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

Ademi Adeniji
Stanford University
ademi@stanford.edu

Abstract

Argument mining, a growing field in natural language generation, includes the automatic identification and generation of argumentative structures within conversation. We experiment with various methods for creating a dialogue agent that can engage in argumentative discourse. We designed a sequence-to-sequence model which applies self-attention to encode and decode responses and maintains a conversation-level memory of discussion history with an LSTM. Our results demonstrate that the model is able to engage in rudimentary discourse, proficiently developing a stance on topics, appraising the validity of contentions, and at times remarking on the ideology underlying its interlocutor's argumentation.

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.

Argument mining is a subset of text generation, but with the additional constraint of responding within the context of a dialogue. 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.

We propose an architecture that uses a Transformer [4] to encode and decode argumentative posts. To invoke discussion context, the encoded sequence is passed through a global LSTM, the output of which attends to the encoded input.

2 Related Work

We describe the two major approaches to natural language generation: retrieval-based systems and generative models.

2.1 Retrieval Systems

The retrieval-based model proposed in [5] 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 [3] 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, any retrieval-based approach is limited in that it can only generate responses present in its database.

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

Table 1: Perplexity on testing sets with sampling and re-ranking described in Section 3.1

2.2 Generative Systems

Generative models eliminate the need for prepared responses or expensive similarity searches and allow the system to generate new responses. The current state-of-the-art generative model used in [5] uses a hierarchical recurrent neural network, encoding and decoding at one level and updating a conversation-level memory 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.

3 Approach

As previously mentioned, our model leverages the Transformer [4] architecture and global attention to generate relevant responses efficiently. At a high level, 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 refer mainly to [5] for metrics. Specifically, we aim to compare against their perplexity numbers to gauge our model's performance. Additionally, the authors provide an online interactive chatbot, which we can use against our model for qualitative comparison. See Table 1 for perplexities reported in the paper. "Sampling" refers to sampling words from the output distribution and "reranking" refers to reranking the samples based on MMI score; both are described in greater detail in [5].

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 $p^{(i+1)}$.

3.3 Model Architecture

We feed mini-batches of our 11,800 discussion instances into our model, whose forward pass for one example is as follows. Due to 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 [4].

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

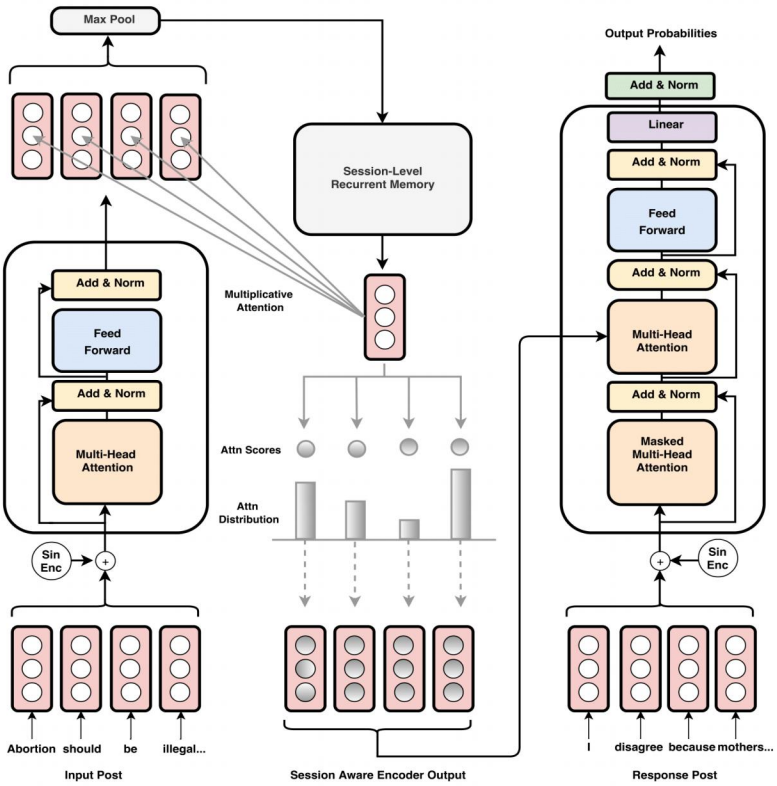


Figure 1: **Transformer Model Architecture w/ LSTM**. We borrow the transformer architecture and use an LSTM between the encoder/decoder to encode session level memory.

`Forward pass:`
`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:

$$\texttt{EmbeddingLayer:}\ \mathbb{R}^n \mapsto \mathbb{R}^{n \times d_m}$$

2. Encode the embedded sequence with the transformer encoder with multi-headed self-attention and position-wise feed-forward layers:

$$\texttt{EncoderLayer:}\ \mathbb{R}^{n \times d_m} \mapsto \mathbb{R}^{n \times d_m}$$

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:

$$\texttt{Decode:}\ \mathbb{R}^{n \times d_m} \mapsto \mathbb{R}^{n}$$

6. Accumulate output probabilities for cross-entropy loss and compute accuracy and perplexity.

3.4 Perplexity and Accuracy

As used in [5], we will adopt the following perplexity metric from [1], $PPL = 2^{-\sum_y \log P(y)}$, where $P(y)$ is the MMI probability. We also calculate word accuracy rate as an indicator of learning during training and validation.

4 Experiments and Results

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.

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

Table 2: *Transformer with session memory sample argument generation*

4.1 Hyperparameter Search

Through log scale exploration we find that using a learning rate of 0.05 with 4000 warmup steps and linear 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 $d_m = 300$ 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 $d_h = 512$. We also share weights between src / tgt vocabulary matrices and logit projection weights.

4.2 Word Embeddings

Initially, we were training 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. We thus turned to GloVe embeddings and tuned vocabulary minimum word frequency to produce a vocabulary size of about 16k.

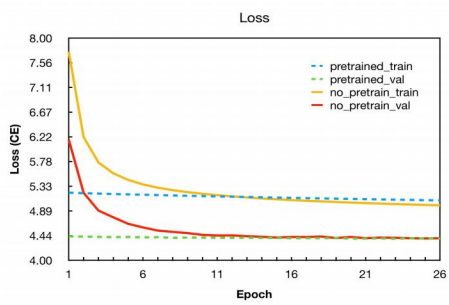
We experimented with both freezing and fine-tuning these embeddings and found that fine-tuning resulted in about a 1% gain after a dozen epochs. Because GloVe does not provide embeddings for special tokens such as `<blank>`, `<unk>`, `<s>`, and `</s>`, fine-tuning the embedding table will allow us to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

Epoch:	Perplexity	Accuracy	Loss
0	205.88	17.30%	5.33
4	84.21	24.83%	4.43
8	72.10	26.35%	4.28
12	65.65	27.36%	4.18
16	62.40	28.00%	4.13

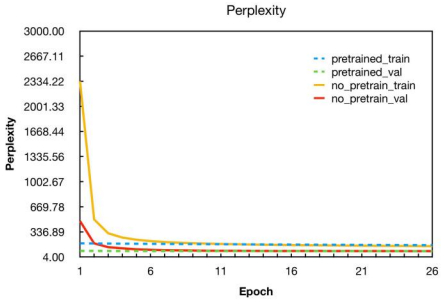
Table 3: *Validation metrics with tuned parameters.*

4.3 Accuracy, Loss and Perplexity

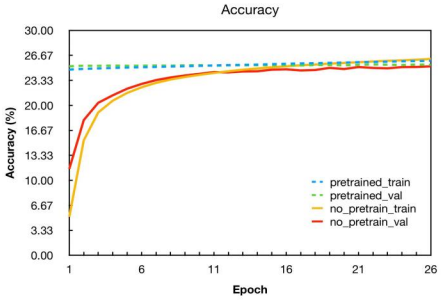
When the model is trained on the entire dataset, our training and validation accuracies reach about 30%. For natural language generation tasks, accuracy is not a perfect evaluation metric as it is insensitive to synonyms and semantically equivalent re-phrasings, in part justifying our low accuracy scores. We look primarily to human evaluation on sample discussions in order to assess our model's performance.



(a) Transformer w/ memory loss



(b) Transformer w/ memory perplexity



(c) Transformer w/ memory accuracy

5 Conclusion

From our qualitative results, we conclude that our dataset is ill-suited for generating more sophisticated language models as is typical of advanced argumentative discourse. Our successfully optimized training metrics may suggest that our cross entropy training objective is overly simplistic for more complex generation tasks. A more involved theoretical formulation of training loss could yield qualitative translation improvements. However, we were impressed by the model's ability to infer the underlying ideology of the human debater on the basis of its arguments. For instance, the model remarked that the originator of a pro-life argument made such a contention on the basis of a Christian ideology. Furthermore, the dialogue agent was surprisingly capable of producing relevant responses, either establishing a resolute position on the proposed topic, or offering a concise appraisal of its validity.

6 Future Work

Less primitive argumentation datasets may help our model learn a more expressive language model. For example, training on congressional policy deliberations may provide more instances of syntactic sophistication for the model to learn from. Fine-tuning on pretrained contextual embeddings such as BERT may serve to capture word relationships more precisely and thus produce better text generation. Although GloVe embeddings are a notable advancement from our baseline's Word2Vec embeddings, BERT embeddings have been shown to yield significant improvements in language-based tasks. Lastly, less commonly used forms of attention such as coattention, content-base attention, or location-base attention may allow for more informative signals to pass through to the decoder. Such may be conducive to generating responses that attend more precisely to the most critical information encoded in the argument query.

Mentors: Annie Hu and Weini Yu

Github link: `https://github.com/vliu15/transformer-rnn-pytorch`

Sharing Project with CS 224N

7 Contributions

Ademi Adeniji - Designed and implemented model architecture and preprocessing. Ran hyperparameter search and integrated GloVe embeddings.
Nate Lee - Implemented preprocessing pipeline and worked on model design.
Vincent Liu - Implemented model architecture and unk proportion thresholding.

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] 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`.

[4] 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`.

[5] 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.