Exploring Multitask Learning for Low-Resource Abstractive Summarization

Ahmed Magooda, Mohamed Elaraby, Diane Litman
University of Pittsburgh
Pittsburgh, PA, USA
{aem132,mse30,dlitman}@pitt.edu

Abstract

This paper explores the effect of using multitask learning for abstractive summarization in the context of small training corpora. In particular, we incorporate four different tasks (extractive summarization, language modeling, concept detection, and paraphrase detection) both individually and in combination, with the goal of enhancing the target task of abstractive summarization via multitask learning. We show that for many task combinations, a model trained in a multitask setting outperforms a model trained only for abstractive summarization, with no additional summarization data introduced. Additionally, we do a comprehensive search and find that certain tasks (e.g. paraphrase detection) consistently benefit abstractive summarization, not only when combined with other tasks but also when using different architectures and training corpora.

1 Introduction

Recent work has shown that training text encoders using data from multiple tasks helps to produce an encoder that can be used in numerous downstream tasks with minimal fine-tuning (e.g., T5 (Raffel et al., 2019) and BART (Lewis et al., 2020)). However, in multitask learning for text summarization, it is still unclear what range of tasks can best support summarization, and most prior work has incorporated only one additional task during training (Isonuma et al., 2017; Chen et al., 2019; Pasunuru et al., 2017; Gehrmann et al., 2018). Also, to our knowledge, no prior work has tried to tackle multitask summarization in low-resource domains.

Our work attempts to address these gaps by answering the following research questions: *Q1) Can abstractive summarization performance be boosted via multitask learning when training from a small dataset? Q2) Are there some tasks that might be helpful and some that might be harmful for multitask abstractive summarization? Q3) Will the same findings emerge if a very different learning model is used or if pretraining is performed? Q4) Will the same findings emerge if a very different small training corpus is used?* To answer Q1, we use a pretrained BERT model (Devlin et al., 2019) within a multitask framework, and train all tasks using a small-sized corpus of student reflections (around 400 samples). To answer Q2, we explore the utility of training on four different tasks (both alone and in combination) in addition to abstractive summarization. To answer Q3, instead of fine-tuning with the BERT model, we perform experiments using the T5 transformer model (Raffel et al., 2019). To answer Q4, we replicate the student reflection experiments using two very different corpora (news and reviews). Our results show that abstractive summarization in low resource domains can be improved via multitask training. We also find that certain auxiliary tasks such as paraphrase detection consistently improve abstractive summarization performance across different models and datasets, while other auxilary tasks like language modeling more often degrade model performance.

2 Related Work

Multitask learning. Abstractive summarization has been enhanced in multitask learning frameworks with one additional task, by integrating it with text entailment generation (Pasunuru et al., 2017), extractive summarization (Chen et al., 2019; Hsu et al., 2018), and sentiment classification (Chan et al., 2020; Ma et al., 2018). While other research has combined multiple tasks, Lu et al. (2019) integrated only predictive tasks, while Guo et al. (2018) used only generative tasks. Recently, Dou and Neubig (2021) proposed using different tasks as guiding signals. However, the guiding signals can only be used one signal at a time with no easy way to combine them. In contrast, our work focuses on both generative and predictive tasks, explores task utility in isolation and in all combinations, and demonstrates generalization of

Data	# Docs	Train	Val	Test
CS	138	209	23	138
ENGR	52	286	32	52
S2015	88	254	28	88
S2016	92	250	28	92
CNN-5%	2500	1500	500	500
Amazon/Yelp	160	58	42	60

Table 1: Dataset summary.

findings across multiple models and corpora. Furthermore, aside from the two auxiliary tasks *(language modeling* (Magooda and Marcjan, 2020) and *extractive summarization* (Pasunuru et al., 2017)) that have been examined before in the context of multitask summarization, we introduce two new additional auxiliary tasks *(paraphrase detection, concept detection)(Section 4.1)*. Finally, while previous work relied on large training corpora (e.g. CNN/DailyMail (Hermann et al., 2015)), we target low resource domains and try to overcome data scarceness by using the same data to train multiple task modules.

Low resource training data. While most abstractive summarization work takes advantage of large corpora such as CNN/DailyMail, New York Times, PubMed, etc. to train models from scratch (Hermann et al., 2015; Nallapati et al., 2016; Cohan et al., 2018), recent work has also targeted low resource domains. Methods proposed to tackle little training data have included data synthesis (Parida and Motlicek, 2019; Magooda and Litman, 2020), few shot learning (Bražinskas et al., 2020), and pretraining (Yu et al., 2021). Our approach is different in that we use the same data multiple times in a multitask setting to boost performance.

3 Summarization Datasets

CourseMirror (CM)[1] is a student reflection dataset previously used to study both extractive (Luo and Litman, 2015) and abstractive (Magooda and Litman, 2020) summarization. The dataset consists of documents (i.e., a set of student responses to a reflective instructor prompt regarding a course lecture) and summaries from four course instantiations: CS, ENGR, S2015, and S2016.

CNN/DailyMail (CNN-5%) is a widely used summarization dataset consisting of around 300k news-oriented documents (Hermann et al., 2015). Since the focus of our research is low resource data, we randomly select 5% (500 documents) from the CNN/DailyMail test and validation sets. Then, to keep the CNN-5% data distribution similar to CM (3 courses for training, 1 for testing), we randomly sample 1500 documents for the training set.

Amazon/Yelp[2] is a dataset of opinions (Bražinskas et al., 2020) that is both small as well as similar to CourseMirror in that documents consist of multiple human comments where order doesn't matter. This dataset contains customer reviews from Amazon and Yelp of 160 products/businesses. For each of these, 8 reviews to be summarized are selected from the full set of reviews.

Table 1 summarizes each dataset in terms of the number of documents and their distribution into training, validation, and test sets. The PDF appendix contains examples from each dataset.

4 Proposed Models

This section describes the different tasks used for multitask learning with the intuition behind them, followed by the two summarization models used.

4.1 Auxiliary Tasks

Extractive summarization (E) aims to classify parts of a document (typically sentences) as either important (i.e. included in a summary) or not. It has been used as as an auxiliary abstractive summarization task (Chen et al., 2019; Hsu et al., 2018) as it can help the model focus on important sentences.

Concept detection (C) detects important concepts (keywords) within an input text. Humans can have a general understanding of a topic's main idea by looking through concepts or keywords (e.g., keywords integrated into early pages of research papers or books). Thus, we hypothesize that this task can help the model focus more on major keywords.

Paraphrase detection (P) aims to classify a pair of sentences as to whether they are conveying the same ideas using different wordings. We hypothesize that the relation between input documents and summaries can be viewed as a potential paraphrasing. We use the MSRP paraphrase dataset (Dolan and Brockett, 2005), in addition to summarization datasets, to train a paraphrase detection task.

Language modeling (L), in general, can help improve generative tasks. Training with LMs aims to skew the vocabulary slightly into the training data distribution.

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

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

4.2 BERT Multitask Integration[3]

We use a pretrained BERT (Devlin et al., 2019) model as a shared sequence encoder followed by a set of different task-specific modules (Figure 1). In the **single task** setting, only abstractive summarization is performed. In the **multitask** setting (integrating one or more auxiliary tasks), encoder weights are also fine-tuned alongside the rest of the model.

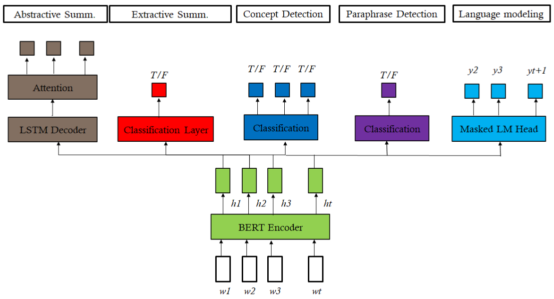


Figure 1: Proposed BERT-Multitask model.

Abstractive summarization (A). While recent work often uses transformers to overcome issues of sequence length (Qi et al., 2020), LSTM based decoders consistently outperform transformer-based ones when trained from scratch on our small CM dataset. Thus, we use LSTMs for our abstractive summarization (primary) task.

Extractive summarization (E): The model consists of a linear layer to classify a sentence as part of the summary or not. Document and input sentence are fed to BERT encoder in the format **[CLS] DW$_1$ DW$_2$...DW$_n$[SEP]SW$_1$ SW$_2$....SW$_m$**, where DW$_i$ is the i$^{\text{th}}$ word of the input document, SW$_i$ is the i$^{\text{th}}$ word of the sentence to classify, and ([CLS], [SEP]) are respectively the starting and separation tokens used by BERT.

Concept detection (C): The module's objective is to classify each word within a sequence as either a part of a concept or not. The module consists of a fully connected layer following the BERT encoder. We prepare the data by extracting concepts using a TF-IDF ranking algorithm (Thaker et al., 2019).

Paraphrase detection (P): The module consists of a fully connected layer classifier. Similar to extractive summarization, the input is passed to BERT in the format **[CLS]Sent$_1$[SEP]Sent$_2$**. Sent$_1$ and Sent$_2$ are the two input sentences for the MSRP dataset, and the input document and human summary for the summarization datasets.

Language modeling (L): The language modeling module consists of a masked language modeling (MLM) attention head, fine-tuned using the MLM objective. Following the original BERT training from Devlin et al. (2019), input tokens are masked with probability 15%, where masked tokens are either replaced by a special token (80%), random word (10%) or left unchanged (10%).

Model training: We train the model by training sub-modules consecutively. Thus, for each of the training epochs, we first train one of the submodules (e.g. abstractive) using the corresponding data batches, then we move to another submodule (e.g. extractive), and so on. Each submodule is trained with Maximum likelihood estimation (MLE). We perform training using multiple optimizers. The intuition is to tune different modules with different rates. We tune the whole model using 3 optimizers: one for the BERT encoder, another for the abstractive decoder, and the last for the other modules. All optimizers are Adam optimizers, with different initial learning rates 5e^{-4}, 5e^{-3}, and 5e^{-5} for BERT encoder, abstractive decoder, and other modules respectively. We also performed experiments using a single optimizer for the whole model. Multiple optimizers consistently outperform a single optimizer.

4.3 T5 Multitask Integration

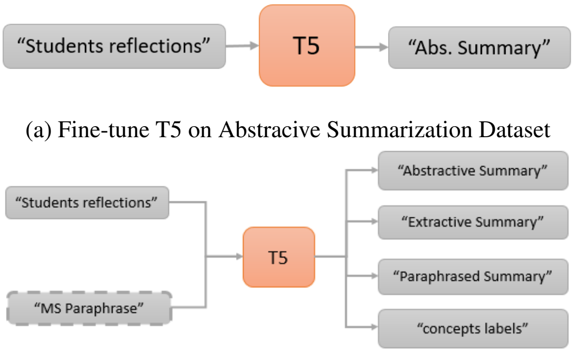
We also make use of the T5 (Raffel et al., 2019), which stores a large amount of knowledge about language and tasks. In the **single task** setting, we fine-tune a pretrained T5 on the abstractive task (**A**), using the low resource datasets.

In the **multitask** settings, we adopt the T5 framework to train the mixture of tasks as text-to-text, which allows us to fine-tune in the same model simultaneously. Figure 2 shows the settings used for training T5 model for both Single abstractive summarization task, and the multitask training with mixture of tasks. Since T5 is pretrained with CNN/DM, we don't perform experiments with CNN-5% using T5. Also note that unlike BERT, T5 represents any task as language modeling. Thus, we dropped the language modeling auxiliary task for T5, as it would be a form of redundancy.

5 Experiments, Results and Discussion

Our experiments evaluate performance using ROUGE (Lin, 2004) on F1. For CM data we report mean ROUGE using a leave-one-course-out vali-

[3] https://github.com/amagooda/MultiAbs.git



(a) Fine-tune T5 on Abstracive Summarization Dataset

(b) Fine-tune T5 on Mixture of tasks

Figure 2: Different fine-tuning conditions for T5. *(- -) indicates optional additive data for Paraphrasing.*

Tasks	R1	R2	RL
Single task (A)	26.82	4.71	21.5
A C	27.11	4.75	21.1
A E	28.51	4.91	21.41
A P	27.83	5.99	**23.05**
A L	27.22	5.47	21.31
A E L	28.36	5.62	21.6
A E P	27.68	5.24	21.81
A E C	27.41	5.81	22.13
A C P	**29**	**6.43**	22.2
A L P	27.71	5.82	21.14
A L C	27.39	6.09	21.36
ALL	27.72	5.55	21.31

Table 2: ROUGE results of *BERT* multitask on *CM*. Gray indicates multitask R is higher than single task score. **Boldface** indicates best R across tasks. *(Q1, Q2)*

dation[4], while for CNN-5% and Amazon-Yelp we report ROUGE using held-out test sets.

Q1: The gray cells in Table 2 show that BERT multitask training for CM data can help improve single-task (A) training. For R1 and R2 we observe improvements across *all* task combinations. While some task combinations also improve RL ((A P), (A E L), (A E P), (A E C), (A C P)), others degrade performance, particularly when language modeling is involved (e.g., (A L), (A L P), (A L C), and (ALL)). Thus, while multitask training can be effective, we need to further explore task choice.

Q2: Prior work showed the utility of extractive summarization (Hsu et al., 2018) and language models (Magooda and Marcjan, 2020) as auxiliary summarization tasks, and we too observe similar behavior for R1 and R2 in Table 2. For RL, however, (A E) and (A L) failed to improve the score. Similarly, our new concept task (A C) improves R1 and R2 but not RL. On the other hand, integrating our proposed paraphrasing task (A P) improves performance for all ROUGE scores. When we integrate two auxiliary tasks, (A E L), (A E P), (A E C), and (A C P) improve all of R1, R2 and RL compared to single task performance. For RL, it seems that adding E with another auxiliary task rather than in isolation improves performance. Also, the (A C P) combination which uses our two proposed tasks (concept, paraphrasing) achieves the best R1, R2, RL in the 3-task setting.

Tasks	R1	R2	RL
Single Task (A)	36.08	10.94	31.57
A E	29.99	8.80	24.80
A C	35.46	10.76	30.81
A P	**36.75**	**12.13**	**32.30**
A C P	36.28	11.59	31.58
A E C	29.19	8.69	25.20
ALL	30.31	9.60	27.97

Table 3: ROUGE results of *T5* (No language modeling auxiliary task) fine-tuned on *CM*. *(Q3)*

Q3: Table 3 shows that some of the CM findings obtained using BERT multitask are similar when a different model such as T5 is used for CM. Similar to BERT, incorporating paraphrasing into T5 helps improve all ROUGE scores when used as a single auxiliary task (A P) and in combination with the concept task (A C P). On the other hand, the utility of (A E C) didn't transfer from BERT to T5.

Q4: Shifting gears from changing the model to changing the data, Table 4 shows that when BERT multitask is applied to CNN-5%,[5] there is now no task configuration that leads to improvement across all of R1, R2, and RL. However, the majority of combinations (6 of 11) improved two out of the three ROUGE scores, especially R2 and RL. Additionally, judging by ROUGE scores of certain combinations such as (A C) and (A P), we can see that the reduction in R1 (0.38, 0.47) is less than the improvements gained in R2 (0.39, 0.61) and far less than RL (2.05, 1.46) respectively. Thus, we can argue that paraphrasing auxiliary task tends to be very helpful either across different data or different models. To further verify the utility of paraphrasing across datasets, we also evaluated the T5 model[6] on the Amazon/Yelp dataset. However, due to the lack of extractive annotation for Amazon/Yelp, we only examine (A P), the best performing T5 combination for CM (Table 3). Table 5 shows that indeed

[4] Individual course ROUGE scores are in the Appendix.

[5] Recall from Section 4.3 that T5 is not used for CNN-5%.

[6] We only examined T5 since for CM, the T5 ROUGE scores (Table 3) were higher than when using BERT (Table 2).

Tasks	R1	R2	RL
Single Task (A)	13.3	0.73	8.98
A C	12.92	1.12	11.03
A E	12.9	0.33	8.76
A P	12.83	**1.34**	10.44
A L	13.43	0.65	8.36
A E L	**14.18**	0.36	10.1
A E P	12.82	0.64	8.53
A E C	11.52	1.05	**11.23**
A C P	11.08	1.09	10.95
A L P	12.79	0.53	8.94
A L C	10.35	0.09	9.81
ALL	11.15	1.26	10.49

Table 4: ROUGE results of *BERT* on *CNN-5%. (Q4)*

Tasks	R1	R2	RL
Bražinskas et al.	36.25	9	22.36
Single task (A)	34	8.8	21.25
A P	**34.1**	**9.1**	**21.7**

Table 5: ROUGE results of *T5* fine-tuned with paraphrasing on *Amazon/Yelp. (Q4)*

paraphrasing is again helpful as an auxiliary task, as it improves all ROUGE scores for Amazon/Yelp.

Finally, while the objective of our work is to explore the utility of auxiliary tasks across models and data, rather than to outperform the prior SOTA, we briefly compare our results to prior work where possible. For CM, multiple task combinations outperform the data synthesis method (CM + synthetic) from Magooda and Litman (2020) on R2 and RL. For example, while (A C P) yielded 0.63 less R1, it had 0.98 and 1.52 higher R2 and RL, respectively. For Amazon/Yelp, while our approach increases R2 by 0.1 compared to Bražinskas et al. (2020), the R1 and RL scores are lower by 2.15 and .66, respectively. These results show that there is still room for improvement, particularly for R1, and suggest a future combination of our approach with such alternative low-resource methods.

6 Conclusion and Future Work

We explored the utility of multitask training for abstractive summarization, using three low resource datasets (CM, CNN-5%, Amazon/Yelp) and two fundamentally different models (BERT, T5) with different preconditions (i.e. BERT not pretrained with summarization dataset versus T5 pretrained with CNN dataset) to verify any observed behavior. We also integrated four different auxiliary tasks, in isolation and together. We conducted several experiments to find if training a multitask model, in general, is helpful, or if some tasks might introduce degradation in model performance. We showed that indeed some tasks might help improve ROUGE scores and some might not help, at least when trained in a low resource setting. We found that among all task combinations, **(Abstractive + Paraphrase detection)** improved almost all ROUGE scores across different datasets (CM , Amazon/Yelp, and CNN-5%) and different models (BERT, and T5), with **(Abstractive + Concept detection + Paraphrase detection)** as another good candidate. We also found that paraphrasing and concept detection, which had not been previously examined as auxiliary abstractive summarization tasks, can be helpful for low resource data. In the future, we plan to continue exploring the generality of our findings, by include new types of low resource data (e.g. discussions, emails), BART as one of the SOTA models, and new auxiliary tasks. We also plan to combine multitask learning with other low resource methods (e.g., data synthesis).

Acknowledgements

The research reported here was supported, in whole or in part, by the institute of Education Sciences, U.S. Department of Education, through Grant R305A180477 to the University of Pittsburgh. The opinions expressed are those of the authors and do not represent the views of the institute or the U.S. Department of Education. We like to thank Khushboo Thaker for helping generating concepts, the Pitt PETAL group and the anonymous reviewers for advice in improving this paper.

References

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.

Hou Pong Chan, Wang Chen, and Irwin King. 2020. A unified dual-view model for review summarization and sentiment classification with inconsistency loss. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 1191–1200. ACM.

Yangbin Chen, Yun Ma, Xudong Mao, and Qing Li. 2019. Multi-task learning for abstractive and extractive summarization. *Data Science and Engineering*, 4(1):14–23.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

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.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Zi-Yi Dou and Graham Neubig. 2021. Word alignment by fine-tuning embeddings on parallel corpora. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2112–2128, Online. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft layer-specific multi-task summarization with entailment and question generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 687–697, Melbourne, Australia. Association for Computational Linguistics.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 132–141, Melbourne, Australia. Association for Computational Linguistics.

Masaru Isonuma, Toru Fujino, Junichiro Mori, Yutaka Matsuo, and Ichiro Sakata. 2017. Extractive summarization using multi-task learning with document classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2101–2110, Copenhagen, Denmark. 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 pre-training 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.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yao Lu, Linqing Liu, Zhile Jiang, Min Yang, and Randy Goebel. 2019. A multi-task learning framework for abstractive text summarization. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 9987–9988. AAAI Press.

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.

Shuming Ma, Xu Sun, Junyang Lin, and Xuancheng Ren. 2018. A hierarchical end-to-end model for jointly improving text summarization and sentiment classification. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4251–4257. ijcai.org.

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*.

Ahmed Magooda and Cezary Marcjan. 2020. Attend to the beginning: A study on using bidirectional attention for extractive summarization. *ArXiv preprint*, abs/2002.03405.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

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.

Ramakanth Pasunuru, Han Guo, and Mohit Bansal. 2017. Towards improving abstractive summarization via entailment generation. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 27–32, Copenhagen, Denmark. Association for Computational Linguistics.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv preprint*, abs/1910.10683.

Khushboo Thaker, Peter Brusilovsky, and Daqing He. 2019. Student modeling with automatic knowledge component extraction for adaptive textbooks. In *iTextbooks@ AIED*, pages 95–102.

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.

A BERT parameters

In our BERT experiments we use the BERT basic uncased model which consists of 12 layers, and a hidden size of 768. We fine-tune the model using a single Nvidia P100 GPU for 85 epoch and a batch size of 4 and 8. The epoch with the highest ROUGE score on the validation set is used later for testing. We tried multiple initial learning rates, as different learning rates might be selected for different courses depending on the validation set performance. The multitask training is done in a sequential fashion, where during each epoch all tasks are trained sequentially (i.e. for each epoch, the abstractive sub-model is trained using all data, followed by the extractive sub-model, etc..). We use a maximum input length of (120, 200, and 250) tokens for CM experiments as the average document length of CM data is around 200 tokens, then used the most suitable length based on the validation set. We tried multiple max input lengths for CM as we noticed that there are repeated sentences within the reflections. So while smaller cut-offs like 120 can truncate some of the reflections (which can be repeated), it would lead to a faster training process. As for CNN-5% we use a maximum of 500 (max is 512 for BERT). Shorter documents are padded and longer ones are truncated. We generate summaries using beam search with beams of length 5. The average length of CM summaries ranges from 35 to 42 tokens, and 56 for CNN. Thus we decided to limit the summary length to 50 tokens.

B T5 parameters

We use the *3B T5* model, which is publicly available. The model consists of 24 layers for encoder and decoder. We set the initial learning rate to 0.001, which the authors used in their summarization experiments. Due to the lack of hardware, we couldn't perform *Beam Search* decoding. We fine-tuned the course mirror data on 7 TPUs on Google Cloud for 5000 steps.

C Data samples

C.1 CourseMirror (CM)

Table 6 shows an example of CM sample from CS course.

C.2 Amazon/Yelp

Table 7 shows an example of sample from amazon/Yelp data.

D Full Results

Full results for BERT and T5 multitask models on CM data are shown in tables (8, and 9).

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 you 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.
Reference Extractive Summary
• Bags as a data type and how flexible they are. • Thinking about a more general idea of coding with ADTs and starting to dig into data structures more specifically. • I thought it was a good thing to go through the implementation of both the add() and remove() methods of the Bag ADT. • Learning about managing arrays in a data structure. • Information held in bags is not in any particular order.

Table 6: 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 7: Sample data from the Amazon/Yelp data.

Tasks	R1	R2	RL	AVG	Δ	R1	R2	RL	AVG	Δ	Row
	CS0445					**ENGR**					
Single Task (A)	26.93	3.98	21.04	17.32	*	27.19	7.27	22.66	19.04	*	1
A C	27.09	4.85	20.12	17.35	+	30.14	7.67	22.96	20.26	+	2
A E	25.62	5.04	19.9	16.85	-	31.75	4.69	22.77	19.74	+	3
A P	**28.13**	**7.13**	**23.45**	**19.57**	**+**	**28.56**	**7.29**	**23.99**	**19.95**	**+**	**4**
A L	25.53	4.69	21.48	17.23	-	30.04	7.36	24.27	20.56	+	5
A E L	28.18	6.48	21.34	18.67	+	33.75	8.64	26.86	23.08	+	6
A E P	28.18	2.68	20.21	17.02	-	27.4	8.72	25.33	20.48	+	7
A E C	**27.4**	**6.58**	**21.36**	**18.45**	**+**	**28.87**	**8.95**	**24.33**	**20.72**	**+**	**8**
A C P	**28.18**	**5.21**	**20.67**	**18.02**	**+**	**30.37**	**10.84**	**26.78**	**22.66**	**+**	**9**
A L P	25.99	4.87	20.15	17	-	28.57	10.15	21.74	20.15	+	10
A L C	32.15	5.42	21.99	19.85	+	25.81	7.66	21.51	18.33	-	11
ALL	**28.34**	**3.89**	**22.79**	**18.34**	**+**	**28.54**	**6.64**	**25.7**	**20.29**	**+**	**12**
	S2015					**S2016**					
Single Task (A)	27.71	4.83	19.4	17.31	*	25.46	2.76	22.93	17.05	*	13
A C	21.92	3.11	17.75	14.26	-	29.32	3.4	23.6	18.77	+	14
A E	27.99	5.07	20.97	18.01	+	28.7	4.87	22	18.52	+	15
A P	**28.6**	**4.84**	**22.33**	**18.59**	**+**	**26.03**	**4.7**	**22.43**	**17.72**	**+**	**16**
A L	26.12	4.43	18.37	16.31	-	27.22	5.4	21.14	17.92	+	17
A E L	23.44	4.35	18.72	15.5	-	28.09	3.01	19.51	16.87	-	18
A E P	26.91	4.85	21.47	17.74	+	28.26	4.72	20.25	17.74	+	19
A E C	**26.43**	**4.45**	**21.62**	**17.5**	**+**	**26.94**	**3.27**	**21.24**	**17.15**	**+**	**20**
A C P	**28.04**	**5.59**	**21.15**	**18.26**	**+**	**29.67**	**4.11**	**20.23**	**18**	**+**	**21**
A L P	26.27	4.69	19.55	16.84	-	30.04	3.59	23.13	18.92	+	22
A L C	26.78	7.46	20.62	18.29	+	24.84	3.84	21.33	16.67	-	23
ALL	**25.71**	**6.39**	**21.31**	**17.8**	**+**	**28.31**	**5.3**	**21.89**	**18.5**	**+**	**24**

Table 8: Full ROUGE results of BERT multitask model. Δ represents the change direction relative to the abstractive only model, where '+' means higher average ROUGE, and '-' otherwise. **Boldface** indicates improving scores across all courses.

Tasks	R1	R2	RL	AVG	Δ	R1	R2	RL	AVG	Δ	
	CS0445					**ENGR**					
Single Task (Abs.)	34.62	9.46	29.84	24.64	*	35.43	9.93	31.07	25.47	*	1
A E	30.01	8.21	22.92	20.38	-	32.04	8.11	27.10	22.41	-	2
A C	34.42	9.71	29.31	24.48	-	35.84	10.14	31.38	25.78	+	3
A P	34.56	**9.81**	**30.11**	**24.82**	**+**	**36.79**	**12.64**	**32.62**	**27.35**	**+**	**4**
A C P	34.70	9.47	30.2	27.79	+	36.16	11.46	31.74	26.45	+	5
A E C	27.43	7.54	24.63	19.86	-	29.41	7.63	26.15	21.06	-	6
ALL	28.34	8.31	26.72	21.12	-	30.11	8.45	28.98	22.51	-	7
	S2015					**S2016**					
Single Task (Abs.)	36.87	12.03	32.34	27.08	*	37.41	12.33	33.02	27.58	*	12
A E	27.65	7.96	22.74	19.45	-	30.25	10.93	26.45	22.54	-	13
A C	34.49	10.40	30.12	25	-	37.09	12.77	32.42	27.42	-	14
A P	36.78	**12.64**	**32.62**	**27.34**	**+**	**38.86**	**13.41**	**33.84**	**28.71**	**+**	**15**
A C P	35.63	11.14	30.85	25.87	-	38.64	14.27	33.52	28.81	+	16
A E C	28.25	7.97	23.15	19.79	-	31.65	11.60	26.86	23.37	-	17
ALL	31.21	10.66	28.99	23.62	-	31.57	10.99	27.20	23.25	-	18

Table 9: Full ROUGE results of T5 Model fine-tuned on CM data under several experimentation settings