A Preliminary Review to Machine Translation

Yan-Yu Chen
Department of Statistics
University of California, Davis
Davis, CA 95616
ynychen@ucdavis.edu

Yinan Cheng
Department of Statistics
University of California, Davis
Davis, CA 95616
yncheng@ucdavis.edu

Ju-Sheng Hong
Department of Statistics
University of California, Davis
Davis, CA 95616
jsdhong@ucdavis.edu

Mingshuo Liu
Department of Statistics
University of California, Davis
Davis, CA 95616
mshliu@ucdavis.edu

Abstract

Machine translation aims to the automatic translation of a sequence of words from a source language to a target language. From the model aspects, encoder-decoder based models was proposed to deal with the sequential data. Among them, transformers have been a wide-spreading architecture in the applications such as natural language processing (NLP). From the word embedding aspects, BERT has achieved a huge success over its competitors. In this project, we review the basic structure of a transformer and BERT and intuitively explain the reason of their accomplishment. To illustrate our algorithm, we translate English to French using machine translation on a dataset used in Zhang et al. [19] and compare the performance with a vanilla transformer model.

1 Introduction and Literature Review

The sequential data is the most commonly data type in the area of NLP. Recurrent neural networks (RNNs, Rumelhart et al. [17]) has been developed for years to tackle this certain problem. Let $\{x_t\}_{t\in\mathbb{T}}$ be the input where, for instance, $\mathbb{T} = \{1, 2, \dots\}$ can be considered as an index set. RNNs, conceptually, model $P(x_t \mid \{x_s\}_{s<t})$ as $P(x_t \mid h_{t-1})$ where the hidden state, h_{t-1}, is a latent variable and one can think of it as a concentration of all words before t. One of the main topic in RNNs is to design the hidden state to maximize the objection function. During the maximization, problems like exploding gradient and vanishing gradients happen while doing backpropagation through long-term dependencies (Bengio et al. [2]). On top of that, we might consider the cases where an early observation plays a crucial role in predicting the future observations or some observations have no effect to the future. To remedy those concerns, Hochreiter and Schmidhuber [7] proposed long short-term memory (LSTM) and Cho et al. [5] proposed gated recurrent unit (GRU) which offered a simplicity over its competitors and was much faster to compute. Conceptually, those models used the "forget gate" and "update gate" make the communication across numerous timestamps possible.

In the area of machine translation, because we have two sequential data, the source language and the target language, we design two RNNs for each of them. Such a modeling is called the encoder-decoder architecture. The workflow can be decomposed into two steps: first, the encoder transforms a sequence of words into a fixed-length context variable; then, the decoder generates the output sequence words by words based on the generated output and the context variable from the encoder. During this process, the same variable provided by the encoder is used in each decoding step. The

Preprint. Under review.

attention layer aims to alleviate this problem. The Nadaraya-Watson kernel regression (Nadaraya [11]) is a simple illustration to machine learning with attention mechanism. The idea is to use a local kernel smoother to estimate the underlying function. Bahdanau attention was proposed in Bahdanau et al. [1] which wrote the context variable as at the decoding step t' as $c_{t'} = \sum_t \alpha(s_{t'-1}, h_t)\, h_t$ where s_t and h_t was the hidden states at step t in the decoder and encoder, respectively, the attention weight α was defined using the softmax function, and t was summed over $\{1, \ldots, T\}$.

Early attention models including certain self-attention models (Cheng et al. [4], Lin et al. [8], Paulus et al. [13]) relied on RNNs for the input representation, which had operations that cannot be parallelized and the maximum path length to be the order of sample size. The transformer model published in Vaswani et al. [18], on the other hand, only depended on the attention mechanism without any convolutional or recurrent layer. Therefore, the computational complexity per layer is minimized and the maximum path length between any two positions of input and output vectors is minimized, which makes it easier to learn long-range dependencies. We will explain the details of the transformer model in Section 3.1.

Before running the models we mentioned previously, an imperative step is to transform the sequence of words into numeric vectors. We will fully describe the process in Section 2.2 but here's a simple example. Suppose we have two words: great and excellent. One could use one-hot word embedding and let great $= (1, 0)$ and excellent $= (0, 1)$ to convert them into numbers. Although it's easy to implement, the problem is that all vectors after one-hot embedding are orthogonal. The words, great and excellent, are obviously synonym. Fail to consider the inner relationship between words could cause catastrophic result in NLP. Using probability model, Mikolov et al. [9] and Mikolov et al. [10] proposed two word2vec models and Pennington et al. [14] proposed GloVe to address the above issue. Based on the structure of transformer, Devlin et al. [6] proposed the BERT model. Using a pretrained transformer encoder, BERT can represent any words based on the context from both direction. We will explain the BERT model in Section 3.2.

The rest of the paper is arranged as followings. In Section 2, we will define the machine translation problem and describe the data preprocessing. In Section 3, we will elucidate the transformer and the BERT models in full details. In Section 4, we will present the result of the analysis and compare the performance with some naive machine translation model. Last but no least, we will conclude this paper in Section 5.

2 Machine Translation

2.1 Problem Definition

Let n be the numbers of sentences in either language. For $i \in \{1, \ldots, n\}$. Let $\{x_{it}\}_{t \in T_i}$ and $\{y_{is}\}_{s \in S_i}$ be the ith sentence in the source corpus and the target corpus where T_i and S_i is the index sets. T_i has the form $\{1, \ldots, |T_i|\}$ where $|T_i|$ is the number of tokens in the ith sentence of the source corpus (similarly for S_i). We make a remark here: the length of the sentences varies not only in corpus but in language. Dealing with the data with non-fixed input length is one of the main challenge in NLP.

In the machine translation problem, our goal is to build a translator that can translate $\{x_i\}$ into $\{y_i\}$ the best. During the training step, the **loss function** is chosen to be the softmax function and applied to calculate the cross-entropy loss for optimization.

2.1.1 Evaluation of the Predicted Sequence

Let $\{\widehat{y}_{is}\}_{s \in \widehat{S}_i}$ be the predicted sequence of the ith sentence in the source language, $\{x_{it}\}_{t \in T_i}$. Papineni et al. [12] proposed BLEU (Bilingual Evaluation Understudy) to evaluate the quality of a translation by giving a higher weight to a longer n-gram precision. Here, n-gram represents for the consecutive n-words in a sentence. BLEU for the ith pair of sentences is defined as

$$\exp\left[\min\left(0, 1 - \frac{S_i}{\widehat{S}_i}\right)\right] \prod_{n=1}^{k} p_n^{1/2^n}$$

where k is the longest n-gram for matching and p_n is the precision of n-gram. For instance, let $\{\widehat{y}_{is}\}_{s \in \widehat{S}_i} = \{A, B, D, E\}$ be a prediction of $\{y_{is}\}_{s \in S_i} = \{A, B, C, D, E\}$. To calculate p_1, since

$\{A\}, \{B\}, \{D\}, \{E\}$ all appear in the target, we have $p_1 = 4/4 = 1$. To calculate p_2, we notice that the 2-grams are $\{A, B\}, \{B, D\}, \{D, E\}$. Because $\{A, B\}$ and $\{D, E\}$ appear in the target, $p_2 = 2/3$. One observation of BLEU is that although predicting short sequence tends to have a higher p_n, the exponential term in BLEU penalizes shorter prediction.

Although BLEU is a great tool at evaluating the quality of machine translation, its non-differentiability makes it hard to be a loss function, as opposed to the softmax function. Ranzato et al. [16] addressed this issue and came up with a solution using the idea of reinforcement learning.

2.2 Data Preprocessing

We describe how to preprocess the data to accommodate the naive one-hot embedding model in this section. The English-French dataset comes from bilingual sentence pairs from the Tatoeba Project. Conceptually, there are a few steps: basic preprocessing, tokenization, creating the dictionary, and representing a sentence using the dictionary. English-French dataset contains 167,130 sequence in both language ($n = 167, 130$). Table 1 contains a few sentences from the dataset:

Table 1: Examples in the Dataset

English	French
We're standing.	Nous sommes debout.
What a bargain!	Quelle bonne affaire!
You have to go.	Tu dois partir.
You seem happy.	Vous semblez heureuses.

The process of training could take few days if the objective is to do the training on the whole dataset even if we use the GPU computing service provided by Google. Therefore, we focus on the first $5,000$ sentences to conduct the analysis. More details on the setting of the parameters can be found in Section 4.

- In the basic preprocessing step, we try to standardize our dataset. To be specific, we replace non-breaking space with space, convert uppercase letters to lowercase ones, and insert space between words and punctuation marks.
- In the tokenization step, the word-level tokenization is engaged, i.e., **go.** becomes **"go"**, **"."**. Figure 1 shows a majority of the text sequences have fewer than 20 tokens. To make sure the inputs have unanimous length, we truncate a sequence by only preserving the first 10 tokens if it is too long, or we append the token <"pad"> until the length of a short sequence reaches 10. In this way, every text sequence will have the same length to be loaded in minibatches of the same shape.
- Next, we replace tokens that appear less than twice by the unknown token ("<unk>") to reduce the vocabulary size. To create the dictionary, we simply refer each token to a unique number. For instance, the first few words in the English dictionary are "1": "Go", "2": "Hi", "3": "Run".
- Lastly, each sentence is presented using the numbering system provided by the dictionary.

In the vanilla transformer model, we use the one-hot word embedding technique on both English and French dataset. However, while using the pretrained BERT model, the input of the encoder (English, in this case) is embedded using the dictionary that construct the BERT model.

3 Methodology

3.1 Transformer

After Vaswani et al. [18] proposed the transformer model in 2017, it has become one of the most significant model in NLP and computer vision. Transformer is one of encoder-decoder structured models as shown in Figure 2 (from Zhang et al. [19]). Though originally proposed for sequence to sequence learning on text data, transformers have been pervasive in a wide range of modern deep learning applications, such as in areas of language, vision, speech, and reinforcement learning.

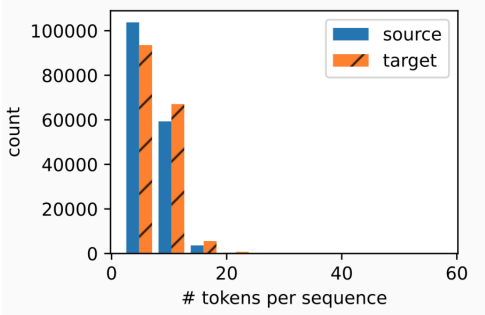

Figure 1: Histogram of the Number of Tokens Per Text Sequence

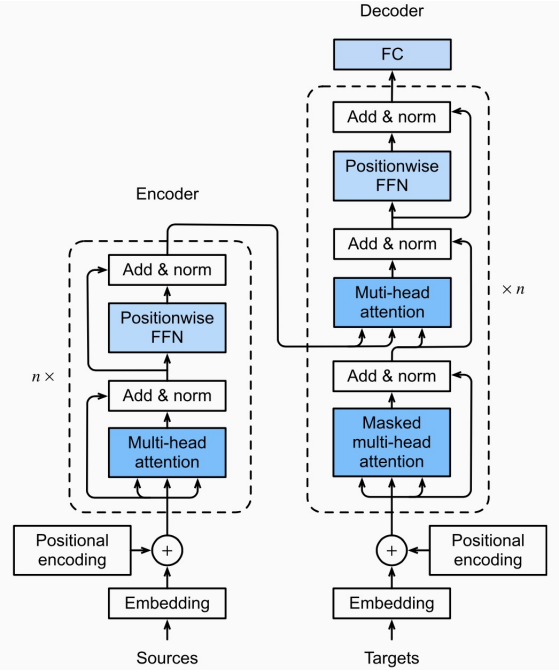

Figure 2: Transformer Architecture

The transformer architecture consists of two main parts, an encoder and a decoder.

- Encoder: the transformer encoder is composed of multiple identical layers and each layer consists of two sublayers. The first one is a multi-head self-attention mechanism and the second one is a positionwise fully connected feed-forward neural network. The first layer takes embedding and positional encoding information of the sources as its input. With residual connections around the two sublayers, the output of each sublayer is required to has the same dimension of its input.

- Decoder: the decoder is also composed of multiple identical layers with residual connections but each layer has three sublayer. Besides the two sublayers in each encoder layer, an additional attention mechanism - encoder-decoder attention is inserted. This sublayer operates multi-head attention to draw information from encoder outputs. There is a masked attention which prevent positions from attending employing outputs of subsequent positions.

Positionwise Feed-Forward Networks (Positionwise FNN) and multi-head attention (Figure 3) are used repeatedly in both encoder and decoder. Residual connection and layer normalization are used after each step of data-processing, and both are key to effective deep architectures.

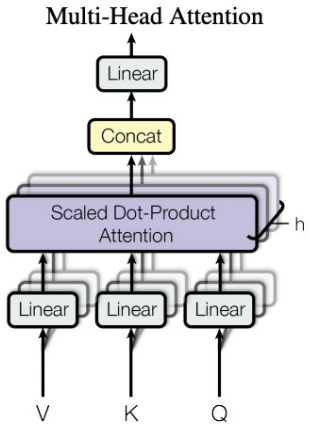

Figure 3: Multi-Head Attention

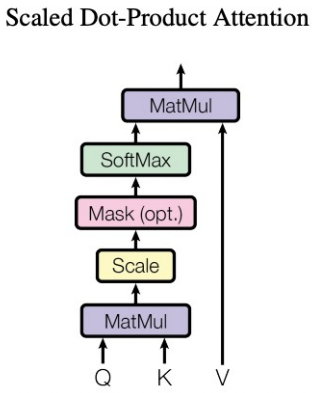

Figure 4: Scaled Dot-Product Attention

3.2 BERT

BERT, bidirectional encoder representations from transformers, was first introduced in Devlin et al. [6]. As an unsupervised learning algorithm, BERT aims to present a token using its neighbourhood. The pretraining comprises the following two tasks: masked language modeling (MLM) and next sentence prediction (NSP).

- Masked Language Model (MLM): BERT randomly masks tokens and uses tokens from the bidirectional context to predict the masked tokens. The non-masked token is ignored. This helps in calculating loss for only those 15% masked words. For those masked words, it will be replaced with
  - a special "<mask>" token with 80% of the chance
  - a random token with 10% of the chance
  - the unchanged label token with 10% of the chance
- Next Sentence Prediction (NSP): BERT predicts whether the following sentence is the next subsequent sentence of the first sentence in the original text. When generating sentence pairs for pretraining, half of them are indeed consecutive sentences with the label "True"; while the other half is randomly sampled from the corpus with the label "False".

3.3 The Intuitions

3.3.1 Transformer

In one sentence, the encoder-decoder attention is trained to associate the input sentence with the corresponding output word. The intuition behind the encoder-decoder attention layer is to combine the input and output sentence. The encoder's output encapsulates the final embedding of the input sentence. It is like our database. So we will use the encoder output to produce the key and value matrices. On the other hand, the output of the Masked Multi-head attention block contains the so far generated new sentence and is represented as the query matrix in the attention layer. Again, it is the "search" in the database. It will eventually determine how related each English word is with respect to the French words. This is essentially where the mapping between English and French is happening.

3.3.2 BERT

As we learn different aspects of the language, we realize that getting exposed to a variety of text is very helpful to apply Transfer learning. We start reading books to build a strong vocabulary and understanding of the language. When certain words in a sentence are masked or hidden, then based on our knowledge of the language and reading the full sentence from left to right and from right to left (Bidirectional). We can now predict the masked words with better accuracy (Masked Language Modeling). It is like filling in the blanks. We can also predict when the two sentences are

related or not (Next Sentence Prediction). This is a simple working of BERT: Bidirectional Encoder Representations from Transformers.

4 Data Analysis

To show the scope of the methodology mentioned in Section 3, we used the English-French dataset and trained on the first $5,000$ data. We trained the data for 200 epochs and for each epoch, the dataset was randomly split into several batch with batch size 64 to mimic stochastic gradient descent. The number of hidden layer used in the vanilla transformer was 64 and the inputs and the number of hidden layer in the feed-forward network were also 64. The number of multi-head was 4. The learning rate is set to be 5e-3 to increase to converge speed. To avoid the overfitting and stick at a local optima, the dropout rate was set to be 10%. The model was trained on GPU provided by Colab from Google.

To get the word embedding from BERT model, we pretrained the BERT on `WikiText-2`. The `WikiText` language modeling dataset is a collection of tokens extracted from the set of verified articles on Wikipedia. The dataset contained $20,256$ vocabularies. The BERT we implemented was the BERT-tiny which converted each token into a length 64 vector. After pretraining BERT, it was joined with the same decoder we used in the vanilla transformer to solve the downstream machine translation problem.

Figure 5 showed the performance of the vanilla transformer, we observed that the algorithm converged and had loss approximately 0.036. To evaluate the quality of translation, we implemented BLEU and computed the average BLEU on the training dataset to be 0.59. Table 2 contained some examples of the translation:

Table 2: Translation by vanilla Transformer

English	Translation by Machine	BLEU
go.	va!	1.000
come over.	venez chez moi!	0.000
go home now.	va chez toi , maintenant.	0.550
we're strong.	nous sommes <unk>.	0.658

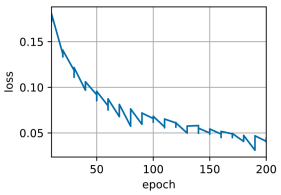

Figure 5: The loss function with the vanilla transformer model.

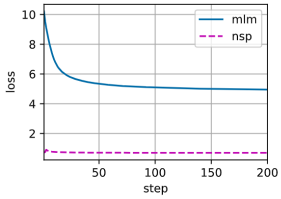

Figure 6: The loss functions of MLM and NSP with BERT-Tiny.

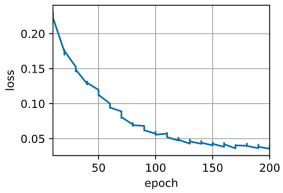

Figure 7: The loss function with the BERT embedding transformer model.

Figure 6 presented the process of the pretraining. The purpose of pretraining was to create vector representation for each token regardless of the downstream tasks. As a result, we used both MLM and NSP to evaluate the pretrain. Figure 7 showed the performance of the transformer using pretrained BERT, we observed that the algorithm converged and had loss approximately 0.024. To evaluate the quality of translation, we implemented BLEU and computed the average BLEU on the training dataset to be 0.58. Based on the result, we observed that the loss of Transformer with BERT is only 2/3 of the loss of vanilla Transformer and they had comparable BLEU. Further discussion on the limited improvement on BLEU is provided in Section 5. Table 3 contained some examples of the translation:

Table 3: Translation by Transformer based on BERT

English	Translation by Machine	BLEU
go.	va!	1.000
come over.	viens chez moi!	0.658
go home now.	va chez toi , maintenant.	0.550
we're strong.	nous sommes <unk>.	0.658

5 Conclusions and Discussions

We first discuss the advantage of Transformer and BERT. Transformer replaces the RNN block in the encoder-decoder structure with (multi-head) self attention layer. This design allows the possibility of parallel computing; therefore, we gain a shorter computing time. However, a pure self attention layer does not take the position information into account, as a result, a positional encoding is added to the embedded token. BERT model is an unsupervised machine learning algorithm that embeds the input vector from another dataset based on its neighborhood. BERT can be connected with a downstream model for our application. It authorizes us to fine-tune the parameters in BERT as well so we are starting from a nice initial point.

Next, we discuss the limitation of BERT. As we observed from the result, BERT model didn't increase the BLEU by a lot. There are two possible explanation. First, the objective to be minimized was the cross-entropy instead of the BLEU. It's noteworthy that lower cross-entropy did not necessarily guarantee lower BLEU. The reason BLEU was not chosen to be the objective function is because it is not differentiable. Discussions and solution was discussed in Ranzato et al. [16]. Secondly, it is not a good idea to simply using BERT to warm up a machine translation model (Zhu et al. [20]). Zhu et al. [20] proposed a new algorithm called BERT-fused where BERT were first used to extract representations for an input sequence, and then, through attention mechanisms, the representations were fused with each layer of the encoder and decoder of the machine translation model.

Radford et al. [15] proposed GPT models as a successor of BERT. Unlike BERT models, GPT models are unidirectional. The major advantage of GPT models is the size. GPT-3 models (Brown et al. [3]), the third-generation GPT model, are trained on 175 billion parameters, about 470 times the size of BERT. The massive pretrained model allows users to fine-tune NLP tasks to accomplish novel tasks using limited data.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] J. Cheng, L. Dong, and M. Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[5] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[11] E. A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

[12] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[13] R. Paulus, C. Xiong, and R. Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[14] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[15] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018.

[16] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[19] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

[20] J. Zhu, Y. Xia, L. Wu, D. He, T. Qin, W. Zhou, H. Li, and T.-Y. Liu. Incorporating bert into neural machine translation. *arXiv preprint arXiv:2002.06823*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyper-parameters, how they were chosen)? [Yes] See Section 4
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

Code can be found in the author's Github.