Mitigating Data Scarceness through Data Synthesis, Augmentation and Curriculum for Abstractive Summarization

Ahmed Magooda and **Diane Litman**
University of Pittsburgh
Pittsburgh, PA, USA
{aem132, dlitman}@pitt.edu

Abstract

This paper explores three simple data manipulation techniques (synthesis, augmentation, curriculum) for improving abstractive summarization models without the need for any additional data. We introduce a method of data synthesis with paraphrasing, a data augmentation technique with sample mixing, and curriculum learning with two new difficulty metrics based on specificity and abstractiveness. We conduct experiments to show that these three techniques can help improve abstractive summarization across two summarization models and two different small datasets. Furthermore, we show that these techniques can improve performance when applied in isolation and when combined.

1 Introduction

Training complex neural models usually requires large amounts of data. However, data annotation still poses a challenge for many domains. Thus, much research focuses on data manipulation (e.g., synthesis, augmentation) and additional ways to handle data differently during training. Prior work on the synthesis of textual data has focused on back translation (Parida and Motlicek, 2019; Wang et al., 2018; Sennrich et al., 2016) and word replacement (Wang and Yang, 2015; Zhang et al., 2015). We, on the other hand, propose a different approach to data synthesis through paraphrasing. However, synthesis involves data manipulation on the input level, which might expose the model to grammatically or logically incorrect input. Thus, we explore a second approach to data manipulation based on augmentation rather than synthesis. Augmentation aims to move data manipulation from the input side to any part of the model. This, in turn, can help the model be more resilient to over-fitting. Third, we explore using data more efficiently by integrating curriculum learning into the training process. Curriculum learning reorders training samples based

on external criteria, which can help train the model gradually and more efficiently without the need for any external data. We also introduce new difficulty metrics based on specificity and abstractiveness for curriculum construction. Finally, we explore combining multiple techniques (synthesis and curriculum) to overcome the data scarcity issue. Thus, our contribution is threefold: 1) We introduce a simple approach for data synthesis through paraphrasing. 2) We use data augmentation by sample mixing to move augmentation into the model. 3) We integrate a curriculum into the training process and introduce two new difficulty metrics.

2 Related Work

Abstractive summarization for low resource data. Prior proposed methods for tackling domains with scarce data have included finetuning pre-trained models (Bajaj et al., 2021; Yu et al., 2021; Magooda and Litman, 2020) such as BART (Lewis et al., 2020) or using few-shot learning (Bražinskas et al., 2020; Sarkhel et al., 2020). Our work differs in several aspects. First, our work doesn't focus on improving a certain summarization model; in contrast, we focus on using data efficiently, which can be applied to various models. Second, we focus on techniques that can improve the training process without additional data, e.g., synthesis, augmentation, and curriculum learning.

Data synthesis and augmentation. Data synthesis for text summarization is underexplored, with only a few approaches such as back-generation (Parida and Motlicek, 2019) and template-based summary re-writing (Magooda and Litman, 2020). We propose doing data synthesis by paraphrasing, which is simpler than the back-translation and template methods. While combining synthesis with paraphrasing has been studied in other contexts (Wang et al., 2015; Iyyer et al., 2018), our work differs in both goals and techniques. Wang et al. (2015) proposed synthesizing data, then crowd-

Data	# docs	# refs	Train	Val	Test
CM ALL	368	44	294	37	37
Amazon/Yelp	160	8	58	42	60

Table 1: Dataset summary.

sourcing paraphrases to train semantic parsers, while Iyyer et al. (2018) synthesized data to train a paraphrasing model. Our work, to our knowledge, is the first to use a strong language model finetuned for paraphrasing to synthesize data for text summarization. Finally, for data augmentation, we base our work on the MixText approach (Chen et al., 2020). While the original MixText model is used for classification-based tasks, we introduce a variation for generative tasks (called MixGen) and use it for abstractive summarization.

Curriculum learning. Curriculum learning aims to improve the training procedure with the same amount of data. It has been applied in NLP (Sachan and Xing, 2016, 2018; Tay et al., 2019; Xu et al., 2020; Wang et al., 2020) for machine comprehension, question generation, reading comprehension, NLU and machine translation, respectively. We build on the approach introduced in (Xu et al., 2020); however, the core differences are both the downstream tasks (classification versus abstractive summarization) and the difficulty metrics. In contrast to the only other summarization work that we know of, Kano et al. (2021) focus on large datasets, while we focus on low resource domains. We also introduce two different difficulty metrics (ROUGE and specificity).

3 Summarization Datasets

CourseMirror (CM) is a student reflections dataset that has been used in prior work to study extractive (Luo and Litman, 2015) and abstractive (Magooda and Litman, 2020)[1] summarization. The dataset consists of documents and summaries from four courses. Table 1 summarizes the dataset in terms of the *number of documents (# docs)* and *average reflections per document (# refs)*. We compiled all courses into one dataset (named CM ALL), then split the documents into training, validation, and test sets (80%, 10%, 10%, respectively) by sampling equally from all courses.

Amazon/Yelp (A/Y) is another small dataset, now consisting of opinions (refer to the appendix

for examples) (Bražinskas et al., 2020)[2]. The dataset contains customer reviews from Amazon (He and McAuley, 2016) and Yelp. The data contains 160 products/businesses split into training, validation and test sets as shown in Table 1. Each of the products/businesses contains a set of 8 reviews.

4 Proposed Model

4.1 Synthesis via paraphrasing with GPT-2

Influenced by work in style transfer (Krishna et al., 2020), we propose synthesizing new human summaries by using paraphrasing to generate other potential summaries that are paraphrases of the original human summary. We use the paraphraser trained by Krishna et al. (2020). They finetuned a large GPT-2 language model with data from PARANMT-50M (Wieting and Gimpel, 2018) to direct the model into generating diverse paraphrases that they later used for style transfer.

4.2 Augmentation with sample mixing

MixText is a data augmentation approach based on mixing two input samples by weight summing the features corresponding to the two samples at any level of the model (specific layer of the encoder, after encoder, etc.) using λ. The model is then expected to produce a probability distribution over the available classes, similar to a λ weighted sum of the two samples' gold predictions. We train the model using KL divergence between the predicted distribution and the expected one. We adapt the approach of Chen et al. (2020) for text generation tasks by modifying the decoding process and loss calculation; we call our approach MixGEN. Like the original MixText, we use two input samples and pass them to the encoder. We pass the samples up to a specific layer, then the two hidden states are summed together weighted differently using the λ parameter. On the decoder side, first, we construct the expected values using the following:

$$for\ i\ \in\ min(L_1, L_2)\ \ PD_i = [P_1, P_2, P_3, ..P_v],$$

$$for\ j\ \in\ [1, v] \begin{cases} P_j = \lambda, & \text{if } S_1[i] = j \\ P_j = 1 - \lambda, & \text{if } S_2[i] = j \\ P_j = 0, & \text{otherwise} \end{cases}$$

where v = vocab size, and L_1, L_2 are the human summary of input sample 1 and input sample 2,

---

[1]https://petal-cs-pitt.github.io/data.html

[2]https://github.com/abrazinskas/FewSum

respectively. PD_i is the probability distribution expected for $token_i$. $S_1[i]$, $S_2[i]$ are the ith token of the first sample's and second sample's human summary, respectively. In simple wording, during decoding, we expect the output probability distribution across the vocabulary of the decoder at token position i to have two high values, one with value λ at vocabulary token corresponding to the ith token of the first sample's human summary, and another value of $1 - \lambda$ at vocabulary token corresponding to the ith token of the second sample's human summary. This should continue as long as i is less than or equal to both human summaries' length. Once i is greater than the minimum summaries' length, then the expected distribution would only correspond to the longer summary. Finally, the text generation on the decoder side is auto-regressive, thus, expected token is passed at the end of each generation step. However, If we pass the argmax of the expected distribution, then we will end up always passing the token corresponding to the sample with higher weight (Alpha) vs. (1-Alpha). Thus, we randomly sample from the two input samples based on their weights using the following equation.

$$for \ i \ \in \ [1, L_{min}], \ P_i \sim U(0,1)$$
$$for \ i \ \in \ [1, L_{min}] \begin{cases} P_i <= \lambda, & T_i \ from \ S_1 \\ P_i > \lambda, & T_i \ from \ S_2 \end{cases}$$

where S_1 and S_2 are the first and second sample respectively, L_{min} is minimum length of both S_1 and S_2. P_i is the sampling probability and $U(0,1)$ is a uniform distribution.

4.3 Curriculum learning (Cur.)

Curriculum learning aims to help the model training process by introducing easier samples first followed by more difficult ones according to a particular difficulty metric. We use the curriculum construction approach introduced in Xu et al. (2020). In this approach, we split data into N buckets based on a difficulty metric. We then train the model in a difficulty incremental setting. In this work, we use two different curriculum difficulty metrics.

Specificity (S) measures how specific or vague a piece of text is. We argue that the more specific a piece of text is, the more complicated it can get. For example, text like (Nothing, Everything is Easy, etc.) are not specific and easy for the model to learn and vice versa. We feed the model less specific pieces of text first during training, then introduce the more specific ones as training progresses. Specificity is calculated (Appendix) on the reflection/review level, so we use the average values of the whole set of reflections/reviews as the document value. For example; for a training sample of an input document D consists of N independent reflections/reviews $[r_1, r_2, r_3, .., r_n]$, we calculate the specificity value for the sample as follows:

$$D_s = \sum_{i=1}^{N} S(r_i)/N$$

where $S(r_i)$ is specificity value of the i's reflection/review.

ROUGE (R) is the standard metric for evaluating summarization performance. Thus, we decided to use ROUGE scores as a difficulty metric. For a training sample, we calculate different ROUGE scores between the input document D and its corresponding human summary S, then use *average of (R1, R2, RL)* as the difficulty metric. According to (Liu et al., 2018), the higher the ROUGE score, the less abstractive the summary is compared to the input, and vice versa. We argue that the more abstractive samples are harder to learn.

5 Experiments

5.1 Parameters

Baselines: To our knowledge, in prior work there is no data synthesis technique used for summarization except back generation (Parida and Motlicek, 2019) and template synthesis (Magooda and Litman, 2020). Thus, we developed two synthesis baselines (**shuffle; shuffle + mask**). We generated 10 samples for each of the original training samples for both baselines by randomly shuffling the reflections/reviews. Additionally, for the shuffle-mask baseline, we randomly mask 50% of the reflection/reviews 50% of the time.

Paraphrasing with GPT-2: We generate N synthetic samples for each original sample by generating N paraphrases of the human summary and shuffle the input reflections. We varied N between [5, 10] to monitor the effect of synthetic data size.

MixGEN: We integrate MixGEN by combining each sample with N other samples during training. We used N=3 for our experiments. Moreover, we use mixing probability α=0.75 as specified by the original code implementation[3].

---

[3]https://github.com/GT-SALT/MixText

Curriculum learning: In the curriculum learning experiments, we use a specificity prediction model that consists of a DistilBERT (Sanh et al., 2019) encoder with a logistic regression classification layer (Appendix). We normalize the whole training data values between 1 and N, where N is the number of buckets to split the data. We use N=10. Similarly, we normalize the average ROUGE value to also be between 1 and N.

5.2 Model Training

In all of our experiments, we use the BERTSum[4] model proposed by Liu and Lapata (2019). We used the same parameters in the original code (Appendix). We conducted experiments on CM and A/Y datasets using proposed methods in a regular training and in a (**pretraining→fine-tuning**) setting, where we perform pretraining with synthesized data and fine-tuneing using original data.

6 Results

Pretraining	finetuning	CM		
		R1	R2	RL
No Pretraining				
None	Original	36.34	11.39	26
	shuff.	**38.57**	11.72	26.94
	shuff.+mask	37.07	11.52	26.45
	Cur.(S)	36.88	**12.41**	**27.63**
	Cur.(R)	37.01	12.13	27.11
	Mix(n=3)	36.87	11.98	26.57
With synthetic data pretraining				
Synth.(n=5)	Original	39.39	12.85	26.66
	Cur.(S)	40.68	13.59	26.26
	Cur.(R)	39.35	12.33	26.48
Synth.(n=10)	Original	**41.14**	**14.24**	26.98
	Cur.(S)	39.81	12.94	26.81
	Cur.(R)	40.22	14.12	**27.33**

Table 2: ROUGE results of BERTSum model with different augmentation techniques on CM data.

Tables 2 and 3 show results obtained through conducting experiments on CM and A/Y datasets. Considering **data synthesis and augmentation**, we first see that the two baselines (shuffle and shuffle+mask) can improve performance compared to no data manipulation across all ROUGE scores except RL for shuffle baseline on the A/Y dataset. This shows that reducing the model dependency on the input sentence order can help the model depend more on the actual input text. Moving to the proposed augmentation technique (MixGEN),

Pretraining	finetuning	A/Y		
		R1	R2	RL
No Pretraining				
None	Original	27.71	3.83	17.83
	shuff.	28.34	4.04	17.74
	shuff.+mask	28.01	4.21	17.87
	Cur.(S)	28.69	4.28	17.95
	Cur.(R)	**28.8**	**4.33**	**18.18**
	Mix(n=3)	27.85	3.95	17.89
With synthetic data pretraining				
Synth.(n=5)	Original	28.27	4.36	17.84
	Cur.(S)	27.95	4.4	18.01
	Cur.(R)	**28.56**	4.25	18.06
Synth.(n=10)	Original	28.49	**4.54**	18.08
	Cur.(S)	28.52	4.5	**18.15**
	Cur.(R)	28.21	4.35	17.87

Table 3: ROUGE results of BERTSum model with different augmentation techniques on A/Y data.

we see that we can get a performance gain across all ROUGE scores across both datasets by mixing training samples compared to normal training with a single sample. Similarly, we can see that providing synthetic data with the proposed paraphrasing approach can help outperform both using original data as well as baselines with (41.14, 14.24, 26.98) compared to (36.34, 11.39, 26) and (38.57, 11.72, 26.94) for original and shuffle baseline respectively on CM, and (28.49, 4.54, 18.08) compared to (27.71, 3.83, 17.83) and (28.34, 4.04, 17.74) on A/Y. Additionally, we can see that increasing the synthetic data size helps to improve the model performance across all ROUGE scores for both CM and A/Y datasets (N=5 vs. N=10).

Now moving to **curriculum learning**, we can see that integrating a curriculum to reorder training data differently using any of the two proposed difficulty metrics can lead to consistent improvements across all ROUGE scores for both CM and A/Y datasets. Additionally, we can see that curriculum can improve scores compared to the two augmentation baselines across all ROUGE scores except R1 for CM data. On the other hand, we don't see consistent ROUGE score improvement when using curriculum for fine-tuning after pretraining with synthetic data. We hypothesize that this behavior might be due to performing the pretraining phase without curriculum integration, unlike the fine-tuning phase. We plan to conduct experiments with a curriculum integrated into both pretraining and fine-tuning to validate our hypothesis. Furthermore, while both curriculum difficulty metrics (i.e., Specificity and ROUGE) introduced improvement compared to training with no curriculum, we

---

[4] Easy to use and one of the SOTA summarization models. https://github.com/nlpyang/PreSumm

didn't observe any consistent improvement pattern in using one metric over the other.

7 Conclusion and Future work

In this work, we showed that we could mitigate the effect of data scarcity in different datasets (i.e., CourseMirror and Amazon/Yelp) for abstractive summarization using three simple data manipulation techniques. We showed that synthesizing data with paraphrasing to use for pretraining can boost the model performance across all ROUGE scores for different datasets. Additionally, we showed that mixing samples for training can also push the model to be more resilient to overfitting and improve its performance. Finally, we showed that reordering training samples through curriculum, using the proposed difficulty metrics **(i.e., Specificity, and ROUGE)** would help improve all ROUGE scores across different datasets without the need for any additional data (either true or synthetic). In the future, we plan to try more N values for synthesis and MixGen. Additionally, we plan to investigate other curriculum difficulty metrics. We plan to use BART model as one of the SOTA models for abstractive summarization. Finally, we are doing additional experiments on multitask learning, and we plan to combine both techniques in one framework targeting low resource domains.

Acknowledgements

References

Ahsaas Bajaj, Pavitra Dangati, Kalpesh Krishna, Pradhiksha Ashok Kumar, Rheeya Uppaal, Bradford Windsor, Eliot Brenner, Dominic Dotterrer, Rajarshi Das, and Andrew McCallum. 2021. Long document summarization in a low resource setting using pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 71–80, Online. Association for Computational Linguistics.

Arthur Bražinskas, Mirella Lapata, and Ivan Titov. 2020. Few-shot learning for opinion summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4119–4135, Online. Association for Computational Linguistics.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157, Online. Association for Computational Linguistics.

Xiangmin Fan, Wencan Luo, Muhsin Menekse, Diane Litman, and Jingtao Wang. 2017. Scaling reflection prompts in large classrooms via mobile interfaces and natural language processing. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, pages 363–374. ACM.

Ruining He and Julian J. McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 507–517. ACM.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. 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 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Ryuji Kano, Takumi Takahashi, Toru Nishino, Motoki Taniguchi, Tomoki Taniguchi, and Tomoko Ohkuma. 2021. Quantifying appropriateness of summarization data for curriculum learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1395–1405, Online. Association for Computational Linguistics.

Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating unsupervised style transfer as paraphrase generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 737–762, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. 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 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Wencan Luo and Diane Litman. 2015. Summarizing student responses to reflection prompts. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1955–1960, Lisbon, Portugal. Association for Computational Linguistics.

Ahmed Magooda and Diane Litman. 2020. Abstractive summarization for low resource data using domain transfer and data synthesis. In *The Thirty-Third International Flairs Conference*.

Shantipriya Parida and Petr Motlicek. 2019. Abstract text summarization: A low resource challenge. 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 5994–5998, Hong Kong, China. Association for Computational Linguistics.

Mrinmaya Sachan and Eric Xing. 2016. Easy questions first? a case study on curriculum learning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 453–463, Berlin, Germany. Association for Computational Linguistics.

Mrinmaya Sachan and Eric Xing. 2018. Self-training for jointly learning to ask and answer questions. 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 629–640, New Orleans, Louisiana. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv preprint*, abs/1910.01108.

Ritesh Sarkhel, Moniba Keymanesh, Arnab Nandi, and Srinivasan Parthasarathy. 2020. Interpretable multi-headed attention for abstractive summarization at controllable lengths. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6871–6882, Barcelona, Spain (Online).

International Committee on Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Yi Tay, Shuohang Wang, Anh Tuan Luu, Jie Fu, Minh C. Phan, Xingdi Yuan, Jinfeng Rao, Siu Cheung Hui, and Aston Zhang. 2019. Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4922–4931, Florence, Italy. Association for Computational Linguistics.

Wei Wang, Ye Tian, Jiquan Ngiam, Yinfei Yang, Isaac Caswell, and Zarana Parekh. 2020. Learning a multi-domain curriculum for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7711–7723, Online. Association for Computational Linguistics.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563, Lisbon, Portugal. Association for Computational Linguistics.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. SwitchOut: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861, Brussels, Belgium. Association for Computational Linguistics.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342, Beijing, China. Association for Computational Linguistics.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Melbourne, Australia. Association for Computational Linguistics.

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020.

Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6095–6104, Online. Association for Computational Linguistics.

Tiezheng Yu, Zihan Liu, and Pascale Fung. 2021. AdaptSum: Towards low-resource domain adaptation for abstractive summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5892–5904, Online. Association for Computational Linguistics.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

A BERTSum training parameters

We train the model for 200K steps using a batch size of 140. We use 20K steps for BERT warmup, 10K steps for decoder warmup, and a max position of 512. We use 4 Nvidia Quadro RTX 5000 GPUs. We use the checkpoint with highest ROUGE score on validation set for testing.
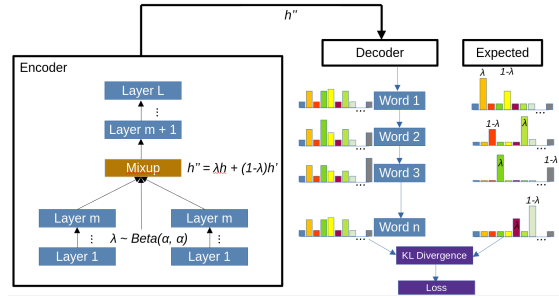
B MixGen Model

Figure 1 shows both the original MixText model and the modified MixText for generative tasks (MixGen).
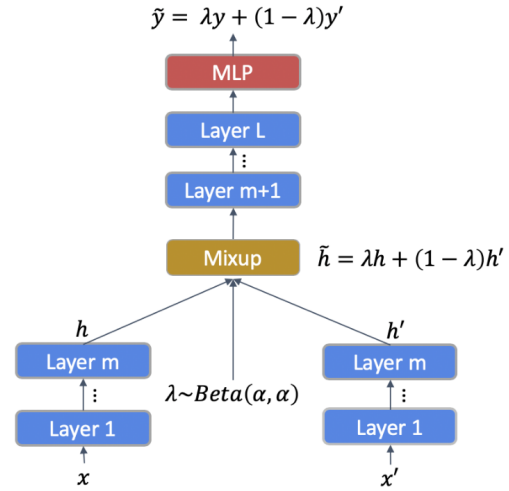
C Specificity Model

C.1 Model

CourseMirror data is also annotated for specificity. The data contains human annotations for around 7000 reflections[5] using the scheme introduced in (Fan et al., 2017). Table 4 shows the score distribution for CourseMirror specificity dataset. We use the data to train a specificity prediction model. We use the model to predict the specificity values for both CourseMirror and Amazon/Yelp datasets. The specificity prediction model (figure 2) uses DistilBERT encoder to produce reflection embedding, the embeddings are then used as features to train a logistic regression classifier. To keep the number of tuned parameters to minimum, the DistilBERT weights are frozen during the training process. The embeddings are used as fixed features, and all the training is performed on the logistic classifier side.

[5] https://petal-cs-pitt.github.io/data.html

(a) MixText For generative tasks.

(b) Original MixText.

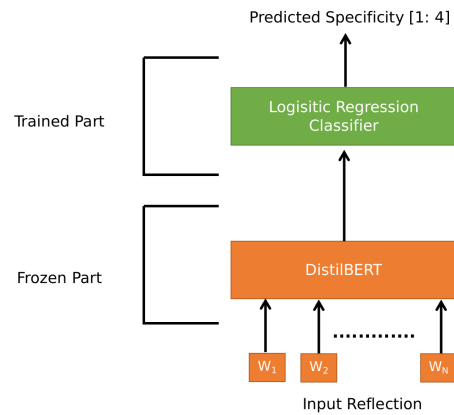Figure 1: Mixtext model and the modified MixGen for generative tasks.

Figure 2: Specificity prediction model used.

1	2	3	4
1354	2035	2377	1058

Table 4: CourseMirror Specificity dataset score distribution.

D   Data samples

D.1   CourseMirror (CM)

Table 5 shows an example of CM sample from CS
course.

D.2   Amazon/Yelp

Table 6 shows an example of sample from ama-
zon/Yelp data.

Prompt
Point of Interest (POI): Describe what you found most interesting in today's class.

Student Reflection Document
• the dynamic bag • I found the creation of the Bag to be the most interesting. • Learning about bags was very interesting. • Dr. Ramirez cleared up my understanding of how they should work. • I was really interested in learning all about an entirely new data structure , the Bag. • I 'm also noticing that as these classes get farther along , there is more focus on real world factors that determine strength of code like speed • The bag concept was cool how basically acts like a bag in real life with its usefulness. • Bags as a data type and how flexible they are. • Discussing the Assignment 1 • I found the examples and drawings the teacher drew on the whiteboard the most interesting. • Abstraction, though seemingly intimidating is kind of just giving programmers a break right? • We 're given so many more abilities and operations without having to know exactly how to code that. • That being said , while I understand the applications being explained to me , it 's hard to just manifest that on my own. • Learning about resizing Bags dynamically • The discussion of the underlying methods of ADTs such as bags was most interesting • the implementation of an array bag • Order does not matter when using a bag. • It is important to keep all of the values in an array together. • To do this , you should move an existing element into the vacant spot. • Looking at ADT 's from both perspectives • Information held in bags is not in any particular order • different ways to implement the bag • Thinking about a more general idea of coding with ADTs and starting to dig into data structures more specifically. • Code examples of key concepts/methods is always helpful. • I thought it was a good thing to go through the implementation of both the add ( ) and remove ( ) methods of the Bag ADT • Today we were talking about a certain type of ADT called a bag. • We talked about certain ways that we would implement the methods and certain special cases that we as programmers have to be aware of. • If you were removing items from ADT bag , you can simply shift the bottom or last item and put it in the place where we removed an item. • This is because , in bags , order does not matter. • Learning about managing arrays in a data structure • The bag ADT and how it is implemented

Reference Abstractive Summary
Students were interested in ADT Bag, and also its array implementation. Many recognized that it should be resizable, and that the underlying array organization should support that. Others saw that order does not matter in bags. Some thought methods that the bag provides were interesting.

Table 5: Sample data from the CourseMirror CS course.

Reviews
This pendant is so unique!! The design is beautiful and the bail is a ring instead of the typical bail which gives it a nice touch!! All the corners are smooth and my daughter loves it - looks great on her.I cannot say anything about the chain because used our own chain.:) Satisfied.
It look perfect in a womens neck!! great gift, I thought for the price it was going to look cheap, but I was far wrong. It look great.Spect great reward from your woman when you give this to her; D
The prettiest sterling silver piece I own now. I get so many compliments on this necklace. I bought it for myself from my hubby for Valentine's Day. Why not? When people ask where I got it, I simply say from my loving hubby. And he is off the hook as to what to get me. win + win.
I love hearts and I love 'love':) I do not have any negative feedback, the necklace is perfect and the charm is perfect. I just thought it would have been slightly bigger. Overall, I love my new heart necklace.
When I received the package, I was surprised and amazed because the necklace is so elegant, beautiful and the same as the picture shown here. I really love this necklace. It has a unique pendant designed. I will recommend it to someone to order it now...
Item is nice. Not a great quality item, but right for the price. Charm was larger than I expected (I expected small and elegant, but it was large and almost costume jewelry like). I think it is a good necklace, just not what I expected.
I got this as a present for my GF on Valintines day. She loves it and wears it every day! Its not cheap looking and it hasn't broken yet. The chain hasn't broken either even though it is very thin. Strongly recomend it!
Over all service has been great the only problem, I ordered a purple Mickey Mouse case for iPhone 4S they sent a black, n I felt it was to much trouble n such a small item to send back so needless to say its put back in a drawer somewhere
Abstractive Summary
This silver chain and pendant are elegant and unique. The necklace is very well made, making it a great buy for the cost, and is of high enough quality to be worn every day. The necklace looks beautiful when worn bringing many compliments. Overall, it is highly recommended.

Table 6: Sample data from the Amazon/Yelp data.