Sequence-to-Sequence Generative Argumentative Dialogue Systems with Self-Attention

Ademi Adeniji
Stanford University
ademi@stanford.edu

Nate Lee
Stanford University
natelee@stanford.edu

Vincent Liu
Stanford University
vliu15@stanford.edu

Abstract

Natural language generation is an area of natural language processing with much room for improvement. Argument mining poses the problem of responding relevantly and argumentatively to input in addition to the challenge of generating coherent text. In this paper, we explore the possibility of creating a dialogue agent capable of holding argumentative discussion. The architecture we propose employs local self-attention to encode and decode individual responses and global attention to invoke information from past exchanges. We achieve 25% word accuracy rate on the Internet Argument Corpus v1, beating the Transformer and traditional RNN-style sequence-to-sequence architectures. Additionally, our architecture can easily be generalized to other discussion-style scenarios, given appropriate data.

1 Introduction

Text generation is crucial to creating artificial systems that can communicate proficiently with humans. Successful natural language generation is an indicator of the proficiency of natural language understanding, the core goal of creating robust natural language systems. Natural language processing (NLP) systems have been successful with probabilistic methods and pointers to extract answers, as with datasets like SQUAD [4], however such methods have seen limited success in natural language generation (NLG).

The problem in NLG with respect to argumentation is that core understanding of language with linguistic variances and ambiguities is difficult for fixed traditional neural networks [5]. Furthermore, argument mining is a subset of text generation, but with the added constraint of responding within the context of a dialogue and argumentation. The system has to not only generate appropriate argumentative responses to a local context but also retain global context of past exchanges in the same conversation to inform its responses. For example, the current state-of-the-art generative model in [11] often produces irrelevant responses. This indicates that existing approaches still fail to capture the core understanding of arguments within an NLG training corpus.

In recent literature, the concept of attention in traditional seq2seq models have dramatically increased model efficacy as seen in machine translation tasks in NLP [7], and question answering tasks. Moreover, attention has increased model intrepretability and created neural models more similar to how humans understand language [8].

In this paper, we propose an argumentative NLG architecture that is fundamentally based off of attention for greater language understanding. Our model, based on the Transformer [7], is built fundamentally off self-attention mechanisms to encode and decode argumentative posts. To invoke discussion context, we leverage memory of previous posts with LSTM cells which help encode memory and context into our encodings. Ultimately, we were able to achieve lower perplexity and better subjective argumentation relevance than the state of the art RNN style seq2seq models.

2 Related Work

Natural language processing and specifically argument mining have seen drastic improvements. Below, we describe relevant recent work that we leverage in our experiments.

2.1 Retrieval Systems

The retrieval-based model proposed in [11] uses a Manhattan LSTM network to select the most relevant response from a list of potential responses, given a user's message. Another method proposed in [6] leverages Latent Semantic Similarity to score a corpus of clustered counter-arguments by their similarity to the user query. This method, however, suffers from inefficiency and scalability. Even with its greedy thresholding optimization for selecting high scoring responses, preparation and traversal of these clusters grow in complexity with the number of possible user query topics (Only three discussion topics were allowed in this experiment). However, all retrieval-based approaches are limited to only generating responses that are present in its database.

2.2 Generative Systems

Generative models eliminate the need for prepared responses or expensive similarity searches and allow the system to generate new responses every time. The current state-of-the-art generative model used in [11] uses a hierarchical recurrent neural network, encoding and decoding at one level and updating a conversation-level state at another. Concretely, a bidirectional GRU encodes input, the output of which is used to update a conversation-level RNN, whose output is then decoded with another RNN to produce a response. At the cost of richer semantic elements and unlimited response diversity, the generative model often misinterprets arguments or produces irrelevant responses.

2.3 Self-Attention

The current trend in natural language processing is to substitute self-attention for recurrence, as proven successful by [7]. While self-attention is better for efficiency and performance, it has also proven to be a difficult mechanism to harness. To accomodate for positional invariance induced by self-attention, the authors utilize positional encodings and residual connections between layers to ensure that positional signals are propagated throughout the entire model. Following keystone language modeling work by [9], [12], we also leverage the Transformer-style architecture to encode and decode individual responses.

2.4 Word Embeddings

The authors in [11] use Word2Vec embeddings to encode the input tokens to the model. Recent work in creating better word embeddings have turned to utilizing final hidden states of complex neural networks trained for language modeling [9], [10], [12]. For machine translation tasks, word embeddings are trained from scratch [7]. Despite the efficacy of contextualized word embeddings, we opt to fine-tune GloVe embeddings [3] for simplicity. We do not train our own embeddings from scratch as the dataset we use does is not as large as those used for machine translation.

3 Approach

As previously mentioned, our model leverages the Transformer [7] architecture for local self-attention and global attention to generate relevant responses efficiently. Our model is a sequence-to-sequence architecture at the conversation level and a Transformer architecture at the response level. Below, we describe our dataset and model in detail.

3.1 Baseline

Because there has not been significant work done with this specific corpus, we build our own traditional RNN-style seq2seq model for memory-less baseline training using bi-LSTMs (referred to as the biLSTM baseline). In terms of baseline metrics, we use the MLE objective as well as their perplexity numbers from [11]. Additionally, the authors of [11] provide an online interactive chatbot,

Table 1: Perplexity on testing sets with sampling and re-ranking described in Section 2.3. 'Sampling' refers to sampling words from the output distribution and 'reranking' refers to reranking the samples based on maximum mutual information (MMI) score which are described in greater detail in [11].

Model	Sampling	Sampling & MMI re-ranking
Context-sensitive Perplexity	88.51	76.97 (-13.0%)
Context-free Perplexity	75.53	73.17 (-3.1%)

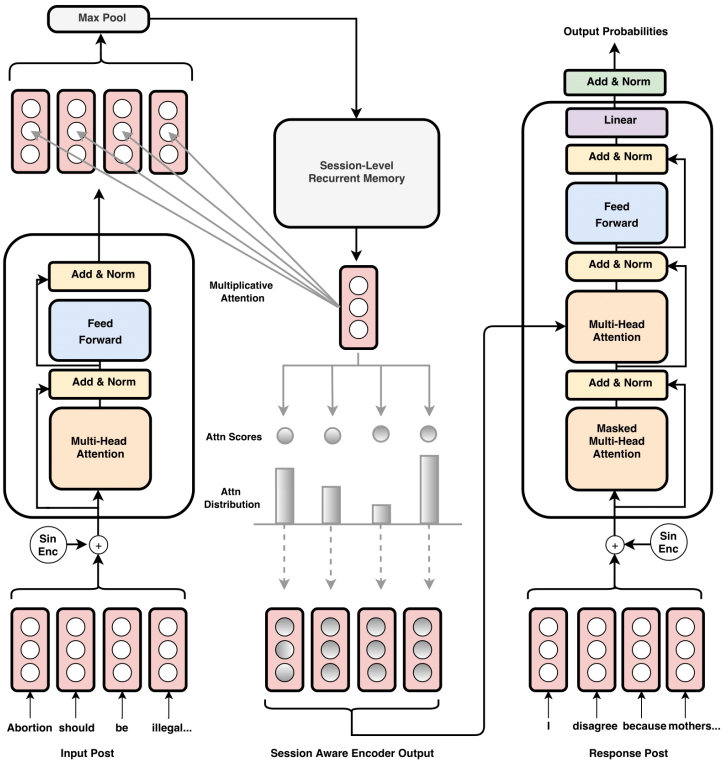

Figure 1: Our memory-enhanced Transformer Architecture - input sequences are fed into the encoding transformer block. Attention is used to generate an argument query encoding and is fed through an LSTM for memory. The encoding vector is finally is passed into our decoder for response generation.

which we can use against our model for qualitative comparison. Table 1 list perplexities reported in the paper.

3.2 Dataset

We train our model on the Internet Argument Corpus Dataset-v1 [2], which contains 11,800 discussion threads on various debate topics with about 390,000 responses (posts) total. We define a training instance to be a discussion, d, which is a sequence of posts. Each post, p is a sequence of tokens, w. We pad and truncate all discussions and posts to maximum lengths, m and n, respectively. We formally define our dataset as follows:

$$d = [\, p^{(1)}, p^{(2)}, ..., p^{(m)} \,]$$
$$p^{(i)} = [\, w_1^{(i)}, w_2^{(i)}, ..., w_n^{(i)} \,]$$

Our gold instances are the same discussion instances except staggered forward by a post such that the label for $p^{(i)}$ in some discussion is the next post $p^{(i+1)}$.

`for` $p^{(i)}$ `in` d:

1. Embed each word in $p^{(i)}$ as a sum of its GloVe lookup and positional encoding of size d_m:
$$\texttt{Embedding:}\ \mathbb{R}^n \mapsto \mathbb{R}^{n \times d_m}$$
2. Encode the embedded sequence with the transformer encoder with multi-headed self-attention and position-wise feed-forward layers:
$$\texttt{Encoder:}\ \mathbb{R}^{n \times d_m} \mapsto \mathbb{R}^{n \times d_m}$$
3. Max-pool over the encoder output and pass this through the LSTM of hidden size d_h:
$$\texttt{MaxPool:}\ \mathbb{R}^{n \times d_m} \mapsto \mathbb{R}^{d_m}$$
$$\texttt{LSTM:}\ \mathbb{R}^{d_m} \mapsto \mathbb{R}^{d_h}$$
4. Attend to the encoder output with this globally-aware query from the LSTM. We opt to use dot product attention if $d_m = d_h$, but multiplicative attention otherwise, softmax the generated distribution, and weight each position in the encoder output with its attention weight.
$$\texttt{GlobalAttention:}\ \mathbb{R}^{n \times d_m}, \mathbb{R}^{d_h} \mapsto \mathbb{R}^{n \times d_m}$$
5. Decode, with the transformer decoder, this attended encoder output with masked multi-headed self-attention and position-wise feed-forward layers:
$$\texttt{Decoder:}\ \mathbb{R}^{n \times d_m} \mapsto \mathbb{R}^{n}$$
6. Accumulate log-likelihood probabilities for cross-entropy loss and compute accuracy and perplexity.

Figure 2: Forward pass through our memory-enhanced Transformer model.

3.3 Model: Memory-Enhanced Transformer

We feed mini-batches of our 11,800 discussion instances into our model, whose forward pass for one example is as follows. Due to CUDA/GPU memory constraints, we are restricted to batches of size 4. We borrow the Transformer architecture[1] for steps (2) and (5), but design the rest ourselves. For in-depth explanations of the Transformer encoder and decoder, see [7].

3.4 Perplexity and Accuracy

As used in [11], we will adopt the following perplexity metric from [1], $PPL = 2^{-\sum_y \log P(y)}$, where $P(y)$ is the log-likelihood probability. We also calculate word accuracy rate as an indicator of learning during training.

We describe our most significant experiments as well as report our performance metrics and qualitative results. We define accuracy in terms of exact token by token match of input and output sequences.

4 Experiments

4.1 biLSTM Seq2Seq Baseline

For our custom baseline, we borrow the traditional RNN-style seq2seq model with a biLSTM encoder and LSTM decoder for a preliminary study of argument generation. We use a word-level model, since the character-level model (not shown) consistently produces "unsure" responses for any given input. Although perplexity was able to hit approximately 90, we still maintained high loss throughout. We hope to improve upon this baseline in terms of bias using our memory-enhanced Transformer model.

[1] Code adapted from `https://github.com/jadore801120/attention-is-all-you-need-pytorch`. See Section 6 for our contributions.

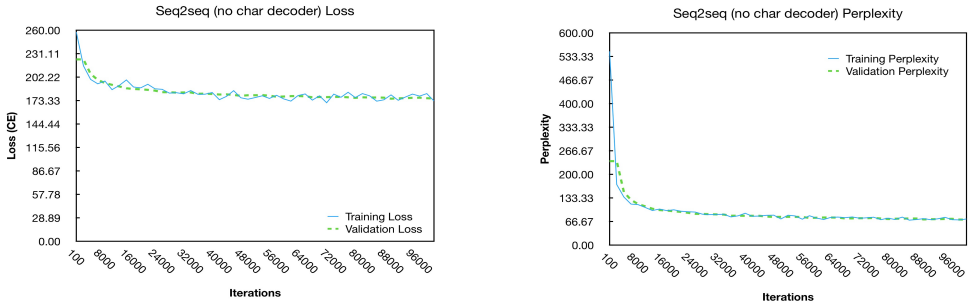

Figure 3: Bi-LSTM-style training curves. Model achieves perplexity of approx. 90. Model bias is high, high loss is maintained throughout the entire training although variance is low.

4.2 Hyperparameter Search

For biLSTM seq2seq baseline hyperparameter tuning and discovered that, as many other translation tasks, **0.001** was relatively performant, with a embedding size of **256** and hidden size of **512**. Figure 3 shows our final training curves for our baseline, with approx. 175 loss and approx. 70 perplexity.

Through hyperparameter tuning of our memory-enhanced transformer, we find that using a learning rate of **1e-3** with **4000** warmup steps with no annealing yields the best training curve. Because we are using **300**-dimensional GloVe embeddings, we find that setting the hidden size of the encoder and decoder to $d_m = \mathbf{300}$ provides easy compatibility with residual connections and eliminates the need for an extra projection layer between the embedding and encoder layers. Because we feel that the LSTM needs to capture more information (globally), we set $d_h = \mathbf{512}$. We also share weights between source / target vocabulary matrices and logit projection weights to help the model train quicker.

We keep most Transformer encoder and decoder parameters the same, except set the hidden size of the position-wise feed-forward layers to **512** and the number of blocks to **3**. It is interesting to note during hyperparameter tuning that increasing model complexity was often very detrimental to performance within small margins. We also use the Adam optimizer with $\beta_1 = \mathbf{0.9}$ and $\beta_2 = \mathbf{0.98}$.

4.3 Fine-Tuning GloVe Embeddings

Initially, we trained word embeddings from scratch with a vocabulary of roughly 25k. However, accuracy never exceeded 15% and it became apparent that we did not have the data to produce representative word embeddings ourselves. We adopted to GloVe embeddings and tuned vocabulary minimum word frequency to produce a vocabulary size of about 25k.

We experimented with both freezing and fine-tuning these embeddings and found that fine-tuning resulted in about a 1% gain after a dozen epochs. Because GloVe does not provide embeddings for special tokens such as `<blank>`, `<unk>`, `<s>`, and `</s>`, fine-tuning the embedding table allowed us to learn those embeddings with more task specfic information.

4.4 Thresholding `<UNK>` Tokens

When initially evaluating our model qualitatively through an interactive mode, we were surprised to find that the model had not really learned to generate any coherent text, despite reaching 25% accuracy in training and validation (it was only generating 'the', 'in', 'world', 'person', and '<unk>' tokens, repeatedly). To fix this problem, we employ an `<unk>` proportion tolerance threshold of 5% in our preprocessing pipeline in order to avoid training on instances for which we do not have at least 95% word embedding coverage.

4.5 Omitting Session-Level Layernorm and Residual

Because the Transformer banks so heavily on layer normalization and residual connections, we decided to see what would happen if we applied layer normalization and residuals to our tensors before they were passed into the decoder (after being weighted by global attention) to increase

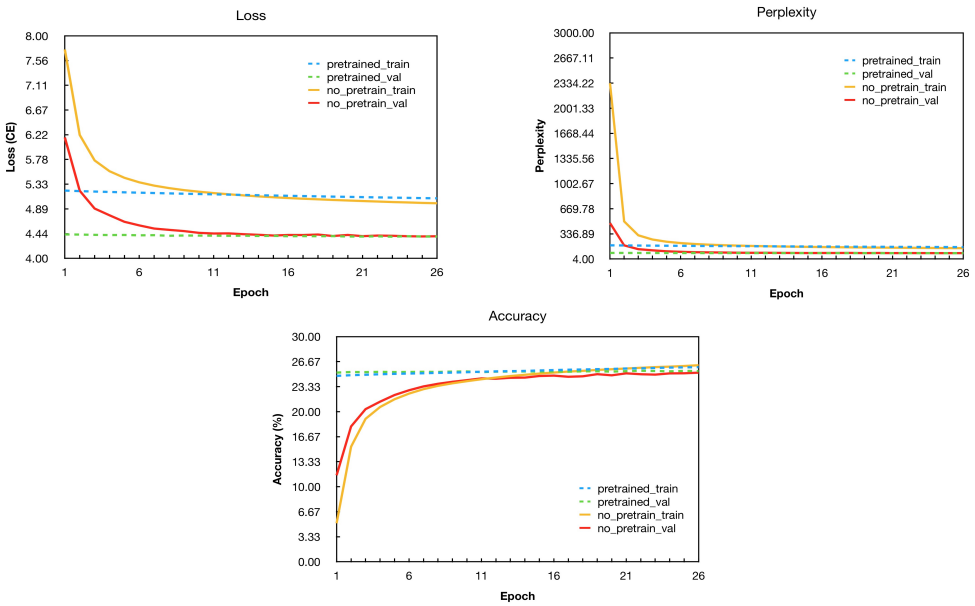

Figure 4: Memory-enhanced transformer argumentative objective training curves. Maximum accuracy is 26%, with lowest perplexity around 80. Pretraining yields immediate improvements in accuracy, loss, and perplexity in early epochs.

performance. This additional operation ended up being detrimental to both training and validation metrics.

We figured that because we were only re-weighting the encoder outputs with a softmaxed attention distribution, the layer normalization and residual was only undoing the effects of our attention mechanism. We think that the positional signals would still be carried through our global memory layer since we are only re-weighting the encoder outputs, and subsequently the positional encodings.

4.6 Pretraining with Denoised Autoencoder and Paired Prediction

The general trend in NLP is to transfer learning between models and objectives. We pretrain in two phases, the trained weights of which help achieve convergence much quicker on the actual task.

1. Denoised Autoencoder. As discussed in lecture, it is important to have an encoder that accurately encodes the input sequence. In this pretraining phase, we train our entire model to predict its input sequence without freezing any layers. After 10 epochs, the model is able to predict its input sequence with 96% accuracy.

2. Paired Prediction. We proceed with the trained weights from the previous pretraining phase. From the denoised autoencoder training, the model learns good encodings of its input, but has not learned to decode its inputs accurately. Thus, in this phase, we still do not rely on session-level dependence, but rather disassemble each discussion into a series of training examples (one per time step) so that the model is trained to predict source $\rightarrow$ target per example. This effectively helps the model to learn how to decode (and encode) per response. The word accuracy rate by the end of 20 epochs is around 25%.

3. Full Training. Taking the trained weights from the last pretraining phase, we then run full training on our dataset. We note that the model starts 1-2% below its convergence value and achieves 25% accuracy in half the number of epochs that it normally would. However, this pretraining only slightly boosted the overall performance of the model at convergence, leading us to believe that the nature of our dataset is not conducive to fully expressive argument generation, despite having some qualitatively appropriate responses to input queries.

Table 2: biLSTM baseline outputs

Query	Output
I think the bible is real and it should be respected	i 'm not sure that you are not a christian. i do n't think it is a matter of the bible . i do n't think it is a matter of a person .
Women's rights and suffrage should be upheld under the law	i do n't know what i said , but i do n't know what you are talking about .

Table 3: Memory-enhanced Transformer outputs

Query	Output
I believe in God	i'm not sure what you mean by "god" .
Well then , have you heard of the Bible ?	i do n't think it 's a good idea .
I think that gun control has the potential to solve a lot of the problems with school shootings .	i think that 's a good thing
I think that abortion should be illegal because its effectively murder	i think that 's a good point . i think that if you are a christian , you are a christian

5 Analysis

5.1 Outputs

Tables 2 and 3 show the outputs of our biLSTM seq2seq baseline and memory-enhanced Transformer, respectively. After pruning <UNK> tokens, we are able to prevent our model from repeatedly generating such tokens and leading to nonsensical output (not shown).

In general, our biLSTM baseline prefers long responses that approximately mimic each other after each '.' token. This is likely because the decoder LSTM is losing information from earlier generation iterations. This leads to the decoder being inclined to generate more words instead of ending the sequence. Additionally, though the biLSTM is able to make attacking responses or unsure responses, many of the responses within the same output contradict each other or are not entirely relevant.

In contrast, our memory-enhanced Transformer model does not output excessively long responses and also provides relevant responses to our queries. However, these responses can be mostly characterized as assertions such as "i think that's a good thing" or declarations of "i don't know." Further studies are needed to visualize the attention within the Transformer networks to note the attended positions within the input sequences.

5.1.1 Common Occurrences

In other generations of our model's output (not listed in Table 3), we find that the model tends to align its responses with a subset of common topics and will try to apply argumentative responses from queries to these topic subsets. Topics such as "Christianity" and "persons" are highly debated topics within our dataset. This answer morphing can be related to the "abortion" example, where being a "Christian" was brought up in the response. This is still a valid argumentative response, as "abortion" and "Christianity" are typically paired together in the news media.

5.2 Accuracy as an Imperfect Metric

When the model is trained on the entire dataset, our training and validation accuracies generally do not reach exceed 25%. For natural language generation tasks, accuracy is not the best evaluation metric as it is insensitive to synonyms and semantically equivalent re-phrasings, in part justifying our low accuracy scores. It is also a harsh metric, only awarding points to exact matches in the gold sequence. Therefore, we also look to human evaluation on sample discussions in order to assess our model's performance.

5.3 Denoised Autoencoder Objective

After adding our denoising objective as a pretraining task, our memory-enhanced Tranformer is immediately able to achieve better training and validation performance. This boost in initil performance is significant due to the finicky nature of training Transformer-style architectures. In line with previous findings in [9] and other pretraining objectives, we save a significant amount of training time while immediately getting acceptable accuracies, as seen in Figure 4. This can be attributed to the ability of the denoising objective to align the argumentation encodings to ensure that the argumentation information are actually encoding the text in a way suitable for natural language understanding.

6 Conclusion

From our qualitative results, we conclude that our dataset is sub-optimal for generating sophisticated argumentative language models, and therefore meaningful argumentative discourse. However, given the limited scope of the data given to our model, we were able to confirm that the use of self-attention in generating individual responses is qualitatively and quantitatively better than using traditional RNN-style seq2seq architectures. This is seen in the appropriateness of the responses as outputted by the transformer model.

We also find that our methods of pretraining - denoising the autoencoder and predicting source-target pairs - do well to help the model learn the task step by step. We believe that it is not only helpful, but almost necessary to train the model to learn good encodings of inputs before training it to decode from those encodings. This allows the model to start actual training with a base understanding of task-specific language.

6.1 Future Work

Because our dataset is the main bottleneck in training our model, less primitive argumentation datasets may help our model learn more expressive argument generation.

With the current success of increased model complexity combined with obscene amounts of unsupervised data (and computational power), we feel that the expressive potential of our model could be vastly enhanced by some massive language modeling pretraining, similar to techniques in [9], [12]. Because learning a good base language model is crucial to success most NLP tasks, we feel that it will greatly contribute to the quality of outputs in ours as well. With more data, we can potentially increase model complexity to enhance expressivity. The access to more data will also allow us to train our own embeddings from scratch, which will inevitably be superior to any pre-neural word embeddings. Since embeddings generated from [9], [10] are hidden states from language models, trained embeddings from scratch and passed through the encoder would be an equivalent substitute for contextual word embeddings.

In summary, future avenues of reserach would investigate what the model could learn with (1) access to more structured data for language modeling and (2) better argumentative datasets coupled with greater model complexity.

7 Contributions

We have only borrowed code for Transformer encoder and decoder forward pass. Other than that, we have written complete preprocessing, encoder-session-decoder interweaving with attention, denoised autoencoder / paired prediction pretraining, interactive inference, and GloVe loading / training.

Ademi Adeniji: Preprocessing with `<UNK>` thresholding; GloVe embeddings; researched recent work in argument mining techniques.

Vincent Liu: Session level LSTM module with softmaxed attention; interactive inference; pretraining with denoised autoencoder and paired predictions.

Nathaniel Lee: GPU configuration; training data metrics; biLSTM seq2seq baseline.

Mentors: Annie Hu and Weini Yu
Github link: `https://github.com/vliu15/transformer-rnn-pytorch`

References

[1] P. F. Brown, V. J. D. Pietra, R. L. Mercer, and J. C. Lai, "An Estimate of an Upper Bound for the Entropy of English," *Computational Linguistics*, vol. 10598, no. 1, pp. 31–40, 1992, ISSN: 0891-2017. [Online]. Available: http://acl.ldc.upenn.edu/J/J92/J92-1002.pdf.

[2] M. Walker, J. F. Tree, P. Anand, R. Abbott, and J. King, "A corpus for research on deliberation and debate," in *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, N. C. (Chair), K. Choukri, T. Declerck, M. U. Doğan, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, Eds., Istanbul, Turkey: European Language Resources Association (ELRA), May 2012, ISBN: 978-2-9517408-7-7.

[3] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," pp. 1532–1543, 2015, ISSN: 10495258. DOI: 10.3115/v1/d14-1162. arXiv: 1504.06654.

[4] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas: Association for Computational Linguistics, 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264. [Online]. Available: http://aclweb.org/anthology/D16-1264.

[5] A. Gatt and E. Krahmer, "Survey of the state of the art in natural language generation: Core tasks, applications and evaluation," *CoRR*, vol. abs/1703.09902, 2017. arXiv: 1703.09902. [Online]. Available: http://arxiv.org/abs/1703.09902.

[6] G. Rakshit, K. K. Bowden, L. Reed, A. Misra, and M. A. Walker, "Debbie, the debate bot of the future," *CoRR*, vol. abs/1709.03167, 2017. arXiv: 1709.03167. [Online]. Available: http://arxiv.org/abs/1709.03167.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," no. Nips, 2017, ISSN: 1469-8714. DOI: 10.1017/S0952523813000308. arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[8] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," *CoRR*, vol. abs/1803.04967, 2018. arXiv: 1803.04967. [Online]. Available: http://arxiv.org/abs/1803.04967.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2018, ISSN: 0140-525X. DOI: arXiv:1811.03600v2. arXiv: 1810.04805. [Online]. Available: http://arxiv.org/abs/1810.04805.

[10] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *Naacl*, pp. 2227–2237, Feb. 2018, ISSN: 9781941643327. DOI: 10.18653/v1/N18-1202. arXiv: 1802.05365. [Online]. Available: http://arxiv.org/abs/1802.05365.

[11] D. Thu Le, C.-T. Nguyen, and K. Anh Nguyen, "Dave the debater: a retrieval-based and generative argumentative dialogue agent," pp. 121–130, 2018.

[12] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," *OpenAI*, 2019. [Online]. Available: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.