Sound Natural: Content Rephrasing in Dialog Systems

Arash Einolghozati* Facebook arashe@fb.com

Anchit Gupta* Facebook anchit@fb.com

Keith Diedrick Facebook kdiedrick@fb.com

Sonal Gupta Facebook sonalgupta@fb.com

Abstract

We introduce a new task of rephrasing for a more natural virtual assistant. Currently, virtual assistants work in the paradigm of intent-slot tagging and the slot values are directly passed as-is to the execution engine. However, this setup fails in some scenarios such as messaging when the query given by the user needs to be changed before repeating it or sending it to another user. For example, for queries like 'ask my wife if she can pick up the kids' or 'remind me to take my pills', we need to rephrase the content to 'can you pick up the kids' and 'take your pills'. In this paper, we study the problem of rephrasing with messaging as a use case and release a dataset of 3000 pairs of original query and rephrased query. We show that BART, a pre-trained transformers-based masked language model with auto-regressive decoding, is a strong baseline for the task, and show improvements by adding a copy-pointer and copy loss to it. We analyze different trade-offs of BART-based and LSTM-based seq2seq models, and propose a distilled LSTM-based seq2seq as the best practical model.

1 Introduction

Virtual assistants have achieved very high accuracy in the parsing of queries for execution (Gupta et al., 2018) , such as finding weather or setting a reminder. However, in some scenarios, parsing of the queries alone is not enough to execute those queries as expected. For example, when the user asks "Tell Alice I'll meet her in 10 minutes", a more appropriate message to send is "I'll meet you in 10 minutes" instead of the tagged content in the query, "I'll meet her in 10 minutes". The other scenario where rephrasing is needed to confirm the user's request is when the user asks "Remind me to brush my teeth tonight". A more natural response would be "OK, I'll remind you to brush your teeth tonight".

To make a virtual assistant sound more natural, it needs to rephrase the user's query content before executing it. The task is different from paraphrasing, as we do not want to change the user's wording, e.g., the language formality or choice of words. Instead, we need to make minimal syntactic changes to make the utterance sound natural. As a use case, we work on the messaging domain, where we focus on rephrasing a message that needs to be inferred from the user query. Unlike the confirmation case, the message rephrasing is more complicated and can involve syntactic, pronoun or verb changes. Note that our goal is not to paraphrase the user's message but to rephrase it minimally, making it sound more natural when being sent to another user. As such, we need to maintain the semantics and the style of the original content.

Our contributions are as follows: (1) We introduce a new task and release a Message Content Rephrasing (MCR) dataset for this task consisting of 3k queries with tagged content and possible rephrases, (2) We explore various modeling approaches to achieve high accuracy on the MCR and modify existing pre-trained models to accommodate for the nature of this task, and (3) We show that distilling the pre-trained models into simple models can significantly close the performance gap.

2 Data

We first collected a task-oriented dataset of messaging utterances by asking our annotators to come up with natural scenarios in which a user wants to send a message to a second user.

We observed that the collected utterances consist of two types of messaging queries: 1) where the content needs to be rephrased (REPHRASE) and 2) where the content should be used in verbatim (EX-

*equal contribution

ACT). As such, the utterances were sent to another set of annotators to mark the message content in the utterances, disregard the ones that do not contain one, and mark whether the utterance belongs to the EXACT or REPHRASE class. We used two annotators for this task, with a possible third for disagreement resolution. Examples of each class is shown in Table 1 where the original content is tagged by brackets around it.

Next, we sent the utterances belonging to the REPHRASE class to yet another set of annotators and asked them to rephrase the message content in a way that it would be natural to send to the second user without any additional context.

During annotation, our goal was to minimally rephrase a sentence, e.g., keeping the words and attributes (e.g., formality) of the original content as much as possible. In order to ensure high quality, we asked three annotators to independently rephrase the utterances. In around 30% of the cases, there was not a majority (i.e., two or more annotators agree) and we asked a fourth annotator to resolve. Most of the disagreements were due to changing words that did not need to be changed for minimal rephrasing but there were cases that the minimal rephrase was not obvious. We will discuss this further when introducing our metrics.

Overall, we have around 3k examples (almost half for each class) which we split by 70/20/10 for train/test/validation, respectively. We can see from the training data that rephrasing involves making a question and/or changing the subject pronoun. As such, we cluster the rephrasing into three main categories. In Table 1, the first example only needs a pronoun change, the second needs the form to be changed to a question, and the third needs both. We have also put the statistics for the changes in table 3. [1]

As mentioned earlier, there is a huge overlap between the source and target sequences. As such it can be viewed as a post-editing task more than a generation task. In Table 2, we have showed some basic statistics about the training data for the REPHRASE class in MCR.

3 Evaluation

Our goal is to maximize the rephrasing accuracy while also maintaining a very high accuracy on the EXACT class. Our first metric is the Exact Match (EM) accuracy in which the predicted rephrase should be the same as the original content for the EXACT class and equal to the top rephrased candidate for the REPHRASE class. The downside of this metric is that for utterances such as 'ask her to pick up her phone', we would penalize rephrases such as 'can you pick up your phone' if the gold label was 'pick up your phone'. As such, in order to smooth this metric, we also use EM_{any} in which the rephrased content is correct if it matches any of the provided annotations.

Since the required changes to rephrase the content are usually small, the BLEU score may not be useful. On the other hand, not all the wrong rephrases are equal, e.g., when the model hallucinates. Metrics such as BLEU can penalize these phenomena more than the EM metrics. We also use SARI (Xu et al., 2016), which is commonly used for text-editing tasks. It measures the average F1 score of three editing actions for ngrams: Keep, Add, and Delete.

4 Modeling Approaches

We assume that the gold tagging for the content inside the query is provided. Our base model is an LSTM seq2seq model with two-layers for both encoder and decoder using Glove (Pennington et al., 2014) initialized word embeddings (20k vocab size) concatenated with ELMo (Peters et al., 2018) embeddings to represent the tokens. We also use the pointer-generator mechanism (See et al., 2017), which can choose between copying from the source or generating new tokens using a pointer-attention mechanism. As we can see in Table 4, the copy mechanism is crucial in our task, as most of the tokens are copied from the source.

The copy pointer works as follows: We calculate two token output probabilities; one over the full vocab P_{vocab} using the standard softmax and another P_{copy} over the source tokens. We use attention between the current decoder hidden h_d^t and the encoder outputs H_e^T to obtain P_{copy}. To generate the output, we weigh between copying and generation using a learnable parameter α_{mix}, i.e., $P_{output} = (1 - \alpha_{mix})P_{vocab} + \alpha_{mix}P_{copy}$. More precisely:

$$q^t, K, V = h_d^t W_q^T, H_e^T W_k^T, H_e^T W_v^T$$

$$P_{copy}^t = softmax(q^T K)$$

$$\alpha_{mix}^t = sigmoid(W_{mix}.cat(q^T K, V)),$$

[1] The dataset can be downloaded from dl.fbaipublicfiles.com/rephrasing/rephrasing_dataset.tar.gz

Query	Query	Rephrase
REPHRASE	Let Kira know [I can pick her up]	I can pick you up
REPHRASE	Message Donna and ask [when dinner is]	when is dinner
REPHRASE	message Brad and ask [if he has my keys]	do you have my keys
EXACT	Tell Jo [I will be on time]	I will be on time
EXACT	ask my boss [will I have to work on Friday]	Will I have to work on Friday

Table 1: Examples of queries and the rephrased utterance

Source	Target	Keep	Add	Delete
7.9	9.3	5.9	3.4	2.0

Table 2: Average Length and overlap between source and target

EXACT (no changes)	57%
REPHRASE (pronoun)	8%
REPHRASE (question)	13%
REPHRASE (pronoun+question)	22%

Table 3: Frequency of the needed changes

where W_{mix} is learned.

We have shown the LSTM results alongside ablation on the ELMo and Copying mechanism in Table 4. We can see that copying is crucial, especially for the EXACT class. We show the results for copying the content part of the source in the first row.

We also experiment with using BART (Lewis et al., 2019) for this task. BART is a powerful pre-trained seq2seq model trained on a de-noising objective over massive amount of web data. The training details and the validation results are listed in the Appendix. During our initial experiments with BART, we realized it can replace proper nouns when rephrasing. Even though BART is a de-noising autoencoder and it has a high proclivity to copy the source through its encoder-decoder attention heads, it is still done over the whole vocabulary space (50k bpe tokens) and not the dozen of source tokens. To address this PEGASUS (Zhang et al., 2019) is pre-trained by generating a selected masked sentence from the input, where some of the selected sentences are not masked. We instead opt to add an explicit copying to BART in the fine-tuning stage.

Since the pre-trained model has no explicit copy mechanism, adding it naively during the fine-tuning phase, as above, is not effective. In this case, the decoder prefers to use the well-trained generator instead of a randomly initialized attention head for copying. We use two strategies to mitigate this: we initialize the copying attention head with the average of the last layer's pre-trained decoder attention head. We also add an explicit loss that forces the decoder to use the copying mechanism when it can. As such, for all the target tokens that can be found in the source, we add a hinge loss: $\lambda max(T - P, 0)$ to the cross-entropy loss which forces the copying probability P for those token to be above a threshold T. Hyper-parameters λ and T are optimized over the validation set, 0.25 and 0.9, respectively.

We show results using the BART large model in Table 4. Vanilla BART yields strong results compared with the LSTM seq2seq model for the rephrasing class but slightly lags for EM_{exact}, which requires pure copying. On the other hand, by adding the explicit copying to BART, it significantly improves the accuracy for both classes. Moreover, the gap between EM and EM_{any}, the biggest for BART, which shows the proportion of errors due to subtle differences with the resolved annotation, as opposed to serious problems such as hallucination.

4.1 Distilling BART

Deploying models such as BART can be prohibitive for real-time applications. It has 514M parameters and around 10X average CPU inference latency compared with the LSTM model that has only 9.6M parameters. Moreover, unlike the pointer-generator LSTM model, BART with copying still exhibits an over-generation problem while the LSTM model makes many grammatical errors. As such, we look into Knowledge Distillation (KD) (Hinton et al., 2015) to transfer the language modeling capability of BART while keeping its copying behavior. Transferring the language model of massive pre-trained models into smaller models has been of high interest recently (Sanh et al., 2019; Turc et al.,

Model	EM	EM_{any}	BLEU	EM_{exact}	$EM_{rephrase}$	SARI
Exact Copy	55.0	55.0	80.6	100	0	26.3
LSTM seq2seq	84.1	85.8	91.0	96.6	68.9	83.1
LSTM seq2seq w/o ELMo	81.3	82.4	89.4	93.8	66.1	81.3
LSTM seq2seq w/o Copy	54.7	55.8	78.9	62.3	39.2	69.7
BART	88.2	90.5	**96.0**	95.5	79.2	**86.4**
BART w copy	**89.3**	**92.1**	**96.1**	**96.9**	**80.0**	**86.5**
LSTM + seq-level KD	84.1	86.1	90.5	96.6	68.9	82.9
LSTM + seq-level KD + FT	85.4	87.2	94.0	95.5	73.0	83.7
LaserTagger	87.4	88.7	94.6	**97.2**	75.8	84.0

Table 4: Rephrasing Model Performance

2020; Sun et al., 2019). Knowledge transfer to simple models has also been discussed in lesser extent (Tang et al., 2019; Mukherjee and Awadallah, 2019). We use the sequence-level distillation introduced in (Kim and Rush, 2016) and train the LSTM model using the BART output. We found that fine-tuning on the gold labels after the the KD step is also beneficial to the performance.

4.2 Edit vs Generate

In a pure generation framework, e.g., BART without the copying loss, all the tokens are generated from scratch. On the other side of the spectrum, models such as LaserTagger (Malmi et al., 2019) keep the original utterance and try to edit by adding or removing as needed. Adding the copying mechanism to our models can be considered a middle ground between editing and generation. We use the framework introduced in (Malmi et al., 2019) to edit the queries. It tags each word as Keep or Delete plus the optional phrase that needs to be added before it. We procure the list of phrases that yield high coverage over the training data in MCR. By using the top 100 phrases, we get coverage over 95% of the training data. Note that the verb conjugations needed in our problem can cause lack of generalization when using such limited vocabulary.

We train a tagging model using the RoBERTa encoder (Liu et al., 2019) with one layer of MLP and CRF on top of it. We have listed the editing model performance on the last line of the Table 4. We can see that the editing yields better EM than the LSTM model but worse than BART. It is unsurprisingly the best model when no rephrasing is needed. On the other hand, the type of rephrasing errors it makes may be worse than the generative models as evidenced by the lower SARI score. For example, we find grammatical errors such as "did you I leave my sunglasses there". This is perhaps because the added words are treated as categorical classes and not as words in a LM.

4.3 Error Analysis

We cluster the errors into three categories with an additional 'Correct' class which means that the prediction is correct but does not match any of the gold annotation exactly. A prominent example of the latter is the addition of politeness prefixes such as 'Could you' to the beginning of a request.

The Grammatical error class represents cases that the semantic can be understood but contains some grammatical errors such as mismatch between the noun and verb forms. In the Semantic error class, the meaning is seriously hurt and contains two sub-categories; hallucinating new content and omission of parts of the content. The Copy-related error category happens mostly for the proper nouns when they are not exactly carried over in the output. Since this is observed mostly in the vanilla BART, we decided to separate this category from the rest of the semantic errors.

Note that if there are multiple classes of errors in the output, we pick the most prominent type of error for that utterance. In Table 5, we have shown the prevalence of each Category. we can see that in BART models, the majority of the ostensible errors are actually correct but the BART model without the explicit copying has the biggest copy-related errors among all models. Moreover, while Grammatical is the biggest category in both the distilled LSTM and the LaserTagger, the latter makes many more semantic errors which echoes our qualitative observation.

5 Related Work

5.1 Pre-trained Models for Generation

Pre-training transformers on massive amounts of unlabeled data has resulted in recent advances in language understanding and generation tasks (Devlin et al., 2019; Radford, 2018). Pre-trained encoder-decoder models have unified the benefits for both discriminative and genera-

Model	Semantic	Grammatical	Copy Related	Correct
BART	4%	13%	**24%**	59%
BART w Copy	14%	10%	8%	**68%**
Distilled LSTM	8%	**45%**	8%	39%
LaserTagger	**25%**	38%	4%	33%

Table 5: Prevalence of each category of the models' mispredictions

tive tasks through pre-training as de-noising auto-encoders (Song et al., 2019; Lewis et al., 2019; Raffel et al., 2019). (Chen et al., 2019) fine tune such a big pre-trained model and add a copy pointer for a few shot structured tabular data summarization task.

5.2 Paraphrasing

Paraphrase generation using seq2seq models (Sutskever et al., 2014) has been recently discussed in the literature. Prakash et al. (2016) used residual LSTM seq2seq networks to perform paraphrasing. Unlike paraphrasing, in MCR, preserving the semantics of a message is necessary but not enough. Instead, we make minimal changes to make the sentence sound natural.

5.3 Sentence Editing and Simplification

Automatic post-editing is applied to paraphrases and machine translation (Grangier and Auli, 2018). Similar to this is Grammatical Error Correction which seeks to correct errors such as grammar and punctuation (Ng et al., 2014; Zhao et al., 2019). Sentence revision (Ito et al., 2019) extends this to cases for which major rewriting may be needed. Sentence simplification (Nisioi et al., 2017) aims at using techniques such as shortening the sentences to make a text more readable. On the other hand, style transfer is the task of making an utterance conform to a specific style such as formality (Logeswaran et al., 2018; Sennrich et al., 2016). From this perspective, the rephrasing task can be viewed as changing the style from the third-person to the second-person language and/or forming a question.

6 Conclusion

In this paper, we introduce a new task of message rephrasing in task-oriented dialog. We release a dataset, MCR, for this task and propose a new model (BART with copy). We show that adding an explicit loss to a pre-trained generative model during fine-tuning can improve the copying performance without hurting its generation power. We also show that by distilling the pre-trained model into a much smaller LSMT seq2seq model with copy pointer, we can significantly improve the LSTM seq2seq model's language model capability while still keeping its accurate copying.

References

Ahmed Aly, Kushal Lakhotia, Shicong Zhao, Mrinal Mohit, Barlas Oguz, Abhinav Arora, Sonal Gupta, Christopher Dewan, Stef Nelson-Lindall, and Rushin Shah. 2018. Pytext: A seamless path from NLP research to production. *CoRR*, abs/1812.08729.

Zhiyu Chen, Harini Eavani, Yinyin Liu, and William Yang Wang. 2019. Few-shot NLG with pre-trained language model. *CoRR*, abs/1904.09521.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

David Grangier and Michael Auli. 2018. QuickEdit: Editing text & translations by crossing words out. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 272–282, New Orleans, Louisiana. Association for Computational Linguistics.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Takumi Ito, Tatsuki Kuribayashi, Hayato Kobayashi, Ana Brassard, Masato Hagiwara, Jun Suzuki, and Kentaro Inui. 2019. Diamonds in the rough: Generating fluent sentences from early-stage drafts for academic writing assistance. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 40–53, Tokyo, Japan. Association for Computational Linguistics.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lajanugen Logeswaran, Honglak Lee, and Samy Bengio. 2018. Content preserving text generation with attribute controls. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5103–5113. Curran Associates, Inc.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5054–5065, Hong Kong, China. Association for Computational Linguistics.

Subhabrata Mukherjee and Ahmed Awadallah. 2019. Distilling transformers into simple neural networks with unlabeled transfer data.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 85–91, Vancouver, Canada. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2923–2934, Osaka, Japan. The COLING 2016 Organizing Committee.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC^{2}Workshop*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, San Diego, California. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *ICML*.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*

and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4323–4332, Hong Kong, China. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2020. Well-read students learn better: On the importance of pre-training compact models.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *ArXiv*, abs/1912.08777.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics.

Model	BSz	LR	WD	Epoch	Avg Time
BART	10	. 0.00002	e-5	10	1hr
LSTM seq2seq	16	0.001	e-6	40	45min
RoBERTa	16	0.000005	e-4	10	4hr

Table 6: Training Parameters

A Appendix

Here, we describe the details regarding the training as well as the validation results.

In Table 6, we have shown the training details for all of our models. We use ADAM (Kingma and Ba, 2014) with Learning Rate (LR), Weight Decay (WD), and Batch Size (BSz) that are listed for each model. We have also shown the number of epochs and the average training time for the full CS data using 4 V100 Nvidia GPUs.

In all of our BART experiments, we have used the BART large from the PyText[2] (Aly et al., 2018). When adding the copying loss to BART, we fine-tuned the hyper-parameters λ and T over [0.1, 1] and [0.5,1], respectively, with increments of 0.05. We also use the RoBERTa large from the PyText for the LaserTagger experiment.

In our LSTM models, the encoder and decoder are 2-layer LSTMs with hidden dimension of 128 and 256, respectively. We also use dropout of 0.3 for all connections. The ELMo and GloVe embeddings have dimesions of 512 and 200, respectively, and we use the top 8k words in GloVe as our vocabulary.

In Table 7, we have listed the validation results for the models described in the paper.

[2] https://pytext.readthedocs.io/en/master/xlm_r.html

Model	**EM**	**EM$_{\text{any}}$**	**BLEU**	**EM$_{\text{exact}}$**	**EM$_{\text{rephrase}}$**	**SARI**
Exact Copy	57.0	57.0	82.3	100	0	25.5
LSTM seq2seq	84.3	86.2	91.4	96.9	68.7	83.3
LSTM seq2seq w/o ELMo	81.6	82.7	90.2	94.0	66.0	81.8
LSTM seq2seq w/o Copy	53.7	54.6	77.8	61.0	38.7	69.7
BART	87.5	90.0	95.3	96.1	77.2	85.0
BART w copy	**89.1**	**91.0**	**95.6**	**97.4**	**77.9**	**85.8**
LSTM + seq-level KD	84.0	85.9	90.4	96.5	69.0	83.0
LSTM + seq-level KD + FT	85.5	87.4	94.3	95.8	73.1	83.8
LaserTagger	87.6	88.8	94.7	97.1	75.6	84.2

Table 7: Rephrasing Model Performance for the validation set