments.

4.4 Pointer Sentinel

Training PSMM has proven to be sensitive to parameter tuning; however, the biggest surprise came when introducing the Adam Optimizer. Adam Optimizer (with learning rate 0.005) in replacement of Stochastic Gradient Descent results in exploding gradients shortly into training. By closely monitoring the mixture gate, we see that g slowly saturates to 0.99 during initial stages of training, and then as soon as a single step is taken towards reducing the mixture gate, it drops close to zero once, and then gradients explode the proceeding update. Adam Optimizer leverages momentum and velocity, which are perhaps overcompensating within the PSMM architecture.

Tuning regularization parameters was critical for training. Beginning with variational inference based regularization as described by Merity, with 0.5 on the input to each RNN layer and 0.5 on the final output resulted in g toggling correctly early on in training, but both the validation and training perplexities would not go below 200. We found o.5 to be too much regularization. On the other spectrum, too little regularization, and g would remain saturated at 0.99 throughout the the entirety of training. With too little regularization, the model was over-fitting to always use the RNN distribution. A dropout value of 0.8 was optimal. Had we had more time remaining, we would repeat the same test and just increasing the hidden state size from 200 to 650, likely achieving much closer scores to the perplexity of 80.6 produced by Merity [16]. However, the purpose of this test was a direct comparison with similar hyper-parameters, and it is only the relative difference that we were interpreting rather than the absolute value.

Having a stable implementation of PSMM, the next goal was to produce a proof of concept for adapting PSMM to a Seq2Seq model. Applying the unmodified PSSM as is, the visualizations indicated that the mixture gate was learning incorrectly. The pointer distribution was being used for common words such as 'to', 'the', and '<eos>'–while words such as 'mr.' or 'dr.' rarely entered the distribution. One possible explanation is that s could not map the desired behavior, and learning each training step was competing with the next. The hidden state of the LSTM's has a variable number of forward propagations, and that variance is not being captured in this overly biased model.

Model	decoder length	Hidden Size	layers	epochs	train	validation	test
Zaremba, et. al.	n/a	200	2	13	37.99	121.39	115.91
PSMM	1	200	2	13	60.41	95.89	92.17
Modified PSMM	10	200	2	7	82.77	107.14	101.93

Table 2: Two PSMM models as compared to the baseline model

The next intuitive approach was extending s to be a matrix of hidden-state by decoder length. In this case, we would hope that introducing a new dimension allows s to map all relationships regarding the variable number of forward propagations, while the affine transformation leading into the tanh function would learn general rules regarding when to turn on the mixture gate. We call this model Modified PSMM, and the results are in the above table.

The introduction of one more dimension to s shows improvements to map correct over the baseline model. As indicated by visualizations, there are relatively more errors within the gating function in the Modified PSMM model than when the decoder length is simply 1. We hypothesize, that although the model has difficulty balancing the mixture gate, there is still some beneficial learning that has occurred when utilizing the mixture gate. Lastly, the processing speed for Modified PSMM was 250 words per second, whereas PSMM only processed at 130 words per second. This was expected, due to the parallelism we have introduced into the model.

5 Conclusion

The main reason for implementing the Pointer Sentinel Mixture model was to eventually integrate its benefits into the encoder decoder model to allow predictions on rare or OoV words. Thus, augmenting PSMM to generalize to variable length decoding would certainly be the next step given more time.

Separately, in order to improve the performance of the Seq2Seq model, we would have liked to turn to adding the notion of intention [26] to the network where hidden state from the encoder network is fed into an third RNN that models the intention process. Specifically, this dynamic process looks at the back and forth nature of the a conversation where one turn is dependent on the intention in the previous turn. Similarly, work from [21] seems directly applicable in generating context-aware utterances that take into account previous dialog between the two users. This is especially true since their training was heavily influenced by the use of Twitter data, similar to our data set. Furthermore, we would return to adding Beam Search on Trumpbot, following through with our initial efforts to do so.

On the Pointer Model side, we believe there are large gains to be made from simply increasing the hidden states size from 200 to 650 since this would directly increase the parameters in the system to model conversation. In order to fully replicate the original paper's regularization [16], and also to combat this increased size, we would also plan to add Zoneout regularization. Evaluating the Seq2Seq model was difficult when the effectiveness of hyperparameters since perplexity scores were insufficient. Adding more measures of accuracy to gauge training quality, such as BLEU scores, would thus enable us to additionally automate a full hyperparameter search.

Acknowledgments

We would like to acknowledge the mentorship and weekly support of Richard Socher throughout the development of this project. We would also like to thank the countless TAs and other students who helped throughout the project in Office Hours and late night coding sessions.

References

[1] Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[2] Cho, Kyunghyun, Courville, Aaron, and Bengio, Yoshua. "Describing multimedia content using attention-based encoder-decoder networks". In: *IEEE Transactions on Multimedia* 17.11 (2015), pp. 1875–1886.

[3] Cho, Kyunghyun et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[4] Cho, Kyunghyun et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[5] Cho, Kyunghyun et al. "On the properties of neural machine translation: Encoder-decoder approaches". In: *arXiv preprint arXiv:1409.1259* (2014).

[6] Danescu-Niculescu-Mizil, Cristian and Lee, Lillian. "Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs." In: *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*. 2011.

[7] Graves, Alex and Schmidhuber, Jürgen. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural Networks* 18.5 (2005), pp. 602–610.

[8] Hochreiter, Sepp and Schmidhuber, Jürgen. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[9] Kadlec, Rudolf, Schmid, Martin, and Kleindienst, Jan. "Improved deep learning baselines for ubuntu corpus dialogs". In: *arXiv preprint arXiv:1510.03753* (2015).

[10] Li, Jiwei et al. "A diversity-promoting objective function for neural conversation models". In: *arXiv preprint arXiv:1510.03055* (2015).

[11] Li, Jiwei et al. "A persona-based neural conversation model". In: *arXiv preprint arXiv:1603.06155* (2016).

[12] Liu, Chia-Wei et al. "How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation". In: *arXiv preprint arXiv:1603.08023* (2016).

[13] Lowe, Ryan et al. "The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems". In: *arXiv preprint arXiv:1506.08909* (2015).

[14] Luong, Minh-Thang, Pham, Hieu, and Manning, Christopher D. "Effective approaches to attention-based neural machine translation". In: *arXiv preprint arXiv:1508.04025* (2015).

[15] Luong, Minh-Thang et al. "Multi-task sequence to sequence learning". In: *arXiv preprint arXiv:1511.06114* (2015).

[16] Merity, Stephen et al. "Pointer Sentinel Mixture Models". In: *arXiv preprint arXiv:1609.07843* (2016).

[17] Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. "Glove: Global Vectors for Word Representation." In: *EMNLP*. Vol. 14. 2014, pp. 1532–1543.

[18] Serban, Iulian V et al. "Hierarchical neural network generative models for movie dialogues". In: *CoRR, abs/1507.04808* (2015).

[19] Shang, Lifeng, Lu, Zhengdong, and Li, Hang. "Neural responding machine for short-text conversation". In: *arXiv preprint arXiv:1503.02364* (2015).

[20] Sordoni, Alessandro 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*. ACM. 2015, pp. 553–562.

[21] Sordoni, Alessandro et al. "A neural network approach to context-sensitive generation of conversational responses". In: *arXiv preprint arXiv:1506.06714* (2015).

[22] Vinyals, Oriol, Bengio, Samy, and Kudlur, Manjunath. "Order matters: Sequence to sequence for sets". In: *arXiv preprint arXiv:1511.06391* (2015).

[23] Vinyals, Oriol, Fortunato, Meire, and Jaitly, Navdeep. "Pointer networks". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2692–2700.

[24] Vinyals, Oriol and Le, Quoc. "A neural conversational model". In: *arXiv preprint arXiv:1506.05869* (2015).

[25] Vinyals, Oriol et al. "Grammar as a foreign language". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2773–2781.

[26] Yao, Kaisheng, Zweig, Geoffrey, and Peng, Baolin. "Attention with intention for a neural network conversation model". In: *arXiv preprint arXiv:1510.08565* (2015).

[27] Zaremba, Wojciech, Sutskever, Ilya, and Vinyals, Oriol. “Recurrent neural network regularization”. In: *arXiv preprint arXiv:1409.2329* (2014).