CS230

# Intelligent Personal Assistant Dialog Generation using Paraphrasing

**Weiwei Zhang**
Department of Computer Science
Stanford University
wwzhang@stanford.edu

## Abstract

Inspired to generate personal assistant dialogues using paraphrasing, we explore several deep learning architectures for the generic paraphrasing task. We build off on the original seq2seq + LSTM model with bag of words (BOW) sampling proposed in Fu et. al., 2019[1], and experiment with variations of this model using transformer architecture. We have found that the original paper's innovative idea of using BOW sampling from source sentences to generate word embeddings to augment decoding process may not transfer well to transformer architecture, and we need to caution that overly complex transformer architecture can discourage model performances. We have outlined future steps on how we can improve our transformer models as well.

## 1 Introduction

In current-day productionalized personal assistants, all assistant dialogues are hard-coded rather than intelligently generated. Lack of diverse dialogues can make assistants sound rigid and unnatural, and it's tedious for development to maintain a large number of hard-coded dialogues. Ideally we'd have a black box system that takes in existing dialog context and generates a variety of responses that are appropriate, correct and natural. However that's difficult to achieve at the moment.

Therefore, we will explore the possibility to humanize assistant dialogues by intelligently generating variety of good-quality paraphrases using deep learning techniques. We envision a system where we would generate a sample response through dialog generation or just human hard-coding, then paraphrase on the sample response to achieve more diverse, natural dialogs.

Thus we transform this problem into a generic paraphrasing problem, so that we can leverage a variety of mature paraphrasing datasets and techniques.

## 2 Related work

Traditionally, research like I. Perikos et al., 2016[5] and M. Virkar et al., 2019[6] try to address paraphrase generation with more linguistics rule-based ideas. These approaches have more human-interpretable intermediate training steps, but may lose the benefit of end-to-end training in deep learning techniques. Recently many research employ deep learning techniques such as seq2seq model structure with variations of RNN (e.g. Li et al., 2018[3], Gao et al., 2020[4], Zhang et al., 2019[7]). In the past 3 years, transformer architecture and attention mechanism have achieved good

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

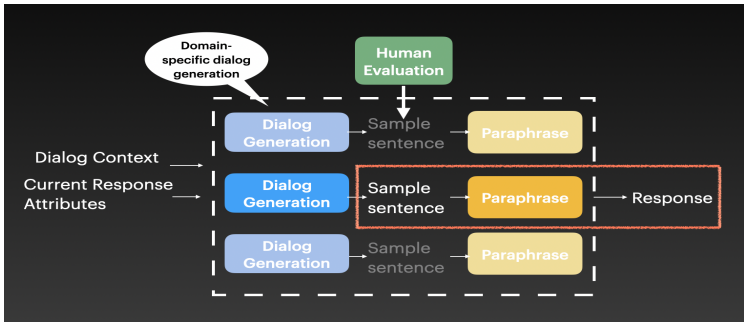


Figure 1: We propose that the model to generate and then paraphrase assistant dialogues can be trained per-domain (e.g. web answers, restaurants, hotels) to improve performance. However since it's hard to curate multiple well-formatted domain data sources, we will only explore generic paraphrase generation highlighted in the red box.

results (e.g. Goyal et. al., 2020[8]).

In this project, we mainly reference ideas from Fu et al., 2019[1] (see Appendix[1] for diagram). Fu's work proposed a seq2seq LSTM network with attention mechanism in decoder. Similar to traditional linguistic rule-based research, this research has a clear content planning and realization phase divide which increases the interpretability of the model. During content planning it creates a bag of words (BOW) sample memory by sampling out of the k-nearest neighbors space of source sentence words within the whole corpus vocabulary. The sampling memory is then used both in decoder RNN cell as part of the decoder input hidden state, and also in the additional attention layers during decoding to influence the output sentences towards the sampled bag of words. Given a good BOW sampling memory, this method can increase the lexical diversity of the output sentences while retaining high paraphrase accuracy.

## 3 Dataset and Features

We use Quora Question Pairs[1] as our corpus. Quora Question Pairs contains 404,290 sentence pairs, labeled paraphrase or not (see Appendix[2] for examples). The percentage of paraphrases in the corpus is 36.92% (assuming no mislabeling). After basic data cleaning such as removing invalid data points (e.g. missing one sentence in the pair), we sampled 50k pairs of paraphrase sentences as training examples and 20k pairs as evaluation examples, so as to be comparable to original paper's dataset size. The corpus has a vocabulary size of 37,117 if we only look at words with occurrence >= 5.

The sentence length at 80 percentile is 16 words per sentence, at 90 percentile is 21, and at 95 percentile is 26. These are good indicators to how long output sentences might be (around 20 words). Similar to real-life personal assistant dialogues, the corpus sentences are short-or-medium lengths mostly, so we would prefer using smaller n-gram metrics (e.g. BLEU-2). In addition, we also generate Word2Vec embeddings (of dim = 300 after tuning) for input sentences to better capture word-level features using FastText[2], in contrast to original paper's n-gram approach, where you build an index array out of corpus vocabulary dictionary.

## 4 Methods

To reiterate, our goal is to generate good-quality paraphrases. We want to leverage the bag of words (BOW) sampling technique proven in Fu et al. 2019[1] to increase relevance and accuracy of words in outputs, and also use a transformer architecture which is a recent SOTA architecture and should ideally retain cross-word correlations within a sentence better.

[1] https://www.kaggle.com/c/quora-question-pairs
[2] https://github.com/facebookresearch/fastText/

We keep the encoder decoder structure in Fu's work, but replace each LSTM cell with a transformer block instead. Here we use Texar[3] library's TransformerEncoder block implementation as is, which consists of a standard self-attention layer and feed forward neural network layer[11]. The feed forward neural network layer consists of two Dense layers that can optionally have residual connections with other layers. We also use Texar's TransformerDecoder implementation, which contains a self-attention layer and a feed forward neural network layer similar to encoder. It also has a standard encoder decoder attention layer, and we make modifications by adding an additional encoder decoder attention layer to use only BOW sampling memory. This is similar to how BOW sampling memory was used in the original seq2seq network's decoder attention layer.

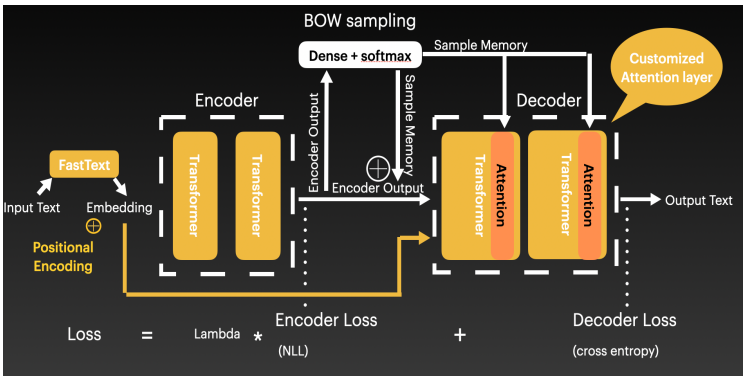


Figure 2: Diagram of new model. Orange parts highlight changes from the original model.

In addition, we explore a variation of this model using pretrained Word2Vec embeddings instead of n-gram input embeddings. Note that there are more complex, innovative ways to generate input embeddings (e.g. Socher et al., 2011[2], Patro., 2018[13]), where sentence-level information is better captured in the word embeddings. Since transformer's self-attention and positional embeddings mechanism would already help retain sentence structure level signals, we won't explore those ideas.

The loss we use is the similar to the original model, where we compute an aggregated loss of encoder output's negative log likelihood loss and decoder output's cross entropy loss:

$$loss = \lambda * NLL_{enc}(output_{enc}, target) + CrossEntropy_{dec}(output_{dec}, target) \quad (1)$$

As for evaluation, we use BLEU(1-4grams) and ROGUE(1-2) scores to be consistent with original model. We use BLEU-2 because sentences in training set are not very long, and therefore smaller N-gram metrics would be more suitable.

Note that we will not discuss interpretability which was a big topic in the original paper, mainly because we don't change how BOW is sampled and therefore don't change the content planning and realization process.

Our code is on github: https://github.com/weiwzhang/dialog_paraphrase.

## 5 Experiments and Discussions

We re-ran baseline model and different variations of customized models on the same AWS instance with all of training data to get comparable results. Here is a table of exemplar models and metrics. Similar to original paper, we use BLEU-2 as the single optimizing metric. Our current best transformer model has not outperformed the baseline, although the gap (0.03) is not huge and can possibly be mended through more tuning.

We try to condition on the same basic hyperparameters to make results comparable. Each of the model in the table has 2 respective blocks in encoder and 2 blocks in decoder. They all use the same batch size of 5 (therefore 10000 batches in training), the same flat learning rate of 1e-3, and the same Adam optimizer with beta1 = 0.9, beta2 = 0.997, and epsilon = 1e-9.

[3]https://github.com/asyml/texar

Table 1: Exemplar Model Runs (B = BLEU, R = ROGUE)

| Model Name | B-1 | **B-2** | B-3 | B-4 | R-1 | R-2 |
|---|---|---|---|---|---|---|
| 1) LSTM + Attn + BOW | 0.5476 | **0.4048** | 0.3111 | 0.2470 | 0.5693 | 0.3245 |
| 2) Transformer + BOW | 0.4529 | 0.3095 | 0.2190 | 0.1492 | 0.4718 | 0.2395 |
| 3) Transformer + BOW + W2V | 0.4782 | 0.3327 | 0.2385 | 0.1649 | 0.5050 | 0.2620 |
| 4) Transformer + BOW + W2V + beam | 0.4706 | 0.3370 | 0.2476 | 0.1766 | 0.5277 | 0.2833 |
| 5) Transformer + BOW + W2V + residual | 0.4944 | 0.3444 | 0.2467 | 0.1706 | 0.5234 | 0.2716 |
| 6) Transformer + BOW + W2V + dim=300 | 0.5181 | 0.3691 | 0.2701 | 0.1944 | 0.5456 | 0.2947 |
| 7) Transformer + W2V + dim=300 | 0.5102 | **0.3701** | 0.2762 | 0.2011 | 0.5571 | 0.3065 |

### 5.1 Analysis on Model Structure

Model 1), 6) and 7) can be compared to analyze model architecture. Model 1) is the original model; Model 6) uses BOW sampling and 7) doesn't. Comparing Model 6) and 7) we can see that the transformer model performs slightly better without BOW sampling, although the difference is small (0.001). This probably indicates that after adding in an extra attention layer using BOW memory in each decoder block, the transformer gets overly complicated. Model 1) performs better than both model 6) and 7. A possible intuitive explanation for this is, in original baseline, BOW sampling memory is passed through an attention layer and then passed into decoder cell as previous state. If sampling does a good job picking out words that represent input sentence meanings, having those words in decoder cell memory would influence decoding in a positive way. In transformer, BOW sampling memory is passed in as an extra encoder decoder attention layer. Encoder decoder attention layers help capture input sequence structure level information in addition to word level information, so it would be able to help capture relationships between words in BOW sampling memory. However it's unclear if this information helps a normal encoder decoder attention layer or the whole decoder block to generate good outputs.

### 5.2 Analysis on Training Process

Model 2), 3), 4), 5) and 6) can be compared to analyze benefits of different training techniques. All of them use the same transformer + BOW sampling architecture. However, Model 2) doesn't use Word2Vec input embeddings comparing to the rest, and Model 4), 5) and 6) use different training techniques comparing to Model 3). We can see that using beam search technique (Model 4), or using residual connections + dropout (Model 5) + less hidden states (Model 6) all improve results. Here we use a beam width = 5. In each transformer block, the residual connections between the 2-layer dense network has dropout rate = 0.3. We have also tried changing hidden states from 500 to 300 units, which has a similar effect to increasing residual connections. Overall the techniques that reduce hidden connections and therefore complexity of the network (Model 6 comparing to Model 5) seem to improve the target metric more visibly, indicating that transformer BOW model might have too many parameters. Also using better input embeddings (Model 3 comparing to Model 2) provide obvious improvement to performance as well.

### 5.3 Precision and Recall

Here we graph the precision and recall graph of baseline and all transformer model outputs. We can see that models with both higher precisions and recalls (baseline, Transformer + BOW + W2V + dim=300, Transformer + W2V + dim=300) also perform better according to BLEU-2 metric, which is as expected. It's interesting to see for example Transformer + BOW + W2V + dim=300 and Transformer + BOW have very similar precisions, and yet their recalls are dramatically different. If we figure out how to move recall metric in this case, we can probably improve Transformer + BOW performance.

### 5.4 Generated Output Examples

Here's an example of input, generated output, and the BOW sampling process. We can see how input and decoder output retain many similarities, but output is able to have variations in syntax as well as vocabulary. The BOW sampling process shows each input word's neighbors and respective scores. The neighbor words are then sampled from this probability distribution of scores to generate the sampling memory which is then used to influence decoder outputs. We can see that the BOW

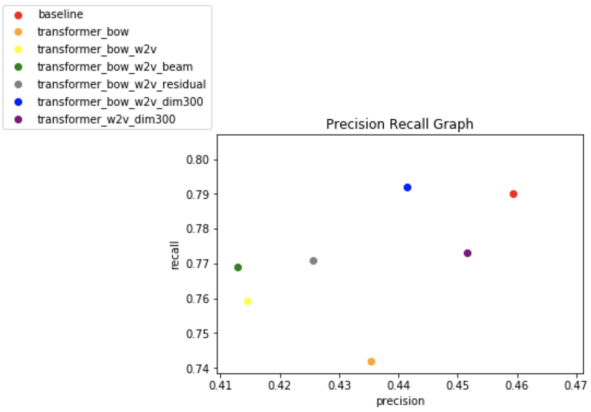


Figure 3: Precision vs. Recall for all models

```
inputs:
    _GOO how does a long distance relationship work ? _EOS
input neighbors:
_GOO: ?(0.9990), ?(0.5280), ?(0.9997),
how: ?(0.9997), long(0.2246), long(0.3734),
does: distance(0.7533), help(0.1384), long(0.5552),
a: distance(0.9169), long(0.0938), long(0.3685),
long: long(0.5463), long(0.9985), long(0.9998),
distance: distance(0.9968), relationships(0.9985), long(0.9999),
relationship: relationship(1.0000), relationships(0.9999), relationship(0.9999),
work: distance(0.9958), work(0.5973), work(0.8818),
?: ?(0.9025), long(0.1884), work(0.3630),
_EOS: _GOO(0.0000), _GOO(0.0000), _GOO(0.0000),
enc_output_bow:
    long ? distance work relationship relationships works survive help cope
enc_sampled_memory:
    long ? distance work relationship relationships works survive help cope
dec_outputs:
    how do long distance relationships work ? _EOS
references:
    do long distance relationships work
```

Figure 4: Example Input, Generated Output, and BOW Sampling

sampling space has varying qualities for each word. Improving this quality can probably lead to better model performances as well.

## 6 Conclusion and Future Work

In this project we have experimented paraphrase generation using a baseline LSTM + attention mechanism network, as well as different variations of transformer. We explored adding a novel BOW sampling memory as additional encoder decoder attention layer in transformer to improve results, as well as using other conventional techniques such as using FastText embeddings, residual connections and dropouts, etc. We have found that the BOW sampling technique in baseline doesn't transfer easily to transformer models, and new transformer models seem overly complex and needs more fine tuning and trimming down to perform better.

Therefore we propose following future steps: 1. modify transformer structure to use BOW as part of the input to existing single encoder decoder attention layer, not in an additional new attention layer, so as to simplify the architecture. 2. further tuning by adding more residual connections and dropouts 3. training loss, precision and recall show more fluctuation in the transformer models comparing to baseline (see Appendix[3]). It's worth analyzing whether we can smoothen the curves and if that'll lead to better model performances.

## References

[1] Fu, Yao & Feng, Yansong & Cunningham, John. (2019). Paraphrase Generation with Latent Bag of Words.

[2] Socher, Richard & Huang, Eric & Pennington, Jeffrey & Ng, Andrew & Manning, Christoper. (2011). Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. *Advances in Neural Information Processing Systems. 24.*

[3] Li, Zichao & Jiang, Xin & Shang, Lifeng & Li, Hang (2018) Paraphrase Generation with Deep Reinforcement Learning

[4] Gao, Silin & Zhang, Yichi & Ou, Zhijian & Yu, Zhou (2020) Paraphrase Augmented Task-Oriented Dialog Generation

[5] I. Perikos & I. Hatzilygeroudis (2016) "A methodology for generating natural language paraphrases," *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA), Chalkidiki, 2016, pp. 1-5.*

[6] M. Virkar & V. Honmane & S. U. Rao (2019) "Humanizing the Chatbot with Semantics based Natural Language Generation," *2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 2019, pp. 891-894.'*

[7] Zhang, Yuan & Baldridge, Jason & He, Luheng (2019). PAWS: Paraphrase Adversaries from Word Scrambling.

[8] Goyal, Tanya & Durrett, Greg & (2020). Neural Syntactic Preordering for Controlled Paraphrase Generation.

[9] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, (2017). Bag of Tricks for Efficient Text Classification.

[10] Zhiting Hu, Haoran Shi, Bowen Tan, Wentao Wang, Zichao Yang, Tiancheng Zhao, Junxian He, Lianhui Qin, Di Wang, Xuezhe Ma, Zhengzhong Liu, Xiaodan Liang, Wanrong Zhu, Devendra Sachan and Eric Xing (2019). Texar: A Modularized, Versatile, and Extensible Toolkit for Text Generation

[11] Ashish Vaswani and Noam Shazeer and Niki Parmar and Jakob Uszkoreit and Llion Jones and Aidan N. Gomez and Lukasz Kaiser and Illia Polosukhin, (2017). Attention Is All You Need.

[12] Patro, Badri N. & Kurmi, Vinod K. & Kumar, Sandeep & Namboodi, Vinay P. (2018). Learning Semantic Sentence Embeddings using Pair-wise Discriminator.

# Appendix

**[1]**

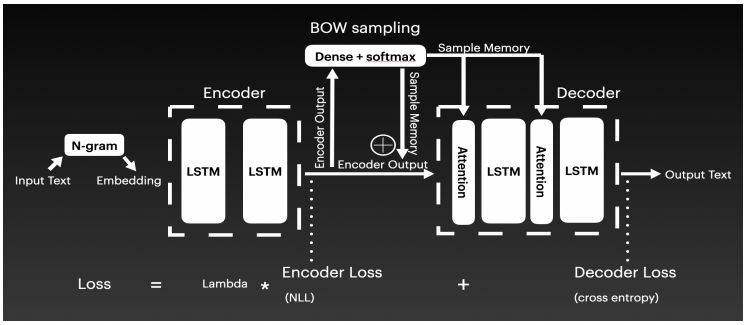


Figure 5: Diagram for original model in Fu et al., 2019

**[2]**

| | id | qid1 | qid2 | question1 | question2 | is_duplicate |
|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 2 | What is the step by step guide to invest in sh... | What is the step by step guide to invest in sh... | 0 |
| **1** | 1 | 3 | 4 | What is the story of Kohinoor (Koh-i-Noor) Dia... | What would happen if the Indian government sto... | 0 |
| **2** | 2 | 5 | 6 | How can I increase the speed of my internet co... | How can Internet speed be increased by hacking... | 0 |
| **3** | 3 | 7 | 8 | Why am I mentally very lonely? How can I solve... | Find the remainder when [math]23^{24}[/math] i... | 0 |

Figure 6: Examples from Quora Question Pair corpus.

**[3]**

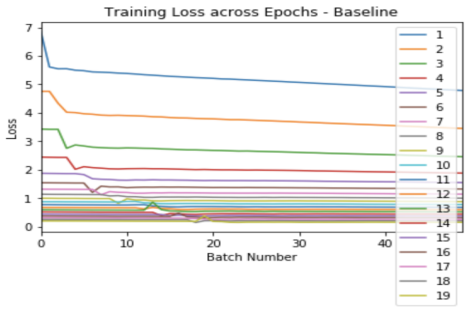


Figure 7: Training loss of baseline model. Graphed only first 50 batches in a total of 1000 batches per epoch for clarify. We can see a steadily decreasing loss with a relatively smooth start of the curves.

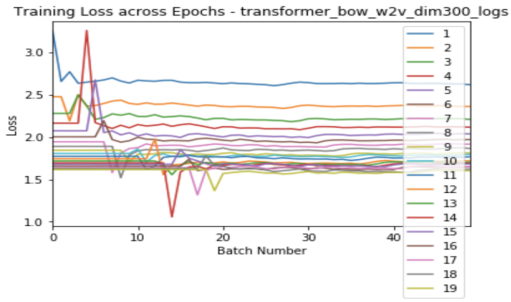


Figure 8: Training loss of transformer model. Graphed only first 50 batches in a total of 1000 batches per epoch for clarify. We can see a lot more fluctuation near the start of training, and such fluctuation seem to impact the loss from decreasing more efficiently throughout training.

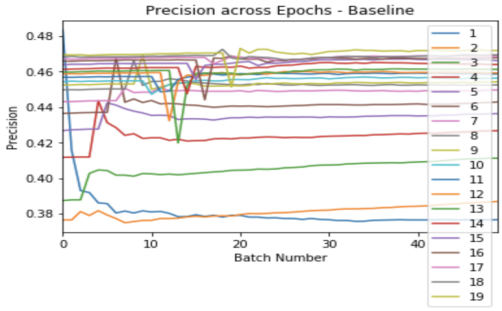


Figure 9: Precision of baseline model. Graphed only first 50 batches in a total of 1000 batches per epoch for clarify. Similar observations as training loss.

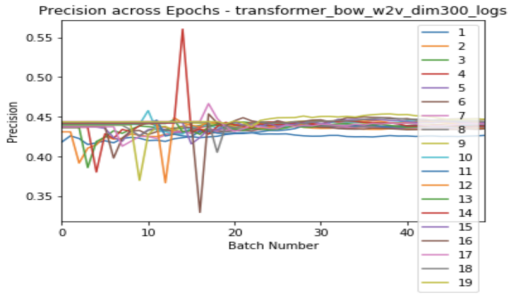


Figure 10: Precision of transformer model. Graphed only first 50 batches in a total of 1000 batches per epoch for clarify. Similar observations as training loss.

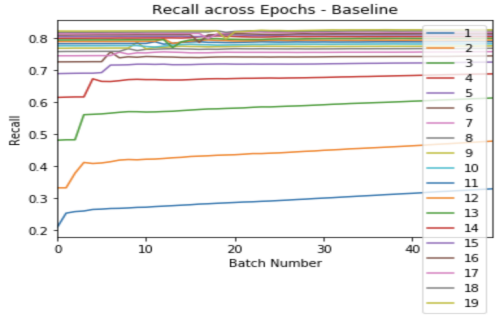


Figure 11: Recall of baseline model. Graphed only first 50 batches in a total of 1000 batches per epoch for clarify. Similar observations as training loss.

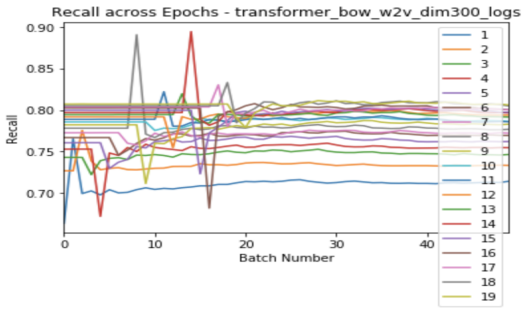


Figure 12: Recall of transformer model. Graphed only first 50 batches in a total of 1000 batches per epoch for clarify. Similar observations as training loss.