Techniques to Enhance Abstractive Summarization Model Training for Low Resource Domains

by

Ahmed Magooda

B.S in Computer Engineering, Cairo University, 2011

M.S. in Computer Engineering, Cairo University, 2016

Submitted to the Graduate Faculty of

the Dietrich School of Arts and Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2022

UNIVERSITY OF PITTSBURGH

DIETRICH SCHOOL OF ARTS AND SCIENCES

This dissertation was presented

by

Ahmed Magooda

It was defended on

March 4, 2022

and approved by

Diane Litman, Department of Computer Science

Adriana Kovashka, Department of Computer Science

Milos Hauskrecht, Department of Computer Science

He Daqing, Department of Informatics and Networked Systems

Techniques to Enhance Abstractive Summarization Model Training for Low Resource Domains

Ahmed Magooda, PhD

University of Pittsburgh, 2022

Nowadays, the amount of information is growing exponentially, and it is challenging to digest even the information for a particular topic. Summarization can reduce the information into a handful of paragraphs, helping human readers digest information easier. Automatic summarization spans different techniques (abstractive, extractive, phrase extractive, etc.). Abstractive summarization specially aims to mimic how humans summarize, as it aims to summarize a large amount of text into a readable, comprehensive summary. Abstractive summarization has benefited from recent advances in Machine learning and Natural Language Processing. However, the majority of prior studies focus on data-rich domains, where large datasets are available. On the other hand, very few studies focus on data scarce domains. A typical practical issue that is rendered in such domains is model overfitting. Training complex models using a few samples can easily lead to overfitting. As a step towards remedying these shortcomings, this thesis aims to enhance abstractive summarization models in low-resource settings by tackling three challenges.

1-Can we adapt widely used data augmentation/synthesis techniques to abstractive summarization to remedy the scarceness issue?

2- How can we benefit from domain transfer or pretraining, and what can be a helpful strategy to do it more efficiently?

3- Can we extract additional information from the data and to use it more effectively?

This thesis first proposes new data synthesis (augmentation) models, novel techniques to synthesize new data for model training. We then introduced a variant of a recent data augmentation technique to be used in generative tasks. Additionally, we explored the utility of using curriculum learning to both improve pretraining and fine tuning processes. Finally, to overcome the third challenge, we propose integrating the summarization model into a

multitask learning setting. We also show that some auxiliary tasks can consistently improve abstractive summarization in a low resource setting. We finally combine multitask learning and data augmentation to observe if the combination would be more helpful than each approach in isolation. We ultimately showed that combining more than one technique can introduce some improvements compared to a single technique. However, overall, using techniques in isolation leads to more consistent improvements.

Table of Contents

[1]We refer to CourseMirror as CM for short
[2]We refer to CourseMirror as CM for short
[3]We refer to Amazon/Yelp as AY for short

List of Tables

List of Figures

Preface

First and foremost, I would like to praise and thank God, who has granted me countless blessings and the ability to finish this thesis.

I would then like to especially thank my advisor, Prof.Diane Litman. You are one of the best persons I have worked with on both managerial and personal levels, and I have learned a lot from you. I appreciate your help, dedication, and encouragement and that you gave me total flexibility in work.

I then want to thank my wife, Shahd. You have been supporting me during my journey for six years. You left your parents and friends to move out to the US with me. I appreciate you had to start from scratch to get your certificate. I know these past six years were not easy, so, Thank you.

Finally, I would like to thank my friends Amr, Salem, Yassin, and Zaghloul, who made my time here in Pittsburgh full of fun. I also would like to thank my lab-mates, colleagues, and committee for their help and supportive feedback.

1.0 Introduction

Nowadays, the amount of information available online is increasing exponentially. It is becoming challenging to digest even the information available for a particular topic/field. *Text summarization* is a helpful technique that can help with such cases. Summarization can reduce large pieces of text into a handful of paragraphs, assisting human readers to digest much information in a few lines. However, this is just the bright side of the story. Getting summaries for a large amount of text is a labor-intensive process, which requires humans to read the full documents before generating a summary. Reading documents would lead us back to our original problem. Luckily automatic machine summarization can help to solve the problem. Machine summarization techniques can be classified into two main categories; 1- Abstractive summarization techniques, and 2- Extractive summarization techniques. Extractive summarization, which aims to extract salient pieces of text, has seen great performance strides during the last years. On the other hand, abstractive summarization, which aims to summarize input text into a readable and comprehensive summary for users to read easily, is still lacking behind. However, abstractive summarization, in turn, has gained much attention due to recent advances in Machine learning and Natural Language Processing (NLP). Performing the task of abstractive summarization typically requires two steps; 1- Reading the input and storing the knowledge in some format, 2- Using the knowledge, producing a condensed version with the same or new vocabulary. In earlier stages, researchers used to read and present the knowledge with the aid of graphs or templates, then later on traverse these graphs to generate summaries or fill these templates [22, 24, 71]. Later on, neural networks, especially recurrent based ones (RNN, LSTM, GRU, Transformers, etc.), helped represent the knowledge in a format of embedding that can be later used to generate a summary [7, 23, 86, 74]. However, the majority of prior studies focus on data-rich domains. Large datasets allow researchers to develop and train complicated and robust models. On the other hand, very few studies focus on developing adequate models for scarce data domains or improving existing models to perform better in such cases.

In this thesis, we try to fill a few gaps that we think are still underexplored in the

domain of abstractive summarization. While many works have focused on abstractive summarization, few of that work focuses on abstractive summarization solutions for low resource domains. That is a domain in which providing human-annotated data is a costly and time-consuming process; thus, these domains tend to have small-sized (low resource) data (i.e., hundreds of samples) used for training machine learning models. Additionally, very little research explored the data augmentation solution, a technique widely used for computer vision and other NLP tasks such as machine translation. Similar to the work on the data side, the work on the auxiliary tasks side is still underexplored. While numerous research explored integrating additional auxiliary tasks, no prior work further explored which tasks are especially useful in low resource settings. Additionally, no prior work combined two paradigms of auxiliary tasks (generative and predictive).

With that said, in this thesis, we decided to pursue ***two different directions and finally combine both of them into a final collective approach***. The ***first direction (data-based direction)*** we pursue focuses on data manipulation, where we try to use data differently regardless of the model. The work in this direction can be easily integrated into different models with minimal or no changes to the model itself. In this direction, we explore different techniques such as data augmentation and domain transfer. Training abstractive summarization models typically require large amounts of data, which can be a limitation for many domains. In this thesis, we explore using domain transfer and data synthesis to improve the performance of recent abstractive summarization methods when applied to small corpora of student reflections. First, we investigated whether tuning state of the art models trained on newspaper data could boost performance on student reflection data. Evaluations demonstrated that summaries produced by tuned models achieved higher ROUGE scores than models trained on just student reflection data or just newspaper data. Tuned models also achieved higher scores than extractive summarization baselines, and additionally were also judged to produce more coherent and readable summaries in human evaluations. Second, we explored whether synthesizing summaries of student data could additionally boost performance. We proposed a template-based model to synthesize new data, which further increased ROUGE scores when incorporated into training. Finally, we showed that combining data synthesis with domain transfer achieved higher ROUGE scores than only one of the

two approaches. Furthermore, we explore enhancing both the data augmentation and the training process by first improving the template-based augmentation model and introducing a different data augmentation approach that works on another part of the training pipeline through mixing multiple training samples. Second, we explore improving the training process by using the training data more effectively through a curriculum.

The ***second direction (model-based direction)*** we pursue, on the other hand, focuses on improving the model itself rather than the data. In this direction, we introduce making changes to the model to incorporate multitask learning. We explore 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, intending to enhance 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 specific tasks (e.g., paraphrase detection) consistently benefit abstractive summarization when combined with other tasks across different architectures, different training corpora, or both.

This thesis is organized as follows. Chapter 2 reviews the different datasets we use across our work, especially the CourseMirror student reflection dataset, which is the main low resource data we use in all of our experiments. Chapter 3 reviews prior work in our contribution areas, particularly: automatic text abstractive summarization, data augmentation in NLP and text summarization specifically, curriculum learning, and finally multitask learning for text summarization.

In the following four chapters, we discuss moving from data-based solutions to model-based ones, to combine them finally. Chapter 4 explores simple domain transfer and introduces a new template-based model for data synthesis. Chapter 5 shows how to improve the domain transfer technique by integrating curriculum learning. Additionally, it introduces a different data augmentation technique that we use in addition to the template-based model. Chapter 6 then switches gears towards model-based solutions. The chapter introduces training the abstractive summarization model in a multitask setting to make use of the training

data multiple times. The chapter also explores different auxiliary tasks and shows which tasks are specifically helpful in the low resources setting. Chapter 7 explores combining all the proposed solutions and finally reports how successfully these solutions can be integrated simultaneously.

1.1 Contributions

We can summarize the thesis contribution into 4 points:

- Shed light on the utility of data augmentation/synthesis to remedy data scarceness in the domain of abstractive text summarization and introduce novel template-based and paraphrasing-based synthesis models.
- Proposing integrating curriculum learning in model training to improve the outcome of the training process.
- Exploring improving abstractive summarization through multitask learning and the utility of using the same data multiple times to train the different submodels. Additionally, doing a comprehensive search across four different auxiliary tasks (Extractive summarization, Language Modeling, Paraphrase Detection, Concept Detection) of two different categories (Generative and predictive). Finally, showing that auxiliary tasks like paraphrase detection can be consistently helpful for the abstractive summarization task in a low resource setting. We also released the code online for other researchers to use it.[1]
- Combining the data augmentation, curriculum learning, and multitask learning in a single model and comparing it with prior approaches.

[1] https://github.com/amagooda/MultiAbs

2.0 Datasets

In this chapter, we introduce the datasets we use in our research. For each dataset, we first describe the dataset and then discuss the reason for including it alongside how and in which part we incorporate it in our research.

2.1 CourseMirror (CM)[1] Summarization

2.1.1 Description

Student reflections are comments provided by students in response to a set of instructor prompts. The prompts are directed towards gathering students' feedback on course material. Student reflections are collected directly following each of a set of classroom lectures over a semester. The set of reflections for each prompt in each lecture is considered a student reflection document. Our work's objective is to provide a comprehensive and meaningful abstractive summary of each student's reflection document. Our dataset consists of documents and summaries from four-course instantiations [52, 54, 51]:

- ENGR (Introduction to Materials Science and Engineering)
- ST15 and ST16 (Statistics for Industrial Engineers, taught in 2015 and 2016, respectively)
- CS (Data Structures in Computer Science)

All reflections were collected in response to two pedagogically-motivated prompts [64]:

- Point of Interest (POI): "Describe what you found most interesting in today's class"
- Muddiest Point (MP): "Describe what was confusing or needed more detail."

For each reflection document, at least one human (either a TA or domain expert) created three different types of summaries (Abstractive, Extractive, Phrase). The abstractive summary is a comprehensive paragraph that summarizes the major points in the reflections. The

[1] We refer to CourseMirror as CM for short

extractive summary is the five most representative sentences selected from the reflections. Finally, the phrase summary is a set of five most representative phrases either selected from the reflections or written in the annotator's own wordings. ***A typical training sample of the CourseMirror dataset is a set of reflections accompanied by one of the human annotations. Thus, if two human annotators annotate a set of reflections, this would sum to two training samples.***

CourseMirror
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.

Reference Phrase Summary

• The dynamic, resizable bag.

- Abstract data types (ADT)
- Methods of the bag.
- Arrays in the bag implementation.
- Order does not matter in bags.

Table 1: Sample data from the CourseMirror CS course.

Course	Prompt	Num Lectures	Reflections			Number of Annotators
			Avg	Min	Max	
CS	POI	23	25	15	33	3
	MP		26	14	37	
ENGR	POI	26	65	29	112	1
	MP		65	29	111	
ST15	POI	22	41	24	56	2
	MP		41	24	55	
ST16	POI	23	44	23	79	2
	MP		44	23	65	

Table 2: CourseMirror dataset summary.

Table 1 shows an example of both extractive and abstractive reference summaries produced by one annotator for the CS course (refer to appendix A for instructions provided to annotators). Abstractive reference summary is what most of this work focuses on, as it is used to train different abstractive summarization models. Additionally, extractive reference summaries are used for experiments involving extractive summarization either as a baseline (Chapter 4) or as auxiliary task (Chapter 6). Table 2 summarizes the dataset in terms of the number of lectures, the type of prompts per lecture, the average number of reflections per prompt, and the number of human summaries for each set of reflections. Table 3 shows

Course	Prompt	Number of tokens								
		Abstractive			Extractive			Phrase		
		Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
CS	POI	39.54	18	69	48.54	17	92	15.69	5	31
	MP	44.43	23	80	47.3	14	91	15.98	5	42
ENGR	POI	36.5	17	52	38	18	110	13	5	22
	MP	38.5	27	76	46.84	17	140	13.23	6	24
ST15	POI	33.79	13	50	31.27	9	93	14.7	5	37
	MP	42	18	77	32.47	10	77	16.6	8	35
ST16	POI	40	30	57	16.52	9	39	14.7	7	30
	MP	44.62	31	59	18.39	8	44	14.45	6	24
Course	Prompt	Abstractive			Extractive			Phrase		
		Num Sentences			Num Sentences			Num Phrases		
		Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
CS	POI	2.4	1	5	4.9	3	5	4.91	3	5
	MP	2.8	2	5	5	5	5	5	5	5
ENGR	POI	2.2	1	3	4.85	3	6	4.77	3	5
	MP	2.4	2	5	5.1	5	6	5	5	5
ST15	POI	1.9	1	3	4.87	2	5	4.87	2	5
	MP	2.4	1	5	4.98	4	5	5	5	5
ST16	POI	2.5	2	4	5	5	5	5	5	5
	MP	2.8	2	3	5	5	5	5	5	5

Table 3: Human summary lengths across courses.

			Inter-Annotator ROUGE			
Abstractive	Course	Annotators	R1	R2	RL	Avg
	CS	A1-A2	39.91	11.75	37.83	29.83
		A1-A3	41.22	8.81	36.86	28.96
		A2-A3	40.07	9.21	36.17	28.48
		Max	39.91	11.75	37.83	29.83
		Min	40.07	9.21	36.17	28.48
		Avg	40.4	9.92	36.95	29.09
	ST15	A1-A2	41.18	14.75	37.41	31.11
	ST16	A1-A2	43.96	14.96	41.33	33.42
Extractive	Course	Annotators	R1	R2	RL	Avg
	CS	A1-A2	53.22	36.65	48.69	46.19
		A1-A3	51.24	31.68	44.55	42.49
		A2-A3	46.86	28.39	42.93	39.39
		Max	53.22	36.65	48.69	46.19
		Min	46.86	28.39	42.93	39.39
		Avg	50.44	32.24	45.39	42.69
	ST15	A1-A2	45.24	24	39.76	36.33
	ST16	A1-A2	57.77	36.71	52.11	48.86
Phrase	Course	Annotators	R1	R2	RL	Avg
	CS	A1-A2	40.91	13.78	34.92	29.87
		A1-A3	42.82	15.29	37.8	31.97
		A2-A3	37.48	10.48	31.54	26.5
		Max	42.82	15.29	37.8	31.97
		Min	37.48	10.48	31.54	26.5
		Avg	40.4	13.18	34.75	29.44
	ST15	A1-A2	45.17	21.35	39.87	35.46
	ST16	A1-A2	50.11	22.82	42.54	38.49

Table 4: CourseMirror summary analysis.

an analysis of the different human summary lengths across the different courses and different prompts, which would provide an estimation of how many tokens/sentences summarization models would produce. Table 4 shows the Inter-Annotator ROUGE (sec 3.7) (IAR) scores for the different summary types (Abstractive, Extractive, and Phrase) across the different courses. IAR provides an idea of the degree of similarity we can expect between human annotators, thus providing comparable scores to judge how automatic models perform. We calculate ROUGE between every two annotators. In table 4 we show the ROUGE scores between every combination of two annotators alongside the maximum, minimum, and average of these combinations. We report the IAR for three out of the four courses (CS, ST15, and ST16), respectively. IAR is not reported for ENGR as one human annotator only annotated it. From table 4 we can argue that ROUGE scores in the range of (40∼44, 10∼15, and 37∼41) for (R1, R2, and RL) respectively are in the range of human agreement and can be considered good performance for automatic abstractive summarization models.

2.1.2 Usage

Configuration	**Data**	**# Docs**	**Train**	**Val**	**Test**
Leave One Course Out (LOCO)	CS	138 = 23x2x3	209	23	138
	ENGR	52 = 26x2x1	286	32	52
	S2015	88 = 22x2x2	254	28	88
	S2016	92 = 24x2x2	250	28	92
Train/Val/Test	Full CM	370	296	37	37

Table 5: CourseMirror dataset split summary (#Docs = num of lectures x num of prompts x num of annotators)

CourseMirror (CM) is the primary dataset used in all of our experiments (Synthesis, curriculum, and multitask). CourseMirror (CM) is a small-sized dataset (i.e., hundreds of samples); this renders it a right candidate for our research focus of enhancing summarization performance in low resources domains. We use CourseMirror data to train and fine tune dif-

ferent summarization models. Additionally, we use it as a part of the training data used to train the rewriting model (part of the synthesis model introduced in chapter 4). We perform experiments with CM data using two configurations Leave-One-Course-Out (LOCO) and the traditional Train/Val/Test splits. For Leave-One-Course-Out, we essentially perform training and validation using the collective data of three out of the four courses and then perform testing using data from the last course. On the other hand, for Train/Validation/Test configuration, we compiled data from all courses into one dataset. We then split the data into training, validation, and test sets (80%, 10%, 10%, respectively) by sampling equally from all courses. Table 5 shows the different data split sizes using the two different configurations.

2.2 CourseMirror (CM)[2] Specificity/Quality

2.2.1 Description

In another line of work that focused more on analyzing reflection quality, Fan et al. [21] and Luo and Litman [53] collected a variant of the CourseMirror data intending to research reflection specificity/quality prediction. The objective is to predict the specificity of student reflections. The predicted values map to how specific reflections are, where higher values correspond to more specific reflections, and lower values correspond to more vague ones. In this work, we use the data collected by Fan et al. [21] and Luo and Litman [53] to train a specificity prediction model that is later used for curriculum learning experiments. The data consists of 4 courses:

- ENGR (Introduction to Materials Science and Engineering)
- ENGR2010 (Old version of Introduction to Materials Science and Engineering, taught in 2010)
- ST15 (Statistics for Industrial Engineers, taught in 2015)
- Chemistry

[2]We refer to CourseMirror as CM for short

Course	Lectures	Reflections	Annotation	Avg Reflections/Prompt
ENGR	28	3626	1-4	122
ENGR2010	4	395	1-4	41
Chemistry	23	1034	1-4	50
ST2015	22	1769	1-4	77

Table 6: CourseMirror specificity dataset summary.

Course	1	2	3	4
ENGR	427	942	1658	599
ENGR2010	239	60	47	49
Chemistry	298	259	250	227
ST2015	390	774	422	183

Table 7: CourseMirror specificity dataset score distribution.

Table 6 shows a summary of the specificity data. The data is annotated according to the flowchart in Appendix C, and table 7 shows the score distribution across the different courses.

2.2.2 Usage

We mainly use the CourseMirror specificity data to train a specificity/quality prediction model. In our experiments on curriculum learning, we cannot use the human annotation specificity values as the courses in the two datasets (CourseMirror summarization and CourseMirror specificity) are different. Instead, we use the values predicted using the trained specificity prediction model. Appendix C describes the model used to predict reflection specificity values for CourseMirror summarization data. The predicted values for reflections are then used to calculate documents' specificity as a difficulty metric for curriculum construction.

2.3 Amazon/Yelp (AY)[3] Opinion Abstractive Summarization

2.3.1 Description

Another dataset that is closely similar to the CourseMirror dataset is Amazon/Yelp (AY) opinion dataset used in [6]. Both datasets are small in size, and in both datasets, documents consist of multiple human reviews/opinions where order doesn't matter. The Amazon/Yelp dataset contains customer reviews from Amazon [29] and Yelp[4]. The amazon data contains samples from 4 different categories (Electronics; Clothing, Shoes, and Jewelry; Home and Kitchen; Health and Personal Care). The data contains 160 (products/businesses), 60 products selected from Amazon, and 100 businesses from Yelp. Each of the products/businesses selected contains a set of 8 reviews selected from the product/business full set of reviews. For each set of 8 reviews, three human summaries are provided, table 8 shows a sample from

[3]We refer to Amazon/Yelp as AY for short

[4]https://www.kaggle.com/yelp-dataset/yelp-dataset

the Amazon data.

Amazon/Yelp	
Reviews	
Review 1	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.
Review 2	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
Review 3	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.
Review 4	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.
Review 5	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...
Review 6	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.
Review 7	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!

Review 8	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 Summaries	
Summary 1	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.
Summary 2	This woman's necklace makes a great gift for any woman or child. The thin sterling silver chain looks elegant and the heart shaped pendant is beautiful and unique. The necklace fits comfortably around the neck and looks great when worn. It's a good quality item for the price which is highly recommended.
Summary 3	This necklace is attractive, unique, and looks nice considering the inexpensive price. It makes a nice Valentine's Day gift or a gift for a child. The chain is thin but durable and should last a long time. The pendant is interesting and something different as a gift for a loved one.

Table 8: Sample from the AY training data.

Data	Products/businesses	Summaries	Avg. Num Words	Avg. Num Sentences
Amazon	60	3 * 60 = 180	56.92	3.68
Yelp	100	3 * 100 = 300	58.06	4.3

Table 9: Details of human summaries.

Data	Training	Validation	Testing
Amazon	28	12	20
Yelp	30	30	40

Table 10: Distribution of AY data.

Data	Annotators	Inter-Annotator ROUGE			
		R1	R2	RL	Avg
Amazon	A1-A2	29.65	5.32	26.94	20.64
	A1-A3	27.71	4.99	24.96	19.22
	A2-A3	28.06	5.48	26.24	19.93
	Max	29.65	5.32	26.94	20.64
	Min	27.71	4.99	24.96	19.22
	Avg	28.47	5.26	25.81	19.85
Yelp	A1-A2	27.93	4.12	24.46	18.84
	A1-A3	28.09	4.46	24.25	18.93
	A2-A3	28.1	4.22	24.41	18.91
	Max	28.09	4.46	24.25	18.93
	Min	27.93	4.12	24.46	18.84
	Avg	28.04	4.26	24.25	18.85

Table 11: AY summary analysis.

Moreover, table 9 shows further details of the human-produced summaries. The data is split into training, validation, and testing according to table 10. Finally, table 11 shows the Inter-Annotator ROUGE (IAR) scores for Amazon and Yelp data. We calculate ROUGE between every two annotators. In table 11 we show the ROUGE scores between every combinations of two annotators alongside the maximum, minimum and average of these combinations.

2.3.2 Usage

We use Amazon/Yelp data as another small-sized dataset that has common aspects with CM data. We use the Amazon/Yelp data in chapters 5, 6, and 7's experiments to further verify how far our findings can transfer between different data domains. The data is used to train and fine tune the different models across this work. The data is split into training/validation/testing splits according to table 10. The data is only annotated for abstractive summarization with no extractive summarization annotation. That is why we had to avoid experiments that involved extractive summarization auxiliary task when we used Amazon/Yelp. On the other hand, we utilized other auxiliary tasks with no issue.

2.4 CNN/DailyMail Summarization

2.4.1 Description

CNN/DailyMail [30] is a widely used summarization dataset consisting of around 300k news-oriented documents. The data is annotated for the abstractive summarization task, and we followed the work done by Chen and Bansal [12] to perform extractive summarization annotation. We show a sample from the CNN dataset alongside the generated extractive summary in Table 12. Moreover, the data is split into training, validation, and testing according to table 13. Finally, table 14 shows further details of input and abstractive summaries sizes.

CNN
Input text
Mixed martial arts fighter Anderson Silva says he will fight for a spot in the Brazilian taekwondo team at the 2016 Olympics in Rio de Janeiro. The announcement was made on Wednesday after a meeting with Brazilian taekwondo officials. Considered one of the best pound-for-pound fighters in the history of mixed martial arts, Silva said he is 'trying to give back to the sport' in which he began his career. Anderson Silva met with Brazilian taekwondo officials and will compete for a spot in the 2016 Olympics team. The former UFC champion said he is 'trying to give back to the sport' in which he began his career. Taekwondo confederation president Carlos Fernandes said it will be an 'honor for our sport to welcome an athlete of this importance'. However, he also made it clear that Silva will have to fight his way into the Olympics and wo n't be helped because of his UFC stardom. Silva is a taekwondo ambassador and a black belt in the sport. The former UFC champion tested positive for two steroids in an out-of-competition test Jan. 9, and also failed a test after his UFC victory over Nick Diaz on Jan. 31. The 40-year-old Brazilian posted a photo of himself via his Twitter page practicing taekwondo last week.
Reference Abstractive Summary
Anderson Silva met with Brazilian taekwondo officials on Wednesday. Silva is currently suspended by UFC after failing drug tests. However, the former UFC champion will fight for Olympics taekwondo spot.
Reference Extractive Summary
Anderson Silva met with Brazilian taekwondo officials and will compete for a spot in the 2016 Olympics team. Considered one of the best pound-for-pound fighters in the history of mixed martial arts, Silva said he is 'trying to give back to the sport' in which he began his career. However, he also made it clear that Silva will have to fight his way into the Olympics and won't be helped because of his UFC stardom.

Table 12: Sample data from CNN dataset.

2.4.2 Usage

The CNN/DailyMail is considered a large-sized data set (i.e., 300K samples) widely used in the summarization community. The CNN data can serve as a perfect candidate for our domain transfer and pretraining experiments for a couple of reasons; 1-The data size can help pretraining a strong neural summarization model, 2- The significant difference in style and domain between the CNN and CourseMirror data can help to show the utility of using domain transfer even with a completely different domain. We use CNN in most of our experiments across all the chapters (either for pretraining or training the rewriting model

Data	Training	Validation	Testing	Total
CNN	85K	3.2K	3.8K	92K
DailyMail	202K	7.8K	9.2K	219K
CNN + DM	287	11	13	311

Table 13: Distribution of CNN/DM data.

Data	Input		Abstractive Reference Summary	
	Num Sentences	Num Tokens	Num Sentences	Num Tokens
CNN	35.2	654.1	3.8	41.9
DailyMail	41.7	690.7	3.9	50.9
CNN + DM	39.8	679.9	3.9	48.3

Table 14: Details of input/output sizes for CNN/DM data.

(Chapter4)).

2.5 Microsoft Paraphrasing

2.5.1 Description

The MSRP [15] paraphrase dataset has been used in much paraphrasing and other related research such as data augmentation [16], multitask learning, and so forth. The data contains 4076 samples. Each sample consists of a pair of sentences with annotation (1 or 0), where one means the two sentences are paraphrases of each other, and zero otherwise. Table 15 shows two examples of the MSRP dataset, where each sample consists of two sentences and the corresponding human label (0 means no paraphrasing and 1 otherwise).

Sample 1
Sentence1: Amrozi accused his brother, whom he called "the witness", of deliberately distorting his evidence. **Sentence2:** Referring to him as only "the witness", Amrozi accused his brother of deliberately distorting his evidence. **Label:** 1
Sample 2
Sentence1: Gyorgy Heizler, head of the local disaster unit, said the coach was carrying 38 passengers. **Sentence2:** The head of the local disaster unit, Gyorgy Heizler, said the coach driver had failed to heed red stop lights. **Label:** 0

Table 15: Sample data from MSRP dataset.

2.5.2 Usage

In addition to the previous two summarization datasets (CourseMirror, and CNN), we use the MSRP dataset in our multitask learning experiments. We use the data as an additional data to train the multitask learning summarization model introduced in chapter 6 to

explore how effectively the paraphrasing auxiliary task can support abstractive summarization training. We combine the MSRP data with the original summarization data to train the paraphrasing auxiliary task.

3.0 Related Work

The challenge of information overload has triggered the research of automatic summarization in the community of natural language processing (NLP). Automatic summarization is the task of taking an input of text documents, speech, or video and producing a concise summary of the most crucial information of the original documents [69]. Summarization, in general, can be categorized into two major categories:

- Extractive
- Abstractive

Each of these two categories has its pros and suffer from its cons. Both similarly got the attention of much research from the NLP community.

3.1 Extractive Summarization

Extractive summarization focuses on enumerating and extracting the most salient pieces of text from the input. Unlike abstractive summarization, which produces a coherent paragraph, extractive summarization aims to extract the most important pieces that satisfy a couple of constraints:

- Salience: Extracted pieces maximize coverage of major points from the input text.
- Novelty: Minimize the redundancy and similarity between the extracted pieces.

Earlier work on extractive summarization involved unsupervised techniques such as handcrafted features to select important sentences [18], constructing relational graphs that can be later traversed [70], or using ILP [4, 62]. Later on, deep networks started to be a basic block of state of the art (SOTA) models across numerous NLP tasks. One of the very early works that first introduced RNN encoders for extractive summarization models is SummaRuNNer [66]. Another line of work proposed integrating pointer networks to select salient sentences and keywords [34]. Recently, huge performance strides were achieved in different NLP tasks

using pretrained language models as encoders [14, 40, 77, 47]. In turn, a lot of summarization research started incorporating pretrained encoders such as BERT [104, 45]. This work deals with the extractive summarization task as a simple classification task. Moreover, since dataset such as CM consists of independent reflections and the human reference summaries are similarly a set of independent reflections, we represent the extractive summarization tasks as a non-auto-regressive classification task, where unlike [66, 104, 45] we classify each reflection of the input as being part of the summary or not regardless of the classification of other prior reflections. Finally, similar to work done by Zhong et al. [104] and Liu and Lapata [45], we use deep encoders to provide a representation for both input document and reflection to be classified, however in this work, we don't only experiment with BERT, but we use multiple encoders such as (BERT, BART, and T5).

3.2 Abstractive Summarization

Abstractive summarization aims to generate coherent summaries with high readability and has seen increasing interest, and improved performance due to the emergence of seq2seq models [88] and attention mechanisms [2]. For example, Rush et al. [79] used a vanilla encoder-decoder model with attention, See et al. [86], Paulus et al. [74], and Gehrmann et al. [23] used pointer networks to solve the out of vocabulary issue, while See et al. [86] used coverage mechanism to solve the problem of word repetition. In addition, Paulus et al. [74] and Chen and Bansal [12] used reinforcement learning in an end-to-end setting. After introducing transformer models, great attention began to shift towards improving the traditional RNN-seq2seq models through transformers. Which, in turn, gave birth to a lot of new abstractive summarization models that incorporate transformers into its building blocks, either using encoder-decoder structure [44, 40, 77] or simply using decoder only structure [44]. Böhm et al. [5] proposed eliminating the dependency on reference summaries and proposed a model to train abstractive summarization model using reinforcement learning, where the reward function is a holistic function of different human judgment criteria (readability, coherency, etc.). Liu and Lapata [45] introduced an extractive and abstractive unified

framework using BERT encoder and transformer decoder. As per integrating GANs in abstractive summarization, Liu et al. [43] proposed formulating the abstractive summarization in a GAN setting, where they trained a generator and a discriminator using reinforcement learning. Due to the complexity of training abstractive summarization models using both RL and GAN frameworks, this work, similar to multiple prior work [79, 86, 44, 40, 77] deals with the abstractive summarization task using the most common approach utilizing encoder-decoder models. However, since the primary focus of our work is not improving the SOTA of abstractive summarization but improving model training for low resource settings, we perform experiments across this work using multiple SOTA abstractive summarization models [86, 41, 77, 45] to verify our findings.

3.2.1 Abstractive Summarization for Low Resource Data

A lot of recent work have been targeting different aspects to improve abstractive summarization models. However, most of these works focus on developing and testing their models in environment where a lot of annotated data exists such as (news, publications, etc.). On the other hand few number of works tackled the problem of improving abstractive summarization models in domains with scarce data. For example; Bajaj et al. [3] used a pretrained abstractive summarizer (BART) [41] to avoid training new model using their scarce data. However, to overcome their data length issue, they proposed doing data compression by first extracting salient sentences before the summarization step. Bražinskas et al. [6] proposed training abstractive summarization model by first training the model in unsupervised setting as a language model that produces new reviews given old ones. Later on the model is then tuned for summarization using a handful of samples annotated with human summaries. Another recent work which was also concerned with low resources domains is done by Sarkhel et al. [84]. While, their experiments were focused on low resources domain, their main objective was interpretability throughout multiheaded attention. Our work differs from these prior works in many aspects, our work spans two different approaches to improve performance (model-based, and data-based). The work doesn't focus on a certain abstractive summarization model, in contrast it involves different models either to verify

generalizability or to perform experiments with SOTA models. We focus more on techniques that can improve the training process as a whole, either by data augmentation, synthesis and curriculum learning or by searching for auxiliary tasks that can be integrated into any model to improve the performance of abstractive summarization (more in following sections).

3.2.2 Domain Transfer in Abstractive Summarization

To our knowledge, training neural abstractive summarization models in low resource domains using domain transfer has not been thoroughly explored on domains different than news. For example, Nallapati et al. [67] reported the results of training on CNN/DM data while evaluating on DUC data without any tuning. Note that these two datasets are both in the news domain, and both consist of well written, structured documents. The domain transfer experiments of Gehrmann et al. [23] similarly used two different news summarization datasets (CNN/DM and NYT). Our work on domain transfer on the other hand differs in several ways from these two prior domain transfer efforts. First, our work involve two entirely different domains: news and student reflections. Unlike news, student reflection documents lack global structure, are repetitive, and contain many sentence fragments and grammatical mistakes. Second, the prior approaches either trained a part of the model using NYT data while retaining the other part of the model trained only on CNN/DM data [23], or didn't perform any tuning at all [67]. In contrast, we do the training in two consecutive phases, pretraining and fine tuning, where the whole model is trained using the in-domain and out-of-domain data. Finally, Gehrmann et al. [23] reported that while training with domain transfer outperformed training only on out-of-domain data, it was not able to beat training only on in-domain data. This is likely because their in and out-of-domain data sizes are comparable, unlike in our case of scarce in-domain data that would easily lead to model overfitting.

Another work that was done after ours, is done by Yu et al. [102]. In their work they study the effect of using large unlabeled data from different domains to pretrain a BART model, and analyze the effect of pretraining and fine tuning with several different domains. Our work differs in several aspects. First, our work doesn't focus on improving a certain

summarization model (BART) in the case of Yu et al. [102]; in contrast, we focus on using data efficiently, which can be applied to various models. Second, in our work we gradually move from using pretraining with out of domain data to using techniques that can improve the training process without additional data, e.g., synthesis, augmentation, and curriculum learning.

3.2.3 Templates in Abstractive Summarization

In a different approach to abstractive summarization, Cao et al. [7] developed a soft template based neural method consisting of an end-to-end deep model for template retrieval, reranking and summary rewriting. While we also develop a template based model, our work on template model differs in both model structure and purpose. Our model is structured to take advantage of properties of domains which don't involve storytelling, but instead tend to be systematic with high dependence on keywords (e.g., technical terms, etc.). In addition, while our proposed model integrates a rewriting module which was inspired by Cao et al. [7], our model is tailored for our goal of synthesizing data in a low resources domain, for use by different abstractive summarizers with high model complexity. In contrast, Cao et al. [7] proposed their model with the goal of directly producing an abstractive summary. Thus, unlike Cao et al.'s end-to-end model which requires a lot of training data, we create sub-modules like rewriting and scoring, which can be trained using not only summarization data but any generic textual data.

3.3 Phrase Summarization

Phrase summarization can be considered a special case of extractive summarization, in which the summarization process is done on a different level of granularity. Phrase summarization aims to extract and retrieve salient phrases instead of complete sentences. Phrase summarization can help in numerous use cases, such as extracting a product's major aspects/characteristics (e.g., frame durability, brewing speed, etc.) by summarizing user re-

views, similar to the work done by Yu et al. [101]. Another line of work on phrase summarization is the work done by Luo et al. [52, 55, 54] on summarizing students' feedback by extracting the most relevant phrases. Additionally, Yao et al. [100] investigated cross-lingual summarization by extracting phrases instead of sentences. Similar to phrase summarization the work done on keyphrase extraction [97, 49, 63, 28] also aims to extract the most prevailing/salient phrases from the input text. In this work, the main focus is on abstractive summarization. However, in chapter 4 we compare against the work done by Luo et al. to verify the potential of abstractive summarization models compared to extractive ones on the CM dataset.

3.4 Data Augmentation

Data synthesis for text summarization is underexplored, with most prior work focusing on machine translation, machine comprehension, and text normalization. Data synthesis approaches have varied from back-translation, word replacement to sentence re-writing. Zhang et al. [103] and Wang and Yang [94] proposed doing data augmentation through word replacement, using WordNet [65] and vector space similarity, respectively. We will use a WordNet replacement method as a baseline synthesis method in the experiments described below. In contrast, Fadaee et al. [20], Wang et al. [95], and Sennrich et al. [87] synthesized/augmented data through back-translation and word replacement using language models. In an approach more similar to our proposed template model in terms of model design (i.e., extracting then rewriting using seq2seq model), Tan et al. [89] trained a multi-task learning model for snippet extraction and answer rewriting, where they used a seq2seq network to rewrite snippets extracted from the paragraph in the form of a final answer. While there are fundamental differences between their model and ours (e.g. multitask vs. single task, end-to-end training, etc.), the motivating purposes are also different. Their model was developed to rewrite the answer in a new format to account for words that might not appear in the snippet but are part of the original answer, while our model was developed to synthesize summaries to augment training data in low-resource domains. The work done by Parida and Motlicek [72]

is another recent work that trained a model to back generate a source document given a summary. We think it might be infeasible to back generate student reflections from a human summary, especially an abstractive one. Chen et al. [9] introduced a new approach for data augmentation (MixText). MixText augmentation allows performing augmentation not only on the input level but on different levels of the model. We build a part of our work on the MixText approach; however, the major difference is the downstream task. Chen et al. [9] proposed their model for classification based tasks. We introduce a variation of the approach suitable for generative tasks (MixGen) and use it in abstractive summarization.

3.5 Curriculum Learning

Curriculum learning has gathered much attention lately as a technique to improve the training procedure. The curriculum helps the model learn gradually by feeding the training data in difficulty ascending order based on predefined difficulty criteria. It has been successfully applied in computer vision [27, 35, 26], and in NLP [80, 81, 90, 93] for machine comprehension, question generation, reading comprehension and machine translation respectively. Xu et al. [98] introduced a new approach for curriculum learning to improve model performance on different natural language understanding tasks, including machine comprehension. We build our work on their idea; however, the core differences are both the downstream tasks and the difficulty criteria. In their work, Xu et al. [98] focused only on classification tasks; however, in ours, we focus on abstractive summarization tasks. Furthermore, we introduce a couple of functions for difficulty evaluation (ROUGE and specificity). Finally, we study the effect of curriculum learning in different configurations, either in isolation or in combination with data synthesis and multitask learning.

3.6 Multitask Learning

Multitask learning has been the focus of a lot of recent research, and it showed a lot of improvements in different NLP tasks. Abstractive summarization has been enhanced by integrating it with text entailment generation [73], with extractive summarization [11, 31], and with sentiment classification [8, 56] in multitask learning frameworks. While these works each integrate only one task, Lu et al. [50] and Guo et al. [25] combined multiple tasks. However, Lu et al. [50] focused on integrating only predictive tasks (syntax labeling and text categorization), while Guo et al. [25] instead used generative tasks (entailment generation and question generation). Our work focuses on both generative and predictive tasks, explores task utility in isolation and in all combinations, and demonstrates generalization of findings across multiple models and data. Furthermore, while two of our auxiliary tasks *(language modeling* [61] and *extractive summarization* [73]) have been examined before in the context of multitask summarization, we introduce two additional auxiliary tasks *(paraphrase detection, concept detection)*. Finally, while previous work relied on large training corpora (e.g. CNN/DailyMail [30]), we target low resource domains and try to overcome data scarceness by using the same data to train multiple task modules.

3.7 Evaluation

Summarization evaluation has been a pressing topic within the NLP community. The issue lays in the subjective nature of summaries. The evaluation process requires using human summaries to judge the machine generated one. However, when humans generate summaries it can be very subjective, especially abstractive summaries. The subjectivity and freedom to use own words render the evaluation process to be difficult and sometimes misleading. Thus, there has been a lot of efforts to formulate an informative evaluation metric. One of the early and far more the standard metric used in the majority of summarization literature is ROUGE. ROUGE [42] is adopted by the summarization community as the standard evaluation metric to evaluate the quality of summarization due to being fairly simple to calculate and it's

reasonable correlation to human evaluation. ROUGE is calculated by counting the n-gram overlap between a human summary (reference) and a machine produced one (prediction). ROUGE score is evaluated on recall, precision and F-measure as follows:

$$ROUGE_N\_Recall = \frac{Count_{match}(N)}{\sum_{S \in RS} \sum_{ngram \in S} Count(ngram)} \tag{1}$$

$$ROUGE_N\_Precision = \frac{Count_{match}(N)}{\sum_{S \in MS} \sum_{ngram \in S} Count(ngram)} \tag{2}$$

$$ROUGE_N\_F = \frac{(1+\beta^2)\ X\ ROUGE_N\_Recall\ X\ ROUGE_N\_Precision}{ROUGE_N\_Recall\ +\ \beta^2 ROUGE_N\_Precision} \tag{3}$$

where N is the length of the n-gram, and $ngram$ is an n-gram of length N. S is a sentence, RS and MS are reference summary and machine summary respectively. $Count(ngram)$ and $Count_{match}(N)$ are counting the number of ngrams in a sentence S, and counting the number of matching ngrams of size N between RS and MS respectively. However, ROUGE unfortunately suffers from inability to match sematically simillar terms, which is a major property of abstractive summarization (summaries can contain new words). With that said a lot of effort has been invested in seeking unorthodox evaluation metrics. For example Chen et al. [10] proposed to do evaluation by extracting a set of question from the topic and measure how many of these questions can be answered using the original text and a summary. In a similar context Durmus et al. [17] proposed integrating question answering evaluation to judge model faithfulness. On the other hand, Clark et al. [13] measured semantic similarity by means of earth moving algorithm and semantic embeddings. In contrast to these automatic methods, a lot of recent work performed human evaluation as an additive evaluation method to complement the automatic evaluation and judge machine summaries across different dimensions (readability, coherence, relevancy, etc.). In the course of this work we use ROUGE, and human evaluation as evaluation metrics for our experiments.

4.0 Template-Based Data Synthesis and Domain Transfer Through Pretraining (Data-Based Direction) (Published in FLAIRS-33 [57])

4.1 Introduction

Recently, with the emergence of neural seq2seq models, abstractive summarization methods have seen great performance strides [86, 23, 74]. However, complex neural summarization models with thousands of parameters usually require a large amount of training data. In fact, much of the neural summarization work has been trained and tested in news domains where numerous large datasets exist. For example, the CNN/DailyMail (CNN/DM) [30, 67], New York Times (NYT) [82] and Gigaword [68] datasets are in the magnitude of 300k, 700k, and 3000k documents, respectively. In contrast, in other domains such as student reflections, summarization datasets are only in the magnitude of tens or hundreds of documents (e.g., CourseMirror [52]). We hypothesize that training complex neural abstractive summarization models in such domains will not yield good performing models, and we will indeed later show that this is the case for student reflections.

To improve performance in low resource domains, we explore three directions. First, we explore domain transfer for abstractive summarization. While domain transfer is not new, compared to prior summarization studies [32, 36], our training (news) and tuning (student reflection) domains are quite dissimilar, and the in-domain data is small. Second, we propose a template-based synthesis method to create synthesized summaries, then explore the effect of enriching training data for abstractive summarization using the proposed model compared to a synthesis baseline. Lastly, we combine both directions. Evaluations of two neural abstractive summarization methods across four student reflection corpora show the utility of all three methods.

4.2 Explored Approaches

4.2.1 Domain Transfer

To overcome the size issue of the student reflection dataset, we first explore the effect of incorporating **domain transfer** into two state of the art abstractive summarization models: pointer networks with coverage mechanism (PG-net) [86] and fast abstractive summarization with reinforcement learning (Fast-RL) [12]. One major reason for choosing these models is that implementations of both are available[1]. This makes it easier to fully understand the models and to make changes needed for domain transfer compared to re-implementing both systems from scratch. Another reason is that while both models are state of the art, they use two different paradigms for training. PG-net trains to optimize a maximum-likelihood objective, while Fast-RL trains with reinforcement learning to maximize ROUGE-1. We later modified the off the shelf code of Fast-RL to accommodate for fine tuning. To experiment with domain transfer, both models were pretrained using the CNN/DM dataset, then fine tuned using the student reflection dataset (see the Experiments section).

4.2.2 Data Synthesis

A second approach we explore to overcome the lack of reflection data is data synthesis. We first propose a template model for synthesizing new data, then investigate the performance impact of using this data when training the summarization models. The proposed model makes use of the nature of datasets such as ours, where the reference summaries tend to be close in structure: humans try to find the major points that students are concerned about, then present the points in a way that marks their relative importance (recall the CS example in Table 1). Examples from two other courses are below:

- Most students found type 2 errors interesting. A minority found type 1 errors, confidence intervals, beta number, and p value interesting. (Stat2015)
- Most students found learning about specific syntax and commands in matlab interesting. Some students were simply interested in programming in general. (ENGR)

[1] https://github.com/ChenRocks/fast_abs_rl; https://github.com/abisee/pointer-generator

- Many students found the concepts of runtime analysis, threads and efficiency between binary and sequential search as interesting. Some other students liked synchronization and vectors. (CS)

We then explore with **combining domain transfer with data synthesis**.

4.3 Proposed Template-Based Synthesis Model

Our motivation for using templates for data synthesis is that seq2seq synthesis models (as discussed in related work) tend to generate irrelevant and repeated words [38], while templates can produce more coherent and concise output. Also, extracting templates can be done either manually or automatically typically by training a few parameters or even doing no training, then external information in the form of keywords or snippets can be populated into the templates with the help of more sophisticated models. Accordingly, using templates can be very tempting for domains with limited resources such as ours.

4.3.1 Model Structure

The model consists of 4 modules:

1. Keywords and template extraction: To convert human summaries into templates, we remove keywords in the summary to leave only non-keywords. We use Rapid Automatic Keyword Extraction (RAKE) [78] to identify keywords. We then split the summary into template (summary without the keywords) and extracted keywords
2. Template clustering: Upon converting human summaries into templates, we cluster them into N clusters with the goal of using any template from the same cluster interchangeably. A template is first converted into embeddings using a pretrained BERT model [14], where template embedding is constructed by average pooling word embeddings. Templates are then clustered using k-medoid.
3. Summary rewriting: An encoder-attention-decoder with pointer network is trained to perform the rewriting task. The model is trained to inject keywords into a template and

perform rewriting into a coherent paragraph. The produced rewrites are considered as candidate summaries.

4. Summary selection: After producing candidate summaries, we need to pick the best ones. We argue that the best candidates are those that are coherent and also convey the same meaning as the original human summary. We thus use a hybrid metric to score candidates, where the metric is a weighted sum of two scores and is calculated using Equations 4, 5, and 6. Eq.4 measures coherency using a language model (LM), Eq.5 measures how close a candidate is to a human summary using ROUGE scores, while Eq.6 picks the highest scored N candidates as the final synthetic set.

$$LM_S = (\sum_{w \in CS} log(P(w)))/(len(CS)) \tag{4}$$

$$R_S = Avg(\sum_{i \in [1,2,l]} R_i(CS, HS)) \tag{5}$$

$$Score = (\alpha LM_S + \beta R_S)/(\alpha + \beta) \tag{6}$$

CS and HS are a candidate and human summary. $P(w)$ is the probability of word w using a language model. α, β are weighting parameters. In this work we use $\alpha = \beta = 1$ for all experiments. $R_i(CS, HS)$ is ROUGE-i score between CS and HS for i=1, 2, and l.

4.3.2 Model Training

Before using the synthesis model, some of the constructing modules (rewriting module, scoring LM) need training. To train the rewriting model, we use another dataset consisting of a set of samples, where each sample can be a text snippet (sentence, paragraph, etc.). For each sample, keywords are extracted using RAKE, then removed. The keywords plus the sample with no keywords are then passed to the rewriting model. The training objective of this model is to reconstruct the original sample, which can be seen as trying to inject extracted keywords back into a template.

4.3.3 Model Usage

To use the synthesis model to generate new samples, the set of human summaries are fed to the model, passing through the sub-modules in the following order:

1. Human summaries first pass through the template extraction module, converting each summary s_i into template t_i and the corresponding keywords kw_i.
2. Templates are then passed to the clustering module, producing a set of clusters. Each cluster C contains a number of similar templates.
3. For each template t_i and corresponding keywords kw_i from step 1, find the cluster C_i that contains the template t_i, then pass the set of templates within that clusters $\{t_j\}\forall j$, if $t_j \in C_i$ alongside the keywords kw_i to the summary rewriting module. This will produce a set of candidate summaries.
4. The summary selection module scores and selects the highest N candidates as the synthetic summaries.

4.4 Experiments

Our experimental designs tackle the following research questions:

- **Question 1 (Q1)** : Would training complex abstractive models with limited in-domain or large quantities of out-of-domain be able to outperform extractive baselines ?
- **Question 2 (Q2)** : Would domain transfer help abstractive models even if in-domain and out-of-domain data are very different and the amount of in-domain data is very small ?
- **Question 3 (Q3)** : Would enriching abstractive training data with synthetic data helps overcome in-domain data scarcity ?
- **Question 4 (Q4)** : How would the proposed template-based synthesis model perform compared to a simple word replacement model ?
- **Question 5 (Q5)** : Would combining domain transfer with data synthesis outperform using each approach in isolation ?

- **Question 6 (Q6)** : Can the synthesis model be extended to perform reflection summarization directly ?

4.4.1 Extractive Baselines (for answering Q1)

While both See et al. [86] and Chen and Bansal [12] used Lead-3 as an extractive baseline, in our data sentence order doesn't matter as reflections are independent. We thus use a similar in concept baseline: randomly select N reflections. Since the baseline is random we report the average result of 100 runs. Following [52], we compare our results to MEAD [76] and to Luo and Litman's extractive phrase-based model. [2] Since they extracted 5 phrases as their extractive summary, we use N=5 for our three extractive baselines (Random Select, MEAD, Luo and Litman). Additionally we compare to running only the extractive part of Fast-RL.

4.4.2 Domain Transfer (for answering Q2, Q5)

To observe the impact of using out-of-domain (news) data for pretraining to compensate for low resource in-domain (reflection) data, we train 3 variants of PG-net and of Fast-RL: model training on CNN/DM; model training on reflections; and model training on CNN/DM then tuning using reflections. Table 16 shows example summaries generated by the three variants of PG-net for a CS document.

For all experiments where reflections are used for training/tuning, we train using a leave one course out approach (i.e, in each fold, three courses are used for training and the remaining course for testing). If the experiment involves tuning a combined dictionary of CNN/DM and reflections is used for training to avoid domain mismatch. To tune model parameters, the best number of steps for training, the learning rate, etc., a randomly selected 50% of the training data is used for validation. We choose the parameters that maximize ROUGE scores over this validation set.

To implement PG-net we use OpenNMT [37][3]. The model uses 128-dimensional word-

[2]Designed for reflection summaries.

[3]http://opennmt.net

Model	**Summary**
CNN/DM	Internal vs. external version of iteration iterarors i was a bit preoccupied today but seeing merge sort. How typically iterating through a linked list can be very inefficient the implementation of iterators iterators and their effectiveness how iterators can be used.
Student Reflections	Most students found the data of data along with its mean and effectiveness interesting, as well as topics related to sse, their, and different. Students also found different a good topic.
Tuned	Most of students were interested in iterators, the concept of iterators, and quick sort and merge sort. They also found analyzing linked lists in regards to runtime to be interesting.
Human Reference	
Most of the students found iterators and linked lists as interesting. Some of them liked merge sort and quick sort. A few of them liked internal vs external iteration and analyzing runtimes of linked lists.	

Table 16: Summaries generated by the three variants of [86] for the same CS reflection document.

embeddings and a 512-dimensional 1 layer LSTM. On the encoder side, we use a bidirectional LSTM. The model is trained with adagrad optimizer and an initial learning rate (LR) of 0.15. The out-of-domain model is trained for 100k steps using the CNN/DM dataset. Following base model training, we tune the model by training it using student reflections. The tuning is done by lowering the LR from 0.15 to 0.1 and training the model for additional 500 steps. The in-domain model is trained only using reflections. We use the same model architecture as above and train the model for 20k steps using adagrad optimizer and initial LR of 0.15. For Fast-RL, to train the base model we follow the authors' instructions[4]. For tuning, the initial LR is decreased from $1x10^{-3}$ to $1x10^{-4}$.

4.4.3 Synthesis Baseline (for answering Q3, Q4)

Following [103], we developed a data synthesis baseline using word replacement via WordNet. The baseline iterates over all words in a summary. If word X has N synonyms in WordNet, the model creates N new versions of the summary and corresponding reflections by replacing the word X with each of the N synonyms. Since the number of synonyms is variable for each word, the number of resultant synthetic samples is different for each original sample. On the other hand, the proposed template model creates a fixed number of samples for each original sample so avoids flooding the synthetic data with one sample and guarantees data balance. Simple word replacement models like the ones proposed by [103] and [94] also suffer from other shortcomings, e.g., not considering word senses, compound words, or technical terms. Our proposed synthesis model focuses more on keeping major keywords, and additionally makes use of the power of seq2seq models to perform rewriting and produce a coherent output.

4.4.4 Template Synthesis Model (for answering Q4, Q5)

To synthesize summaries, we use the same leave one course out approach. For each course, we use the data from the other three courses to train the rewriting module and tune the scoring language model. We can also use the summaries from CNN/DM data as

[4]https://github.com/ChenRocks/fast_abs_rl

additional samples to further train the rewriting module. We then start synthesizing data using that training data as input. First templates are constructed. The templates are then clustered into 8 clusters. We decided to use 8 to avoid clustering templates from POI with MP, as the templates from both prompts would contain very different supporting words. We also wanted to avoid a high level of dissimilarity within each cluster, and allow some diversity. Following the clustering, the rewriting model produces candidate summaries for each human summary. The rewriting model is an encoder-attention-decoder with pointer network. The rewriting model, similar to PG-net model, uses 128-dimensional word-embeddings and a 512-dimensional 1 layer LSTM. On the encoder side, we use a bidirectional LSTM. The model is trained with adagrad optimizer and an initial learning rate of 0.15. After producing

Original Human Summary
almost all the students enjoyed the algorithm for determining the height of binary trees. a few others also mentioned binary trees and trees in general.
Extracted Keywords
algorithm, binary trees, binary trees, trees
Synthesized Sample
most of the students found binary trees and finding the height through recursion as interesting. some mentioned binary trees some of them didn't find anything interesting in the lecture.

Table 17: An example of synthesized CS summary.

the candidate summaries, a language model is used to score them. The language model is a single layer LSTM language model trained on 36K sentences from Wikipedia and fine tuned using student reflections. In this work we decided to pick only the highest 3 scored candidate summaries as synthetic data, to avoid adding ill-formed summaries to the training data. Since we are adding N synthetic summaries for each set of reflections, that means we are essentially duplicating the size of our original reflection training data by N, which is 3 in our case. Table 17 shows a human summary, the keywords extracted, then the output of

injecting keywords in a different template using rewriting.

4.4.5 Template-based Summarization (for answering Q6)

While the proposed template-based model was intended for data synthesis, with minor modification it can be adapted for summarization itself. Because the modifications introduce few parameters, the model is suitable for small datasets. Recall that for data synthesis, the input to the template method is a summary. Since for summarization the input instead is a set of reflections, we perform keyword extraction over the set of reflections. We then add an extra logistic regression classifier that uses the set of reflections as input and predicts a cluster of templates constructed from other courses. Using the keywords and the predicted cluster of templates, we use the same rewriting model to produce candidate summaries. The last step in the pipeline is scoring. In data synthesis, a reference summary is used for scoring; however, in summarization we don't have such a reference. To score the candidate summaries, the model only uses the language model and produces the candidate with the highest score.

4.5 Results

4.5.1 ROUGE Evaluation Results

Table 18 presents summarization performance results[5] for the six extractive baselines, for the original and proposed variants of the two abstractive summarization models Fast-RL and PG-net, and finally for template-summarization. Following See et al. [86] and Chen and Bansal [12], performance is evaluated using ROUGE-1 (R-1), ROUGE-2 (R-2), and ROUGE-L (R-L) [42] on F1.

The motivation for using domain transfer and data synthesis is our first questions **(Q1)**. Table 18 answers this questions. All ROUGE scores for Fast-RL and PG-net that outperform

[5]We performed pairwise t-test for statistical significance over the results of the four courses. Unfortunately, four samples is a sample size, thus, majority of the scores are not significant

Summarization Model		CS			ENGR			
		R-1	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**	
Extractive Baselines	Luo et al. [52]	27.65	6.66	22.76	30.99	8.97	25.38	1
	Mead 5	30.59	8.26	23.78	29.35	7.91	23.12	2
	Random Select 5	26.74	5.89	20.55	26.14	5.35	20.57	3
	Fast-RL CNN/DM	28.95	6.62	22.16	26.6	5.09	21.06	4
	Fast-RL Reflections	27.57	7.26	21.25	21.78	2.91	15.97	5
	Fast-RL Tuned	26.86	6.04	20.91	20.84	3.82	15.61	6
Fast-RL	CNN/DM	28.75	6.38	22.63	27.51	6.42	22.17	7
	Reflection	23.98	4.61	18.22	22.63	4.74	17.62	8
	Tuned	30.28	8.09	*24*	*31.67*	*9.5*	24.41	9
PG-net	CNN/DM	29.83	7.10	18.28	29.30	6.95	17.63	10
	Reflection	25.90	4.62	17.49	26.14	6.05	20.94	11
	Reflection+WordNet	27.15	3.13	17.8	28.11	6.11	21.29	12
	Reflection+Template	26.93	3.49	19.38	29.54	6.96	21.30	13
	Tuned	*37.31*	*10.20*	*24.16*	*38.47*	*13.88*	*27.79*	14
	Tuned+WordNet	*34.13*	7.13	21.96	*32.61*	7.51	21.72	15
	Tuned+Template	***37.88***	***11.01***	***25.30***	***38.98***	***13.97***	***28.65***	16
Template Model Summarizing		*34.8*	*9.3*	23.4	*36.5*	*11.2*	24.1	17

Summarization Model		Stat2015			Stat2016			
		R-1	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**	
Extractive Baselines	Luo et al. [52]	28.84	10.15	25.05	32.96	**12.44**	27.90	18
	Mead 5	26.06	8.84	21.28	32.31	12.30	26.27	19
	Random Select 5	23.50	5.88	19.46	23.77	7.63	20.11	20
	Fast-RL CNN/DM	27.49	7.73	22.05	24.59	8.16	20.66	21
	Fast-RL Reflections	20	5.5	16.08	18.3	6.28	15.2	22
	Fast-RL Tuned	15.73	4.89	12.7	15.39	5.37	12.84	23
Fast-RL	CNN/DM	23.74	5.75	19.63	24.44	7.06	20.51	24
	Reflection	26.61	6.41	19.32	30	5.43	20.74	25
	Tuned	*29.33*	7.21	21.56	29.48	6.73	22.39	26
PG-net	CNN/DM	27.22	7.62	17.80	30.99	10.01	20.29	27
	Reflection	*29.29*	5.66	20.31	32.10	5.92	22.28	28
	Reflection+WordNet	26.11	5.26	20.41	31.92	6.14	22.36	29
	Reflection+Template	*29.65*	5.42	20.54	32.43	5.96	21.53	30
	Tuned	*38.78*	***12.45***	*26.19*	***41.05***	12.17	*28.25*	31
	Tuned+WordNet	*34.65*	9.88	24.31	*36.58*	9.78	24.08	32
	Tuned+Template	***39.12***	*12.23*	***26.93***	*40.95*	12.26	***28.51***	33
Template Model Summarizing		*35.6*	*11.1*	24.8	*38.3*	11.9	25.6	34

Table 18: ROUGE results. *Italics* indicates outperforms baselines. **Boldface** indicates best over all. Underlining indicates best result in a group (i.e., Baselines, Fast-RL, PG-net).

all extractive baselines (in italics) involve tuning and/or use of synthesised data, except for one R-1 exception (row 28).

Our second question **(Q2)** asks if domain transfer via model tuning should benefit both the Fast-RL and PG-net abstractive models. Table 18 shows that fine tuning can indeed improve the training of complex neural models when there is a large amount of out-of-domain data but only a small amount of in-domain reflection data. For PG-net, comparing the CNN/DM out-of-domain and Student Reflection in-domain results in rows (10 and 11) and (27 and 28) with their corresponding tuned results in rows 14 and 31, we see that fine tuning improves R-1 scores from (29.83 and 25.9) to 37.31, (29.3 and 26.14) to 38.47, (27.22 and 29.29) to 38.78, and (30.99 and 32.1) to 41.05 for CS, ENGR, Stat2015, and Stat2016, respectively. Similarly R-2 and R-L are also improved for all courses (rows 10, 11, 14 and 27, 28, 31). We also see the same benefits of model tuning for the Fast-RL model: compared to using only news or only reflection data, the tuned models boost all ROUGE scores for all courses (rows 7, 8, 9 and 24, 25, 26), except for R-1 and R-2 in Stat2016 (row 26). Qualitatively, the examples presented in Table 16 clearly show that tuning yields a more coherent and relevant summary. Comparing the tuned versions of the two abstractive summarization models to each other, Fast-RL (rows 9 and 26) achieves lower ROUGE scores overall compared to PG-net (rows 14 and 31). For 8 of the 12 tuned results, Fast-RL is not even able to beat the best baseline result. One factor might be the number of possibilities for tuning Fast-RL. While PG-net is a single model that needs tuning, Fast-RL involves three models that need tuning (extractor, abstractor, RL agent), which introduces more parameters to optimize and also more choices for tuning (e.g., tune all models or a subset of them). One support for this intuition is that when PG-net and Fast-RL are trained using CNN/DM they achieve similar ROUGE scores; bigger differences appear only after tuning. Over all courses, the tuned version of PG-net consistently outperforms the best baseline result for each metric (rows 14 vs. 1, 2, 3, 4, 5, 6 and 31 vs. 18, 19, 20, 21, 22, 23) except for R-2 in Stat2016.

Our next set of questions **(Q3, Q4. Q5)** asks if enriching low resources data with synthesized data will render the in-domain data more effective in training or tuning abstractive summarization models, and if the proposed template based synthesis model is indeed superior

to simple baselines like word replacement with WordNet. For these experiments we use only PG-net to check the impact of data synthesis, since in general it performs better than Fast-RL on our dataset. We use the synthesized data in two settings: either using it for training (rows 12, 13 and 29, 30) or tuning (rows 15, 16 and 32, 33). Table 18 answers **Q4** by showing that the proposed synthesis model outperforms the WordNet baseline in training (rows 12, 13 and 29, 30) except for R-1 on CS and R-2 and R-L on Stat2016, and tuning (15, 16 and 32, 33) over all courses. It also shows that while adding synthetic data from the baseline is not always helpful, adding synthetic data from the template model helps to improve both the training and the tuning process. In the CS course, tuning with synthetic data enhances ROUGE scores by 0.57%, 0.81%, and 1.14% for R-1, R-2, and R-L, respectively, compared to tuning with only the original data. ENGR ROUGE scores are also enhanced by 0.51%, 0.09%, and 0.86% for R-1, R-2, and R-L, respectively (rows 14 and 16). As for Stat2015, R-1 and R-L increase by 0.34% and 0.74% respectively, while R-2 decreases by 0.22%. For Stat2016, R-2 and R-L increase by 0.13% and 0.26% respectively, and R-1 decreases by 0.1% (rows 31 and 33). Training with both student reflection data and synthetic data compared to training with only student reflection data yields similar improvements, answering **Q3** (rows 11, 13 and 28, 30). While the increase in ROUGE scores is small, our results show that enriching training data with synthetic data can benefit both the training and tuning of other models. In general, the best results are obtained when using data synthesis for both training and tuning (rows 16 and 33), answering **Q5**.

Finally, while the goal of our template model was to synthesize data, using it for summarization is surprisingly competitive, answering **Q6**. We believe that training the model with little data is doable due to the small number of parameters (logistic regression classifier only). While rows 17 and 34 are never the best results, they are close to the best involving tuning. This encourages us to enhance our template model and explore templates not so tailored to our data.

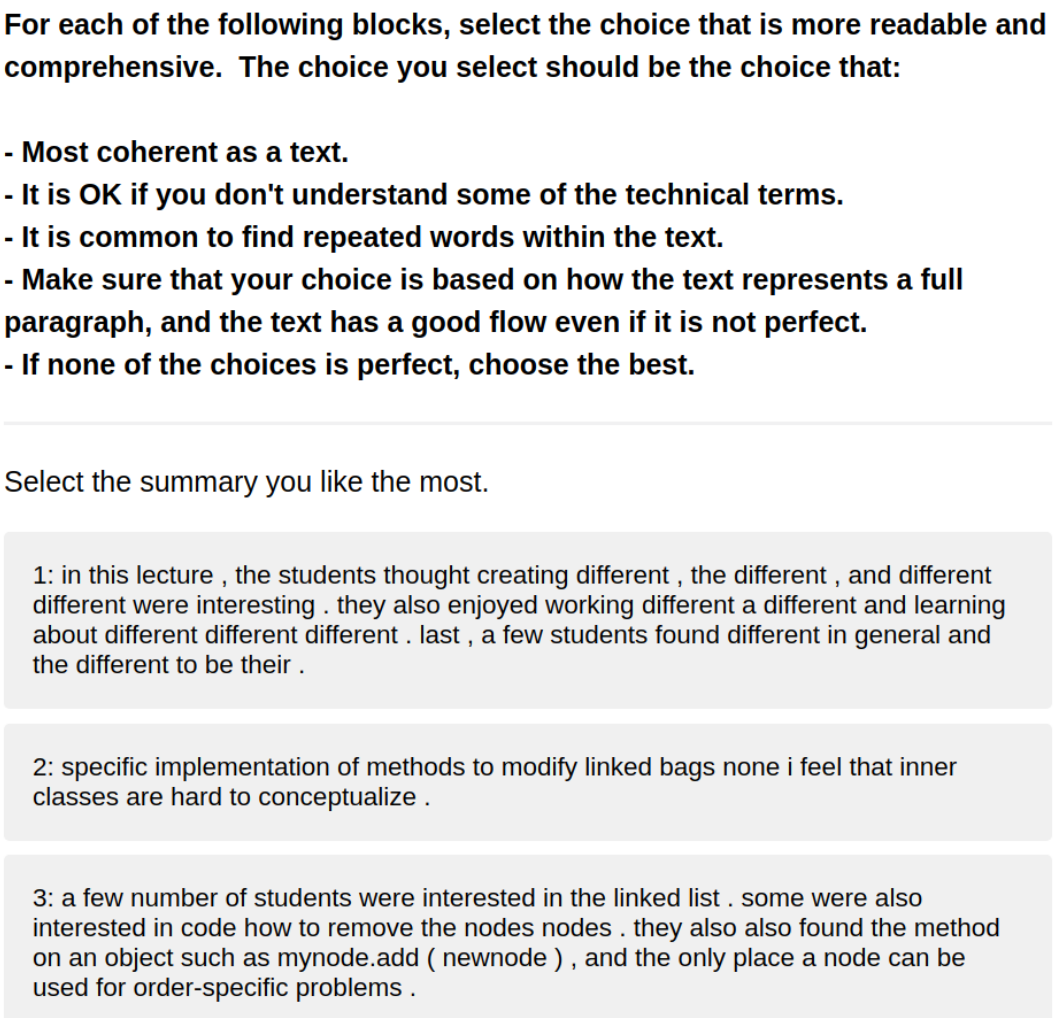


Figure 1: Human evaluation task example.

4.5.2 Human Evaluation Results

While automated evaluation metrics like ROUGE measure lexical similarity between machine and human summaries, humans can better measure how coherent and readable a summary is. We thus perform a small human evaluation of machine-produced summaries for 2 courses, namely Stat2015 and CS. Since the study is small scale, we picked only one of Stat2015 and Stat2016 since they are the same course over two years and thus similar. We picked CS over ENGR based on the number of documents. Since PG-net produced higher ROUGE scores than Fast-RL, we perform the evaluation using only summaries from PG-net.

Our evaluation study investigates whether tuning the PG-net model increases summary coherence, by asking evaluators to select which of three summaries for the same document they like most: the PG-net model trained on CNN/DM; the model trained on student reflections; and finally the model trained on CNN/DM and tuned on student reflections. 20 evaluators were recruited from our institution and asked to each perform 20 tasks similar to Figure 1. Summaries are presented to evaluators in random order. Evaluators are then asked to select the summary they feel to be most readable and coherent. Unlike ROUGE, which measures the coverage of a generated summary relative to a reference summary, our evaluators don't read the reflections or reference summary. They choose the summary that is most coherent and readable, regardless of the source of the summary.

Table 19 shows the percentage of evaluators selecting summaries from each model. For both courses, the majority of selected summaries were produced by the tuned model (49% for CS and 41% for Stat2015). The results of our human evaluation task provide further evidence that domain transfer to remedy the scarceness of in-domain data can be helpful for improving performance.

4.6 Conclusion

We explored improving the performance of neural abstractive summarizers when applied to the low resource domain of student reflections using three approaches: domain transfer,

Model selection ratio			
Course	**News**	**Reflections**	**Tuned**
CS	31	19.7	49.3
Stat2015	30.9	28.5	40.5

Table 19: Model selection % (all human evaluations).

data synthesis and the combination of both. For domain transfer, two state of the art abstractive summarization models were pretrained using out-of-domain data (CNN/DM), then tuned using in-domain data (student reflections). The process of tuning improved ROUGE scores on the student reflection data, and at the same time produced more readable summaries. To incorporate synthetic data, we proposed a new template based synthesis model to synthesize new summaries. We showed that enriching the training data of the neural summarizers with this synthesized data can further increase the benefits of using domain transfer / tuning to increase ROUGE scores. We additionally showed that the proposed synthesis model outperformed a word replacement synthesis baseline.

5.0 Mitigating Data Scarceness through Data Synthesis, Augmentation and Curriculum for Abstractive Summarization (Data-Based Direction Cont.) (Published in Findings of EMNLP 2021) [59]

5.1 Introduction

The previous chapter showed that pretraining with data from a different domain and synthesizing data using a template-based model could improve abstractive summarization models for low resource domains. In this chapter, we expand the scope a little bit more. We explore different methods of data augmentation and different approaches to improving the used pretraining-fine tuning pipeline. In this chapter, we focus on two different directions; data augmentation and curriculum learning. Data augmentation is widely used in computer vision to increase the model resistance for slight input variants while increasing the training data size.

On the other hand, for the textual domain, data augmentation tends to be more challenging due to the text's structural nature. Most of the work done on data augmentation focus on back translation and word replacement. These two approaches involve augmentation on the input level, which might expose the model to grammatically or logically incorrect input. Additionally, the data augmentation benefits are distributed across the whole model, which might involve already well-trained or less critical parts. Thus, we decided to explore a different approach that can avoid the grammatical and logical constraints on the input side; it can also allow us to push the learning more in-depth into the model to any specific part. The second direction we explore is integrating curriculum learning in the training process. Curriculum learning can help weighting training samples differently based on external criteria. Weighting training samples, in turn, can help training the model gradually. Finally, we explore combining both directions (augmentation and curriculum learning) to overcome the data scarcity issue. The experiments performed in this chapter aim to answer the following research questions:

- **Question 1 (Q1)** : Would the proposed shuffling and paraphrasing data synthesis model

be helpful even without reliance on templates ?

- **Question 2 (Q2)** : Would sample mixing and curriculum learning introduce any improvements for abstractive summarization on low resource domains ? What difficulty metric to use ?
- **Question 3 (Q3)** : Would combining data synthesis with curriculum learning further outperform using each technique in isolation ?
- **Question 4 (Q4)** : Finally, can the proposed shuffling and paraphrasing synthesis technique generalize to more structured data ?

5.2 Datasets

CourseMirror (CM) In this chapter, we use the Train/Val/Test variant of CM data. In order to use the Train/Val/Test variant of CM data for training, we compiled all courses into one dataset. We then split the documents into training, validation, and test sets (80%, 10%, 10%, respectively) by sampling equally from all courses. Refer to chapter 2 for further details.

Amazon/Yelp (A/Y) is the other dataset we use in this chapter. The dataset is another small dataset, consisting of opinions. The dataset contains customer reviews from Amazon [29] and Yelp. The data contains 160 products/businesses split into training, validation and test sets. Each of the products/businesses contains a set of 8 reviews. Refer to chapter 2 for further details.

5.3 Summarization Models Used

Unlike previous chapter in which we used RNN based models [86, 12], in this chapter we move to using more recent SOTA abstractive summarization models that utilizes transformer networks (BART [41], BERTSum [45]). We decided to use these two models as being two

SOTA models for abstractive summarization and being easy to integrate with the proposed techniques in this chapter.

5.4 Proposed Approach

This section describes the approaches we explore in the two directions (data augmentation and curriculum learning).

5.4.1 Data Augmentation

We explore different data augmentation approaches. First, we introduce a couple of simple approaches as baselines and compare them with more sophisticated approaches. We then build on the template-based model discussed in the previous chapter by proposing synthesising data using a pretrained paraphrasing model instead of templates. Finally, we introduce the proposed variant of MixText that we use to augment data for text generation tasks.

5.4.1.1 Baselines

In our experiments, we focus on summarizing student reflections as in CourseMirror and customer reviews such as Amazon/Yelp. The input document consists of several independent reflections/reviews. Thus we decided to explore a couple of data augmentation approaches that can benefit from such property. To our knowledge, in prior work there is no data augmentation technique used for summarization except back generation proposed by Parida and Motlicek [72]. Accordingly, we introduce these two simple techniques to be used as baselines. First, we randomly shuffle the reflections/reviews to generate a new input document. For an input document and its corresponding summary D and S respectively where D consists of N independent reflections/reviews $[r_1, r_2, r_3, .., r_n]$, we generate M augmented samples $[D_{a1}, D_{a2}, D_{a3}, ..D_{aM}]$ by randomly shuffling the reflections in the input document so that $for\ i \in [1..M],\ D_{ai}, S = shuffle(D), S.$ where,

$shuffle(D) = shuffle([r_1, r_2, r_3, .., r_n])$ returns a randomly shuffled version of the passed set of reflection/reviews such as $[r_2, r_{n-3}, r_4, .., r_1]$ and so on. By doing so, we try to reduce the model's dependency on the order of reflections/reviews, as we pass the same set of reflections/reviews and expected summary while having the reflections/reviews reordered differently.

The second approach is shuffling with masking. Like the first approach, we generate M augmented samples for each input document and corresponding summary by shuffling the reflections/reviews. However, we additionally randomly allow for input masking such that 50% of the reflections are masked. We generate M augmented samples $[D_{a1}, D_{a2}, D_{a3}, ..D_{aM}]$ for an input document and its corresponding summary D and S, respectively by randomly shuffling the reflections/reviews in the input document. We then randomly decide to mask 50% of the reflections/reviews or not by randomly sampling a value v between 0 and 1.

$$for\ i\ \in\ [1..M],\ D_{ai} = \begin{cases} D_{ai}[1 : N/2], & \text{if } v \geq 0.5 \\ D_{ai}, & \text{otherwise} \end{cases}$$

We do so to allow the model to generate the summary from a partial set of reflections/reviews and reduce order dependency.

5.4.1.2 Paraphrasing with GPT-2

Like data synthesis with a template-based model, we propose synthesizing new human summaries using paraphrasing instead of templates. Using templates for augmentation can involve implicit reliance on textual properties that might not be present in all domains. Unlike opinion or reflection summarization, news or scientific paper summaries, for example, might not have an underlying structure that can be common in many summaries. Thus, relying on template extraction might not lead to a successful augmentation. With that said, instead of extracting templates, we propose using a paraphraser to generate different potential summaries that are paraphrases of the original human summary. We use the paraphraser trained by Krishna et al. [39]. They fine tuned a pretrained large GPT-2 language model with data from PARANMT-50M [96] to direct the model into generating diverse paraphrases that they later used for style transfer.

5.4.1.3 Mixtext for text generation (MixGEN)

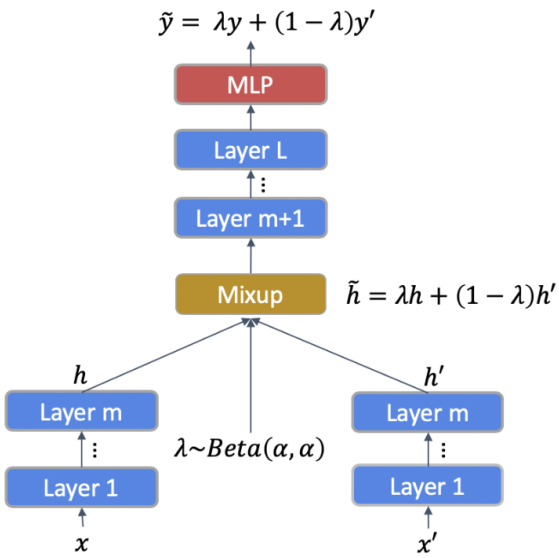


Figure 2: MixText model.

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. Figure 2 shows the original MixText model proposed by Chen et al. [9]. We adapt the approach of Chen et al. [9] for text generation tasks by modifying the decoding process and loss calculation; we call our approach MixGEN (Figure 3).

Similar to 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

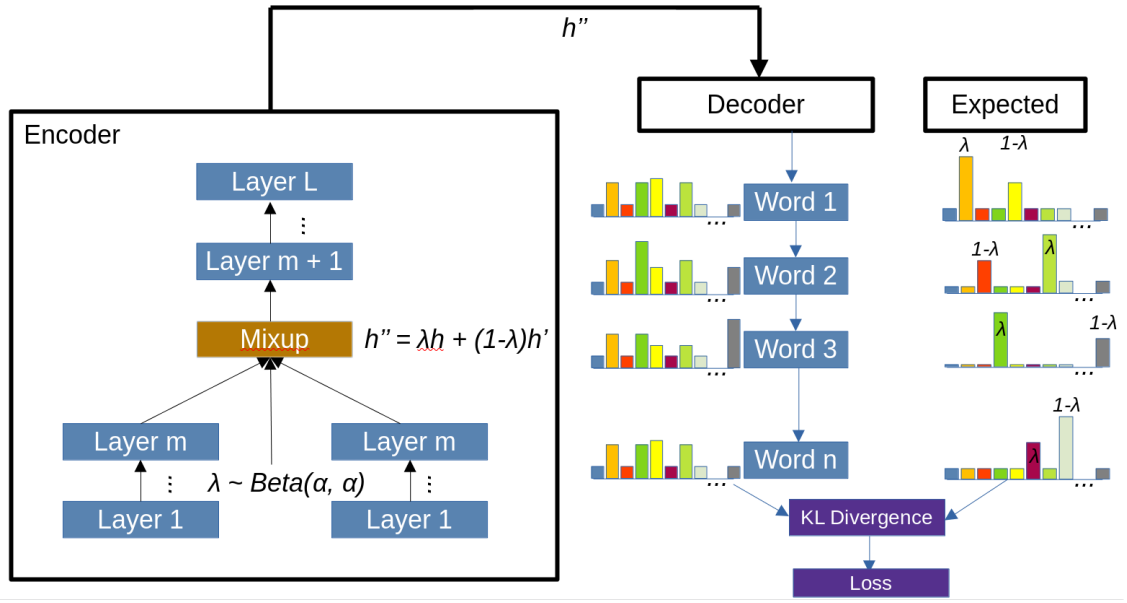


Figure 3: MixText for generative tasks.

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, respectively. PD_i is the probability distribution expected for token$_i$. $S_1[i]$, $S_2[i]$ are the i^{th} 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 i^{th} token of the first sample's human summary, and another value of $1 - \lambda$ at vocabulary token corresponding to the i^{th} 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 (λ) vs. (1-λ). Thus, we randomly sample from the two input samples based on their weights using the following equation.

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

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.

5.4.2 Curriculum Learning

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 by Xu et al. [98]. In this approach, we split data into N buckets based on a difficulty metric. We then train the model in a difficulty incremental setting by sampling from harder buckets as the training process progresses while keeping the easy ones. After a while, the whole data is then used for training. This would allow the model to focus on easier samples first, learn them then move to the harder ones. In this work, we use two different curriculum difficulty metrics to infer which measures can be more suited to our domain.

5.4.2.1 Specificity

Specificity is a measure of 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, or Everything is Easy* are not specific and easy for the model to learn and vice versa. Thus, we decided to use specificity as a pointer to how hard or easy it is to learn a piece of text. 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 C) 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 consisting 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.

5.4.2.2 ROUGE

ROUGE is the standard metric used for evaluating text summarization performance. Thus, following Xu et al. [98], we decided to use ROUGE as a difficulty measure. In their original approach Xu et al. [98] used the downstream's task evaluation metric as the difficulty measure for their curriculum. We decided to use ROUGE scores by measuring the input-output ROUGE. In this method, for a training sample, we calculate the ROUGE score between the input document D, and it is corresponding human summary S and use it as a difficulty measure. We can think of the ROUGE value between input D and summary S, as to how abstractive the summary is. According to Liu* et al. [44], the higher the ROUGE score, the less abstractive the summary is compared to the input, and vice versa. We argue that the more abstractive a sample (i.e., less ROUGE score) is, the harder the sample is to learn.

5.5 Experiments

5.5.1 Parameters

5.5.1.1 Baselines

To our knowledge, in prior work there is no data synthesis technique used for summarization except back generation [72] and template synthesis [58]. 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.

5.5.1.2 Paraphrasing with GPT-2

We used the paraphrasing model trained by Krishna et al. [39] [1]. 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.

5.5.1.3 MixGEN

We integrate MixGEN by combining each sample with n other samples during training. We varied n between [3, 5, 7][2, 3] for our experiments. Moreover, we use mixing probability α=0.75 as specified by the original code implementation[4].

[1]https://drive.google.com/drive/folders/1RmiXX8u1ojj4jorj_QgxOWWkryDIdie-?usp=sharing

[2]We found that increasing n more than 3 leads to performance degradation. Thus, we only show results of n=3 here.

[3]Integrating MixGen with BART is complex, and since experiments on BERT didn't introduce huge improvements, we decided to perform experiments only using BERT.

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

5.5.1.4 Curriculum learning

In the curriculum learning experiments, we use a specificity prediction model that consists of a DistilBERT [83] encoder with a support vector machine classification layer (refer to Appendix C for further details and model evaluation). The model is trained using CM specificity data (refer to chapter 2) and predicts values between 1 and 4. We calculate the whole input document's specificity value by averaging the specificity values of all the reflections. We normalize the whole training data values between 1 and n, where n is the number of buckets to split the data. Following the work done by Xu et al. [98] we use n=10. Similarly, we normalize the average ROUGE value to also be between 1 and n.

5.5.2 Model Training

In all of our experiments, we use BERTSum and BART models. We fine tune the models using 6 Nvidia Quadro RTX 5000 GPUs. For BERTSum, we use the BERTSum pretrained on CNN/DM checkpoint[5]. We used the same parameters in the original code. We conducted experiments on CM and AY datasets using proposed methods in regular training and in a **pretraining→fine tuning** setting, where we perform pretraining with synthesized data and fine tuning using original data.

In BART experiments, we use the DistilBART model trained with CNN/DM data, consisting of 6 layers on both encoder and decoder sides and a hidden size of 4096. We use a batch size of 1 sample per GPU. We use a maximum input length of 1024 tokens with right-side padding for shorter inputs and we truncate longer inputs.[6]. We generate an output of a maximum length of 128 tokens on the decoder side. The generation is done using beam search with a beam size of 10 beams. Similar to BERTSum, we conducted experiments on CM and AY datasets using proposed methods in regular training and in a **pretraining→fine tuning** setting, where we perform pretraining with synthesized data and fine tuning using original data.

[5]https://github.com/nlpyang/PreSumm

[6]The hardware couldn't accommodate larger batch sizes, and the maximum input length of 1024 is the maximum length BART model can process. For CM, we only have 8 inputs longer than 1024 tokens and none for AY.

5.6 Results

		CM			AY		
Pretraining	fine tuning	R1	R2	RL	R1	R2	RL
No Pretraining							
None	Original	36.34	11.39	26	27.71	3.83	17.83
	shuffle	38.57	11.72	26.94	28.34	4.04	17.74
	shuffle + mask	37.07	11.52	26.45	28.01	4.21	17.87
	Synth.(n=5)	39.39	12.85	26.66	28.27	4.36	17.84
	Synth.(n=10)	**41.14**	**14.24**	26.98	28.49	**4.54**	18.08
	Cur.(S)	36.88	12.41	**27.63**	28.69	4.28	17.95
	Cur.(R)	37.01	12.13	27.11	**28.8**	4.33	**18.18**
	Mix(n=3)	36.87	11.98	26.57	27.85	3.95	17.89
With synthetic data pretraining							
Synth.(n=5)	Original	39.39	12.85	26.66	28.27	4.36	17.84
	Cur.(S)	40.68	13.59	26.26	27.95	4.4	18.01
	Cur.(R)	39.35	12.33	26.48	**28.56**	4.25	18.06
Synth.(n=10)	Original	**41.14**	**14.24**	26.98	28.49	**4.54**	18.08
	Cur.(S)	39.81	12.94	26.81	28.52	4.5	**18.15**
	Cur.(R)	40.22	14.12	**27.33**	28.21	4.35	17.87

Table 20: ROUGE results of BERTSum with augmentation techniques on CM and AY (highlighted means outperform original, and **bold** means the best scores across a set of experiments)

Table 20 shows results obtained through conducting experiments on CM and AY datasets. Considering **data synthesis and augmentation** (answering Questions **Q1, and Q2**), 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 AY 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), 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

		CM			AY		
Pretraining	fine tuning	R1	R2	RL	R1	R2	RL
No Pretraining							
None	Original	48.55	22.56	34.23	37.17	10.08	23.1
	shuffle	52.08	30.05	**40.2**	35.8	8.15	21.79
	shuffle + mask	**53.37**	**30.09**	39.37	36.19	8.75	22.51
	Synth.(n=5)	48.26	23.56	35	37.84	**10.36**	23.84
	Synth.(n=10)	49.39	23.35	35	**38.15**	10.34	**23.88**
	Cur.(S)	46.84	20.91	34.29	37.94	10.2	23.75
	Cur.(R)	48.2	23.33	34.39	37.83	10.24	23.64
With synthetic data pretraining							
Synth.(n=5)	Original	48.26	**23.56**	**35.05**	37.84	**10.36**	23.84
	Cur.(S)	46.51	22.23	33.01	36.27	8.98	22.78
	Cur.(R)	47.01	21.2	33.73	37.24	9.36	23.3
Synth.(n=10)	Original	**49.39**	23.35	35	**38.15**	10.34	**23.88**
	Cur.(S)	47.35	22.42	32.76	37.17	9.88	23.31
	Cur.(R)	48.63	21.94	34.24	37.65	9.27	23.22

Table 21: ROUGE results of BART with augmentation techniques on CM and AY (highlighted means outperform original, and **bold** means the best scores across a set of experiments)

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 AY. Additionally, we can see that increasing the synthetic data size helps to improve the model performance across all ROUGE scores for both CM and AY datasets (N=5 vs. N=10).

Now moving to **curriculum learning** (answering Question **Q2**), 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 AY 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, answering question **Q3**, we don't see consistent ROUGE score improvement when using curriculum for fine tuning after pretraining with synthetic data. Furthermore, while both curriculum difficulty metrics (i.e., Specificity and ROUGE) introduced improvement

compared to training with no curriculum, we didn't observe any consistent improvement pattern in using one metric over the other.

To further verify our findings, we performed similar experiments using BART model. Table 21 show results obtained through conducting experiments on CM and AY datasets. Similar to BERTSum, we found that different proposed augmentation techniques (i.e., data synthesis and Mixing) can introduce improvements to training with no augmentation except R1 on data synthesis for CM. Moreover, unlike BERTSum, the shuffling and shuffling with masking baselines were able to introduce large improvements on CM but failed to introduce any improvements on AY. Moving to curriculum learning, similar to BERTSum, integrating a curriculum learning using any of the two proposed difficulty metrics can lead to consistent improvements across all ROUGE scores for the AY dataset. However, for CM, we can see improvements on R2 and RL using the ROUGE metric and improvements on RL only using specificity metric. We counted the number of instances (i.e., R1, R2, or RL) where the curriculum with the ROUGE metric achieved a higher score than the curriculum with the specificity metric. We found that using the ROUGE metric outperforms the specificity metric 61% of the time compared to 39% for specificity. We also calculated the average difference in scores between curriculum with ROUGE and curriculum with specificity for instances when curriculum with ROUGE is higher and instances when curriculum with specificity is higher. We found that the average difference in score for instances in which curriculum with ROUGE is higher is 0.7, while for instances in which curriculum with specificity is higher is 0.48. Thus, while both metrics are helpful, we can conclude that the more relevant ROUGE metric can introduce higher and more consistent improvements. Finally, integrating curriculum learning with data synthesis for BART didn't introduce any improvements compared to training with only synthetic data.

To conclude our findings, we summarized our results in table 22. In table 22, we show each finding we observed in our experiments and mark whether the finding is satisfied (Y) for each of the datasets and models or not (N). Y indicates that condition is satisfied for all 3 ROUGE scores and N is otherwise (e.g., Syn(5) outperforms original is marked with Y if for all 3 ROUGE scores (R1, R2, RL) Syn(5) achieves greater or equal value than original). Finally, we assign each Finding a generalization degree (strong, moderate, weak, and none)

that indicates how much the finding transfer across data, models, and both. Each degree indicates a different number of satisfied conditions *(A condition is a combination of a model and a dataset, refer to table 22 for the four conditions)*, where strong indicates complete transferability across data and models (i.e., all four conditions are satisfied), moderate (i.e., 3 out of 4 conditions are satisfied), weak (i.e., 1, or 2 out of 4 conditions are satisfied) and finally none (i.e., 0 out of 4 conditions are satisfied). We can see that for single technique usage (only synthetic and only curriculum), the proposed synthetic with paraphrasing technique outperforms using only original data across the two models and the two datasets (i.e., all 8 conditions) except for one condition (answering questions **Q1, Q2**). Similarly, using a curriculum with ROUGE or specificity difficulty metric outperforms using data without any curriculum for three out of the four conditions (answering question **Q2**). Finally, we can see that combining curriculum learning with data synthesis didn't outperform using only synthetic data for any of the four conditions.

Findings	BERTSum		BART		Generalization
	CM	AY	CM	AY	
Single Technique					
Shuffle baseline outperforms Original	Y	N	Y	N	Weak
Shuffle + mask baseline outperforms Original	Y	Y	Y	N	Moderate
Cur.(S) outperforms Original	Y	Y	N	Y	Moderate
Cur.(R) outperforms Original	Y	Y	N	Y	Moderate
Syn.(5) outperforms Original	Y	Y	N	Y	Moderate
Syn.(10) outperforms Original	Y	Y	Y	Y	Strong
Syn.(10) outperforms Syn.(5)	Y	Y	N	N	Weak
Multiple Techniques					
Syn.(5) + Cur.(S) outperforms Syn.(5)	N	N	N	N	None
Syn.(5) + Cur.(R) outperforms Syn.(5)	N	N	N	N	None
Syn.(10) + Cur.(S) outperforms Syn.(10)	N	N	N	N	None
Syn.(10) + Cur.(R) outperforms Syn.(10)	N	N	N	N	None

Table 22: Summary of results. Generalization indicates how the findings transfer across models, data, and both. Y indicates that condition is satisfied for all the 3 ROUGE scores and N otherwise. Strong, moderate, weak, and none indicate the number of satisfied conditions of (4, 3, 2 or 1, and 0) respectively.

5.7 Shuffling and Synthesis Analysis (Answering Question Q4)

In our experiments, we observed a difference between the performance of shuffle baseline when performed on CourseMirror data and Amazon/Yelp data. Table 23 shows the results of BART model on both CM and AY data using original data and original data with shuffling baseline **(numbers are extracted from table 21)**. We can see that the shuffling on CM introduces good improvements; however, when we moved to AY data, shuffling achieved worse performance than using only original data. One hypothesis is that CM, on average, has large numbers of reflections per document (40) while AY only has eight reviews per document. Having many reflections allows the shuffling to produce potentially different examples. However, with eight reviews only, shuffling may not be able to change the document noticeably.

Data	Configuration	R1	R2	RL
CM $\sim$ (40 Refs/Doc)	Original	48.55	22.56	34.23
	shuffle	**52.08**	**30.05**	**40.2**
AY = (8 Refs/Doc)	Original	**37.17**	**10.08**	**23.1**
	shuffle	35.8	8.15	21.79

Table 23: ROUGE results of BART model on CM and AY data for shuffle baseline.

In order to verify this hypothesis, we decided to create a new version of CourseMirror data. We randomly sample 8 reflections only per document in this version, and we refer to that version as (CM-8). Sampling only eight reflections produces a version of CM similar to AY data in terms of input document size. We then performed experiments to fine tune BART with CM-8 using only original data and data with shuffling. Table 24 shows the results of fine tuning BART on both CM and CM-8 using original data and shuffled data. We can see that our hypothesis can hold, and we can see that by reducing the number of reflections to only 8, the improvement in ROUGE scores decreased a lot. Moreover, we found that for R2, the shuffling didn't introduce any improvements.

Data	Configuration	R1	R2	RL
CM $\sim$ (40 Refs/Doc)	Original	48.55	22.56	34.23
	shuffle	**52.08**	**30.05**	**40.2**
CM-8 = (8 Refs/Doc)	Original	41.58	**15.86**	28.55
	shuffle	**42.1**	13.73	**28.91**

Table 24: ROUGE results of BART model on CM and CM (8 reflections) data for shuffle baseline.

Data	Configuration	R1	R2	RL
CNN $\sim$ (28 Sents/Doc)	Original	**34.33**	**13.14**	**23.58**
	shuffle	33.13	12.71	23.22
	Syn 5	32.66	11.26	22.21
CNN - small	Original	16.37	2	10.61
	Syn 5	**19.48**	**2.55**	**12.76**

Table 25: ROUGE results of BERTSum model trained with real, shuffled, and synthetic data from both CNN and CNN-small datasets.

From the previous experiments (refer to section 5.6), we can conclude that proposed data synthesis technique that depends on shuffling, in general, can introduce improvements for datasets such as CM and AY, where the input is unstructured, and all reflections/reviews are independent. However, we can't guarantee the same behavior if applied to structured data such as news. Thus, we performed another set of experiments to verify how shuffling and shuffling based synthesis would affect more structured data. We performed experiments using a subset of CNN/DM dataset to compare training with original data and training with shuffled data. We randomly sampled 20k samples from CNN and 20k samples from DM data for a total of 40k samples. The average number of sentences per document is around 28 sentences. We then shuffled and randomly sampled 25K samples from the 40k for training and 5k for validation, and 10k for testing. For each sample of the training samples, we produced five more augmented samples via shuffling the input document's sentences and using the same ground truth summary. We then trained the BERTSum model once using only the original 25k samples and another time using the 25k original + 125k augmented samples. Table 25 shows the results of training using both versions of the CNN/DM datasets. We can see that training the model with shuffling didn't introduce any improvements to CNN; however, we can conclude that shuffling is not very harmful even with structured data as we observed a slight reduction in all ROUGE scores.

Now, we move to analyze the effect of shuffling combined with paraphrasing data synthesis. We perform data synthesis with paraphrasing similar to what we did with CM and AY. We generate five paraphrases for each training data sample as alternative human summaries. We then shuffle the input document five times and combine each of the shuffled versions with one of the generated paraphrased summaries as a new synthetic training sample. Unlike unstructured data, for which the data synthesis mechanism was able to help the model improve, using the same shuffling-paraphrasing-based data synthesis approach didn't help the model improve with more structured data. From table 25 we can observe that using the data synthesis approach led to a drop in all ROUGE scores. This might be due to the training sample size we used for the CNN/DM dataset, which is 25K samples. That size can't be considered low resource data, and introducing synthesized data with low quality was harmful to a model that can be trained well enough from 25K samples. To further verify

that hypothesis and confirm the effect of shuffling on structured data, we sampled a smaller version of the CNN/DM dataset. We only sampled 1K samples for training and 500 samples for both validation and testing. According to table 25, we can see that using the proposed shuffling and paraphrasing synthesis technique was indeed helpful for low resource settings, even if more structured data such as news was used. Additionally, as we hypothesized, the synthesis is not effective when much data is available, as it would introduce low-quality data that would be harmful to the model when much high-quality data is already available.

5.8 Conclusion

This chapter 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 and two different models (BERTSum, and BART). We also showed that mixing samples for training can push the BERTSum model to be more resilient to overfitting and improve its performance. Moreover, 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). However, we showed that using ROUGE difficulty metric outperforms using specificity metric in higher number of cases. Finally, we showed that using any of these data manipulation techniques in isolation can improve the performance across two datasets using two different models. Unfortunately, we found that combining two of these techniques (synthesis + curriculum) either didn't introduce consistent improvements or no improvements at all. We also showed that the proposed synthesis (paraphrasing and shuffling) technique could also be helpful in low resource settings not only for reflections-like data but also for structured data such as news.

In the next chapter, we are moving towards a different direction (i.e., model-based direction), where we perform experiments on multitask learning. Finally, in the last chapter,

we combine both directions (i.e., data-based and model-based) into one framework targeting low resource domains.

6.0 Improving Abstractive Summarization With Multitask Learning (Model Based Direction) (Published in Findings of EMNLP 2021) [60]

6.1 Introduction

Recent work has shown the utility of training neural models using huge amounts of data from various tasks. For example, research using T5 [77] and BART [40] 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. Research has shown that other multitask learning models can also help a model better learn a target task. 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 [33, 11, 73, 23]. In their work Gehrmann et al. [23] showed that training extractive summarization in addition to abstractive summarization helps to enhance abstractive summarization performance. Additionally, all of the summarization multitask work is using large summarization datasets. To our knowledge, no prior work has tried to tackle multitask summarization in low-resource domains with little training data.

In this chapter we attempt to address these gaps by answering the following research questions:

- **Question 1 (Q1)** : Can abstractive summarization performance be boosted via multi-task learning when training from a small dataset?
- **Question 2 (Q2)** : Are there some tasks that might be helpful and some that might be harmful for multitask abstractive summarization?
- **Question 3 (Q3)** : Will the same findings emerge if a very different small training corpus is used?
- **Question 4 (Q4)** : Will the same findings emerge if different learning models are used?
- **Question 5 (Q5)** : Finally, what would be the outcome if we used both different model and data?

These five questions aim to find the effect of incorporating multitask learning with multiple tasks on abstractive summarization for low resource data. Additionally we aim to get an initial understanding of which tasks are helpful to abstractive summarization and which are harmful. Finally, these questions try to verify if the behavior is consistent across different conditions (Model type, pretraining, data structure). To answer Q1, we use a pretrained BERT model [14] 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, we replicate the student reflection experiments using a very different corpus of news documents. To answer Q4, instead of fine tuning with the BERT model, we perform experiments using the BART and T5 transformer models [77, 41]. Finally, to answer Q5 we perform experiments using BART and T5 on a different small dataset (i.e. Amazon/Yelp). 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 tasks like language modeling more often degrade model performance when used as an auxiliary task coupled with low resource data.

6.2 Datasets

Table 26 summarizes each dataset in terms of the number of documents and their distribution into training, validation, and test sets. Chapter 2 contains examples from each dataset alongside a more comprehensive distribution of each of the datasets.

CourseMirror (CM)[1] In this chapter we perform experiments with CM data using the two configuration Leave-One-Course-Out (LOCO) and the traditional Train/Val/Test splits (refer to chapter 2). For both BERT and T5 we use the LOCO configuration. On the other hand, we use the other configuration for fine tuning BART.

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

CM LOCO				
Train + validation data	**Test data**	**Train**	**Val**	**Test**
ENGR + S2015 + S2016 (232)	CS (138)	209	23	138
CS + S2015 + S2016 (318)	ENGR (52)	286	32	52
CS + ENGR + S2016 (282)	S2015 (88)	254	28	88
CS + ENGR + S2015 (278)	S2016 (92)	250	28	92

Other datasets				
Data	**# Docs**	**Train**	**Val**	**Test**
Complied CM	370	296	37	37
Amazon/Yelp	160	58	42	60
CNN-micro	2500	1500	500	500

Table 26: Dataset summary.

Amazon/Yelp[2] In our experiments, we use the provided Train/Val/Test splits to fine tune both T5 and BART models.

CNN/DailyMail summarization dataset (CNN-micro) CNN/DM is a widely used summarization dataset consisting of around 300k news-oriented documents (refer to chapter 2). Since the focus of our research is low resource data, we randomly select a small version of CNN/DailyMail, and we refer to it as CNN-micro. To build CNN-micro, we randomly sampled 500 samples for test and validation from the corresponding CNN original test and validation sets. We then randomly sample 1500 documents for training from the original training set. We only used the CNN-micro dataset for BERT experiments. The reason is that CNN/DM is used as part of a huge corpus used for pretraining both T5 and BART. Thus, testing with CNN dataset would be inaccurate.

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

6.3 Summarization Models Used

In the previous chapter, we performed experiments using two of the SOTA abstractive summarization models (BART [41], and BERTSum [45]). In this chapter, we decided to pick BART out of the two models as it showed better performance across all experiments and is easier than BERTSum to integrate within multitask learning scheme. Additionally, we decided to use two additional models to verify our findings. The first one is T5 which is like BART, a model trained with multiple tasks and is easy to integrate with a multitask learning scheme. Moreover, we decided to propose another model unlike BART and T5, which are pretrained in a multitask learning setting. The proposed model simply uses a BERT encoder followed by task-specific sub-modules.

6.4 Proposed Models

This section describes the three summarization models used in this work, the different tasks used for multitask learning, and the intuition behind using each of the tasks. We first introduce the base encoder used for each model. Then we describe how to integrate multitasking with each model, and the basic structure of the different parts of each model.

6.4.1 Model Base

6.4.1.1 BERT

We use a pretrained BERT [14] model as a sequence encoder followed by a set of different task-specific modules.

6.4.1.2 T5-Transformer

We also make use of the recently introduced T5-transformer [77], which stores a large amount of knowledge about language and different tasks, including abstractive summariza-

tion.

6.4.1.3 BART

We finally make use of the recently introduced BART [41], which trains multiple tasks including abstractive summarization with different input manipulation and reconstruction such as (shuffling, masking, etc.).

6.4.2 BERT Multitask Integration

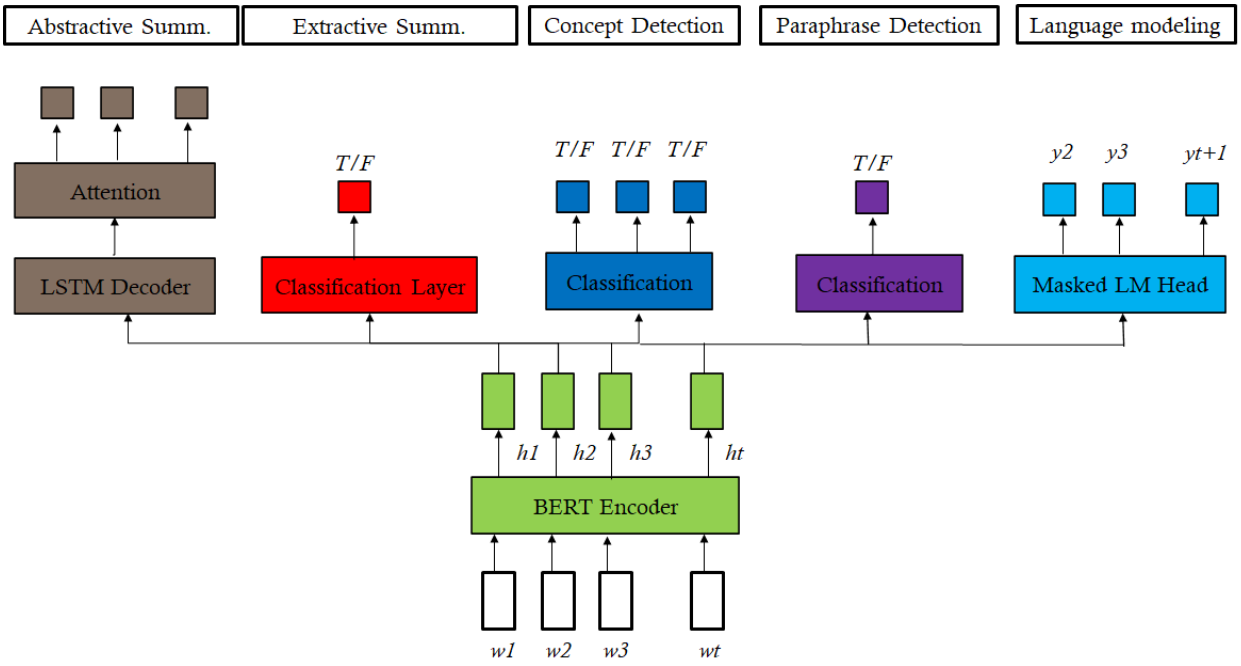


Figure 4: Proposed BERT-Multitask model.

6.4.2.1 Shared BERT encoder

The model uses a pretrained BERT encoder shared between all tasks. Encoder output is passed to task specific sub-models to perform generative and predictive tasks. Encoder weights are also fine tuned alongside the rest of the model in the multitask framework. Figure 4 shows the model architecture.

6.4.2.2 Abstractive summarization

Although a lot of recent abstractive summarization work uses transformer-based models to overcome issues of sequence length [99, 40], we found that LSTM based decoders consistently outperform transformer-based ones when trained from scratch on our data. Thus we decided to use LSTMs in all our experiments. We argue that transformers might require more data to train from scratch compared to LSTMs, however further analysis is needed. While most recent work on text generation tends to use pointer networks [92, 86], we decided to avoid using pointer networks and used only attention and coverage mechanisms [86] to reduce the number of model parameters.

6.4.2.3 Extractive summarization

The extractive summarization module aims to extract the most salient sentences from the input document and present them as a summary. Magooda and Marcjan [61] showed that a non-autoregressive extractive model trained with language modeling can be efficient. The model consists of a linear layer to classify a sentence as part of the summary or not given the input document and the sentence. Document and input sentence are fed to BERT encoder in the format

$$[CLS]DW_1DW_2...DW_n[SEP]SW_1SW_2....SW_m$$

Where DW_i is the i^{th} word of the input document, SW_i is the i^{th} word of the sentence to classify, and ([CLS], [SEP]) are respectively the starting and sentence separation tokens used by BERT.

The output of the classification layer is either 0 or 1; 0 means the sentence is not part of the summary and 1 means otherwise.

6.4.2.4 Concept detection

This task 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 many research papers or books). We assume the same is applicable for summarization; to accurately and efficiently summarize a topic, a prior understanding of the concepts or keywords can help the summarization model locate the important information. The module's objective is to classify each word within a sequence of words as either a part of a concept or not. So for each word w_i in a sequence S of length N, the module produces N predictions $(P_0, P_1, P_2, ..P_N)$, where $P_i = 0$ or 1 for i=0..N; 1 means the word is part of a concept and 0 otherwise. The module consists of a fully connected layer that works as a classifier, where the input is the hidden states generated using BERT encoder. Output on the other hand is either 0 or 1 for each of the words in the input document. Further training details will be presented in sections (6.5.3, 6.5.4, and 6.5.5).

6.4.2.5 Paraphrase detection

The paraphrase detection module aims to classify a pair of sentences as to whether the sentences are paraphrasing each other or not. Which in turn means that both sentences convey the same idea in different wording. Guo et al. [25] proposed training entailment generation with abstractive summarization, as summarization is very relevant to entailment generation. We in contrast view the relation between the input document and generated summaries as a potential paraphrasing. With that said we train a paraphrase detection task. The module consists of a fully connected layer classifier. Similar to extractive summarization, the input is passed to the BERT encoder in the format

$$[CLS]Sent_1[SEP]Sent_2$$

Where Sent_1 and Sent_2 are the two input sentences for MSRP paraphrasing datasets. On the other hand, Sent_1 and Sent_2 are the input document and human summary for CM and CNN-micro datasets. The output of the final classification layer is either 1 if the two sentences are paraphrases and 0 otherwise.

6.4.2.6 Language modeling

We integrate a language modeling module into the multitask training framework. We argue that language modeling, in general, can help improve generative tasks. The language modeling module consists of a masked language modeling attention head. This module aims to skew the BERT vocabulary slightly into the training data distribution by fine tuning it using the masked language model objective. We train the language model module using masked language model objective, in which a subset of the input words are masked and model is expected to predict these words using the surrounding context. Following the original BERT training from [14], input sequence tokens are masked with probability 15%, where masked tokens are either replaced by a special masking token (80%), replaced with a random word (10%) or left unchanged (10%).

6.4.3 T5 Multitask Integration

It is worth noting that unlike BERT, T5 represents any task as language modeling. Thus we decided to drop the language modeling auxiliary task for T5, as it would be a form of redundancy.

Following the T5 framework, we train a set of tasks jointly using the "text to text" format described in the paper. That is, when a text is fed to the model, it is asked to generate a corresponding output in text format as well. To separate tasks from each other, a unique prefix is added to identify each one. For instance, we use "abs_summarize" to distinguish abstractive summarization from the rest of the tasks, and "ext_summarize" to identify the extractive summarization task. In our experiments, we explore utilizing T5 under three experimental conditions.

6.4.3.1 Fine tune T5 on abstractive task

First, we fine tune pretrained T5 transformer on the abstractive task only, using the CM dataset (Figure 5a).

6.4.3.2 Mixture of tasks training

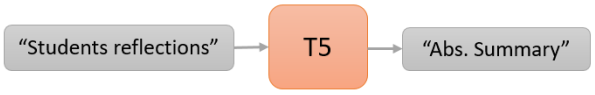
Second, we adopt the T5 framework to train the mixture of tasks as text-to-text, which allowed us to fine tune in the same model simultaneously. Figure 5b shows how we train the combination of tasks in our experiments .

6.4.3.3 Intermediate task transfer

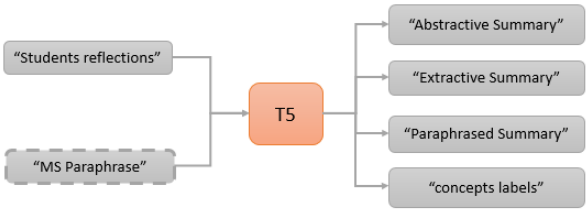
Finally, instead of training multiple tasks jointly with abstractive summarization, we experiment to use them as *intermediate tasks*. First, we *fine tune T5* on a single or a mixture of tasks. Then, we fine tune the resultant model on the main abstractive summarization task. Pruksachatkun et al. [75] studied this idea thoroughly. In their work, they refer to *intermediate tasks* as data-rich tasks. In our work, we use *closely related tasks* as intermediate tasks. Figure 5c shows the process of how we firstly select a task or more to fine tune T5 against, then fine tune again on Abstractive Summarization. Furthermore, T5 is pretrained with CNN/DM, thus we don't perform experiments with CNN-micro using T5.

6.4.4 BART Multitask Integration

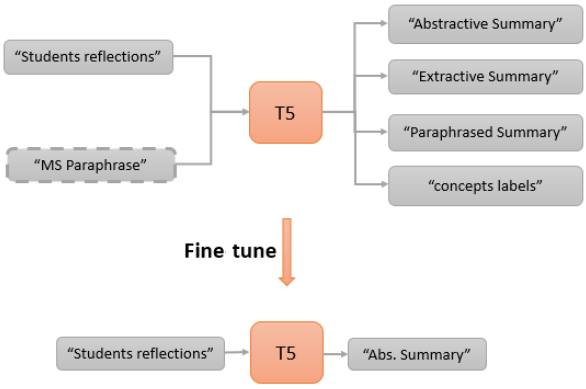
We use the BART encoder-decoder framework to train a set of tasks jointly. Similar to BERT we add 3 additional classification sub modules for concept detection task, paraphrase detection task, and extractive summarization task. The three sub modules operate on the BART encoder output. We use the encoder-decoder modules for both abstractive summarization and language modeling tasks. Finally, instead of training multiple tasks jointly with abstractive summarization, similar to T5 we used them as *intermediate tasks*. First, we *fine tune BART* on a mixture of tasks including abstractive summarization. Then, we fine tune the resultant model on the main abstractive summarization task only. Similar to T5, BART is pretrained with CNN/DM, thus we don't perform experiments with CNN-micro using BART.



(a) Fine tune T5 on Abstracive Summarization Dataset



(b) Fine tune T5 on Mixture of tasks



(c) Intermediate Task transfer using T5 transformer

Figure 5: Different fine tuning conditions for T5. *(- -) indicates optional additive data for Paraphrasing.*

6.5 Experimental Setup

This section introduces data preprocessing, BERT, BART, and T5 model parameters, and evaluation metrics.

6.5.1 Datasets

In addition to using the CM and CNN-micro datasets for abstractive and extractive summarization, when the datasets are used for language modeling all training documents corresponding to a dataset are combined into a single corpus and used for training. Moreover, to train the paraphrase module we combine MSRP data and a summarization dataset. Finally to use a summarization dataset to train the concept detection module, we prepare the concept detection data by initially extracting concepts using a simple TF-IDF ranking algorithm [91]. The algorithm first generates a list of candidate concepts by applying noun phrase chunking. The noun phrases are then ranked using a TF-IDF model trained on Wikipedia. The top N high scoring phrases are selected as concepts. Each word in the input document is then tagged with either 1 or 0, where 1 means the word is part of a retrieved concept and 0 otherwise.

6.5.2 Optimizer

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.

6.5.3 T5 Parameters and Training

We use the *3B T5* model, which is publicly available. The model consists of 24 layers for encoder and decoder. For details of the model architecture, refer to [77]. 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 CourseMirror data on 7 TPUs on Google Cloud for 5000 steps.

6.5.4 BERT Parameters and Training

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-micro 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.

6.5.5 BART Parameters and Training

In BART experiments we use the DistilBART model trained with CNN/DM data which consists of 6 layers on both encoder and decoder sides, and a hidden size of 4096. We fine tune the model using 6 Nvidia Quadro RTX 5000 GPUs. We use a batch size of 1 sample per GPU. Fine tuning is done for 1 epoch using multiple tasks followed by 10 epochs with only abstractive summarization task. The epoch with the highest ROUGE score on the validation set is used later for testing. We use a maximum input length of 1024 tokens with right side padding for shorter inputs. On decoder side, we generate output of maximum length of 128 tokens. The generation is done using beam search with beam size of 10 beams.

6.5.6 Evaluation Metrics

We evaluate performance using ROUGE (1, 2, and L) [42] on F1. Moreover, we perform human evaluation on three different aspects (Fluency, Factual consistency, Relevancy) to overcome the limitations of ROUGE score.

6.6 Results and Discussion

6.6.1 Automatic Evaluation

Our experiments evaluate performance using ROUGE [42] on F1. For CM data we report mean ROUGE using a leave-one-course-out validation for both BERT and T5[3,4], while for CNN-micro, Amazon-Yelp, and BART on CM we report ROUGE using held-out test sets.

Q1-Can abstractive summarization performance be boosted via multitask learning when training from a small dataset?: The gray cells in Table 27 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

[3]Individual course ROUGE scores are in appendix B

[4]We performed pairwise t-test for statistical significance over the results of the four courses. Unfortunately, four samples is a sample size, thus, majority of the scores are not significant

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 27: 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)*

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-Are there some tasks that might be helpful and some that might be harmful for multitask abstractive summarization?: Prior work showed the utility of extractive summarization [31] and language models [61] as auxiliary summarization tasks, and we too observe similar behavior for R1 and R2 in Table 27. 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

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 28: ROUGE results of *BERT* on *CNN-micro*. *(Q3)*

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.

Q3-Will the same findings emerge if a very different small training corpus is used?: We now switch to using different dataset (i.e. CNN-micro). Table 28 shows that when BERT multitask is applied to CNN-micro 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 both paraphrasing and concept detection auxiliary tasks are still good candidates.

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 29: ROUGE results of *T5* (No language modeling auxiliary task) fine tuned on *CM*. *(Q4)*

Q4-W1ill the same findings emerge if different learning models are used?: Shifting gears from changing the data to changing the model, Tables 29 and 30 show that out of the CM findings obtained using BERT multitask, (A P) is the only configuration that performs similarly when different models (BART and T5) are used for CM. Additionally, for T5 combining paraphrasing with the concept task (A C P) introduced improvements on CM. Like BERT, incorporating paraphrasing into BART and T5 helps improve all ROUGE scores when used as a single auxiliary task (A P). On the other hand, the utility of other combinations of tasks didn't transfer from BERT to BART and T5 on CM.

Q5-Finally, what would be the outcome if we used both different model and data?: Finally, we try changing both the data and the model. Table 31 shows the results obtained by using BART model with Amazon/Yelp data. We can see that most of the task combinations except (A L P) improve all ROUGE scores compared to single-task (Abstractive only) training. This shows that similar improvements to BERT on CM data can be seen when using a different model (i.e., BART) on a different dataset (i.e., Amazon/Yelp). Finally, from all the previous experiments, we only found one combination of tasks that introduced improvements consistently across all the configurations (i.e., A P). Thus, to further

Tasks	R1	R2	RL
A	48.55	22.56	34.23
A E	44.21	18.91	30.08
A L	48.54	22.34	34.36
A P	**49.87**	22.59	**35.6**
A C	47.81	20.32	32.96
A E P	45.11	17.35	29.94
A L P	47.11	**23.04**	33.58
A E L	37.18	12.22	25.21
A C P	47.76	21.5	33.43
A L C	48.35	21.44	33.43
A L P C	47.94	21.09	32.94

Table 30: ROUGE results of BART on *CM. (Q4)*

Tasks	R1	R2	RL
A	37.17	10.08	23.05
A L	37.85	10.66	24.03
A P	37.86	10.4	23.95
A C	**38.05**	**10.81**	**24.15**
A L P	36.96	9.45	22.96
A C P	37.52	10.42	23.56
A L C	37.32	10.39	23.57
A L P C	36.46	10.32	24.08

Table 31: ROUGE results of BART on *AY. (Q5)*

Tasks	R1	R2	RL
Single task (A)	34	8.8	21.25
A P	**34.1**	**9.1**	**21.7**

Table 32: ROUGE results of *T5* fine tuned with paraphrasing on *AY. (Q5)*

verify the utility of paraphrasing, we performed another experiment using a combination of a different dataset and a different model. We evaluated the T5 model on the Amazon/Yelp dataset using the (A P) combination only (refer to table 32). Table 32 shows that indeed paraphrasing is again helpful as an auxiliary task, as it improves all ROUGE scores for T5 on Amazon/Yelp.

In conclusion, we can argue that paraphrasing auxiliary task tends to be very helpful across different data (i.e., domains and types) and different models.

6.6.2 Human Evaluation

In addition to automatic ROUGE evaluation, following several recent works [19, 1, 85], we performed a human evaluation on three different aspects (Fluency, Factual consistency, Relevancy) to overcome the limitations of the ROUGE score.

First, we recruited four annotators from our institution (graduate students) with different expertise (i.e., Biomedical and electrical engineering). We performed a pre-task test to assess their understanding of the task. We presented each annotator with four questions; each question consisted of a set of reflections and three handwritten summaries. The three summaries are the original human reference summary and two slightly modified versions of the original summary. Two out of the four questions asked to choose the best summary out of the three provided summaries in terms of relevancy, where we provided a description of relevancy according to Fabbri et al. [19]. The other two questions are similar in format but are directed towards factual consistency rather than relevancy. Figure 6 shows an example

Q1.
Read the following reflections then select the best summary out of (S1, S2, and S3) based on the **Factual Consistency.**

Factual Consistency: is the factual alignment between the summary and the summarized source. A factually consistent summary contains only statements that are entailed by the source document.

Reflections: .
Describe what was confusing or need more details in today's lecture?

- Traversing through trees recursively
- Packages the benefits for the different ways of traversing the trees
- nothing , everything was pretty straightforward
- code for bsts
- In order tracing
- Can a binary search tree be made using a different method of iterating through the tree ?
- Search tree interface
- Why might duplicates be problematic ?
- Nothing was really confusing this lecture.
- Search Tree Interface
- Binary Search Trees
- None
- Everything was pretty clear.
- I had no real questions.
- Nothing was difficult
- Some of the operations

Q1. Select the best summary over all

S1: Most of the students had no problems with the lecture . However, some of them found implementing BSTs and tree traversal as confusing . Some others had problems with search tree interface and packages.

S2: A lot of students had no problems with search tree interface. Some of them found implementing BSTs and tree traversal as confusing. Some others had no problems at all.

S3: Many students had no problems with the lecture. Some students enjoyed learning about implementing BSTs and tree traversal. Some others had problems with search tree interface and packages.

Figure 6: Example of pre human evaluation test

of the four questions from the pre-evaluation test that asks about factual consistency. The four annotators passed the pre-evaluation test, with three out of the four answering all the questions correctly, and only one annotator answered three out of the four questions correctly on the first trial and answered all four on the second trial.

		Factual Consistency	Fluency	Relevancy*
CourseMirror	A	1.83	1.9	1.79
	A P	1.54	1.87	1.75
	A C	**1.96**	**2.42**	**2.21***
	A C P	1.79	2.1	1.67
		Factual Consistency	Fluency*	Relevancy
Amazon/Yelp	A	**2.63**	2.46	**2.5**
	A P	2.46	**2.55**	2.42
	A C	2.46	2	2.38
	A C P	2.5	2.25	2.3

Table 33: Human evaluation scores over (Fluency, Relevancy, and Factual consistency) aspects for both CourseMirror and Amazon/Yelp datasets. **Bold** indicates best score across all tasks for a certain aspect. (*) in header means statistically significant using ANOVA test over all three aspects (i.e. Fluency, Relevancy, and Factual consistency). (*) in cell means statistically significant using paired t-test between combination of tasks (i.e. AC, AP, ACP) and abstractive only

Second, we randomly selected 12 samples from each dataset (**refer to appendix D for examples of annotation samples and pre-evaluation test**). Each sample is annotated by two random annotators out of the four. For each sample, we present annotators with four summaries generated using BART[5] (Abstractive only trained model, Abstractive + Paraphrasing (A P), Abstractive + Concept (A C), and finally Abstractive + concept + paraphrasing (A C P)). The summaries are randomly shuffled, and human annotators are asked to evaluate each summary for (Fluency, Factual consistency, Relevancy) on a scale of

[5]We only sampled summaries from BART as it achieved the best scores on both CM and Amazon/Yelp

	A	A P	A C	A C P
CourseMirror	25 %	16.6 %	**33.3 %**	25 %
Amazon/Yelp	29.16 %	**45.83 %**	4.16 %	20.83 %
All Data	27.08 %	**31.25 %**	18.75 %	22.92 %

Table 34: Percentage of each task output selected by human annotators as best generated summary across all task outputs.

1 to 3. We use the same definitions of (Fluency, Factual consistency, Relevancy) reported by Fabbri et al. [19]. Moreover, we ask the human annotators to pick the best summary overall. We calculate the average human scores across the three aspects for each dataset and combination of tasks. Table 33 shows the average scores for the three different aspects on each dataset. Additionally, table 34 shows the percentage of each task being selected as the best summary across both datasets. Unfortunately, we didn't find major differences in average scores between tasks. Moreover, we performed a one-way (single factor) **ANOVA** test between all four combinations of tasks (i.e., A, AP, AC, ACP) for each individual aspect (i.e., Fluency, Relevancy, Factual Consistency). We only found that the two statistically significant ($\rho \leq 0.05$) aspects are Relevancy for CM and Fluency for Amazon/Yelp. We then performed **paired t-test** between abstractive only and the three combination of tasks (i.e. AP, AC, and ACP) for fluency on AY and relevancy on CM. While in both cases (relevancy on CM and fluency on AY), one of the two proposed tasks (AC for CM and AP for Amazon/Yelp) got higher human scores than the abstractive only summaries. We found that the only statistically significant difference is for relevancy between A and AC on CM. We also found consistent findings when we analyzed the best-selected summaries across datasets (table 34). We found that human annotators preferred summaries generated through training with AC for CM and through training with AP for Amazon/Yelp

6.7 Analysis

In this section, we perform further analysis to get additional insights on models' performance. We perform the analysis on two different dimensions. First, we try to understand how an auxiliary task like concept detection is related to the data properties. Thus, we can estimate the performance of this task on new datasets by looking into its properties. On the other dimension, we perform analysis on the task level to understand how different tasks would lead to change summaries' properties. Thus, we can use specific tasks to generate summaries with certain properties[6].

6.7.1 Concept Distribution

	CS0445	ENGR	S2015	S2016
Input	20.98	17.76	49.49	35.87
Summaries	21.28	21.23	26.19	22.58

Table 35: %Ratio of concept words to total length across reflections and summaries

First we do analysis to get additional insights into different models' performance on datasets and how dataset properties affect the final model performance. We start the analysis by first exploring the effect of concept distribution similarity on the ROUGE score. Table 35 shows the ratio of concepts in input and output for each course in CM dataset. We aim to investigate if having a similar concept distribution between input and output would affect the ROUGE score. We further investigate the effect of combining the task of concept detection with the task of abstractive summarization using two different models (T5, BERT[7]) for the CM dataset. Table 36 shows the results of combining the two tasks using T5 and BERT models on the CM dataset. We can see that for BERT average ROUGE score is improved for (CS0445, ENGR, S2016) and only decreased for S2015.

[6]For now we only focus on abstractiveness property

[7]We did the analysis with T5 and BERT only as we didn't perform experiments with BART using CM leave on course out configuration.

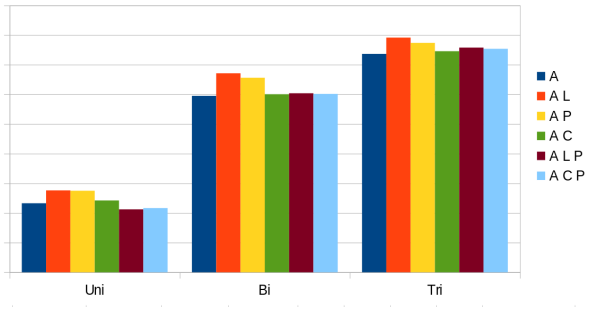
Tasks	R1	R2	RL	AVG	R1	R2	RL	AVG
	BERT							
	CS0445				**ENGR**			
A	26.93	3.98	21.04	17.32	27.19	7.27	22.66	19.04
AC	27.09	4.85	20.12	17.35	30.14	7.67	22.96	20.26
	S2015				**S2016**			
A	27.71	4.83	19.4	17.31	25.46	2.76	22.93	17.05
AC	21.92	3.11	17.75	14.26	29.32	3.4	23.6	18.77
	T5							
	CS0445				**ENGR**			
A	34.62	9.46	29.84	24.64	35.43	9.93	31.07	25.47
A C	34.42	9.71	29.31	24.48	35.84	10.14	31.38	25.78
	S2015				**S2016**			
A	36.87	12.03	32.34	27.08	37.41	12.33	33.02	27.58
A C	34.49	10.40	30.12	25	37.09	12.77	32.42	27.42

Table 36: ROUGE results of BERT and T5 Models fine tuned on CM.

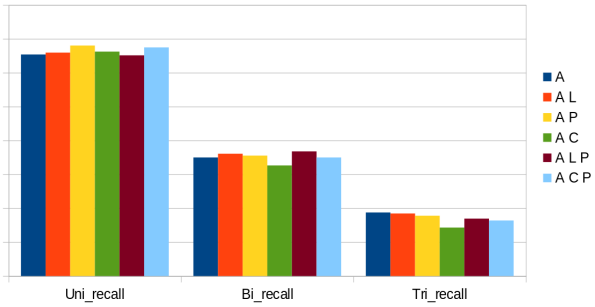
Similarly, in table 35 we can see that the distribution of concepts is similar for CS0445 and ENGR. The distribution is very different for S2015 and less different for S2016. On the other hand, for T5, concepts are only helpful in the case of ENGR and are harmful to the rest of the courses. However, the decrease in the average ROUGE score in CS0445, S2016 is small compared to S2015 course. Finally, we think that while BERT was not affected dramatically by the difference in S2016, T5 was more affected. This can be due to T5 being a more powerful model and able to fit the distribution closely. Thus, we can see that having a similar distribution of concepts in both input and output can improve ROUGE scores through the integration of concept detection task.

6.7.2 Abstractiveness

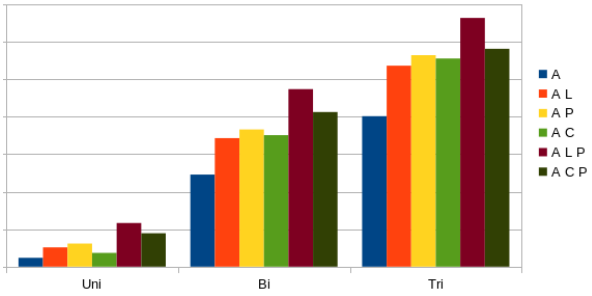
In addition to concept distribution, we performed additional analysis to find which tasks would lead to more abstractive summaries. Abstractiveness in this context refers to models' capability to integrate new words into the generated summaries. In order to measure abstractiveness, we decided to analyze two aspects of generated summaries. First, we look into



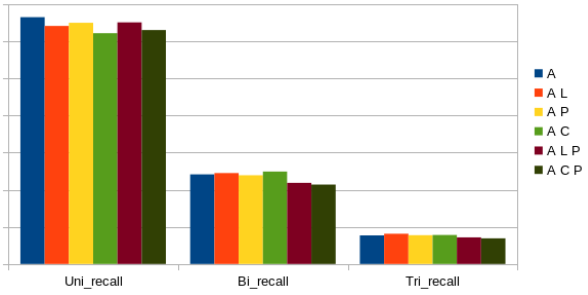
(a) New Ngrams for CM



(b) Ngrams recall for CM



(c) New Ngrams for AY



(d) Ngrams recall for AY

Figure 7: Distribution of new Ngrams and Ngrams recall for both AY and CM datasets

the ratio of new n-grams in generated summaries that don't appear in the input. A higher n-gram ratio indicates the model's ability to add new words that are not copied from the input. However, a higher ratio of new words in isolation doesn't mean the model generates a relevant summary. In order to make sure that the model's tendency to generate new words doesn't lead to generating irrelevant summaries, we also looked into the ratio of human summary n-grams that appear in generated summaries (n-gram recall). Higher n-gram recall indicates that generated summaries are relevant to the input document. Thus, models that achieve high ratios on both aspects can generate more abstractive summaries relevant to the input subject. Figure 7 shows the ratio of new Ngrams in generated summaries by each combination of tasks. Similarly, it shows the Ngram recall for each combination of tasks. We can see that both paraphrasing (A P) and language modeling (A L) help produce more new unigrams, bigrams, and trigrams compared to only abstractive (A) for both CM and AY datasets. Additionally, we can see that both paraphrasing (A P) and language modeling (A L) gets higher Ngram recall than abstractive only (A) except for trigram recall on the CM dataset and unigram recall on the AY dataset. Thus, we can conclude that both the paraphrasing task and language modeling task help abstractive summarization models produce more abstractive and relevant summaries.

6.8 Conclusion

We explored the utility of training a multitask model for abstractive summarization using low resources summarization data. We performed a series of experiments using three fundamentally different models with different preconditions (i.e., BERT not pretrained with summarization dataset versus T5 and BART pretrained with CNN dataset) to verify any observed behavior. We integrated four different tasks in addition to abstractive summarization. We performed 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, three consis-

tently improved ROUGE scores across different types of datasets (CM (reviews/reflections) and CNN-micro (news)) **1- (Abstractive + Paraphrase detection)**; **2- (Abstractive + Concept detection + Paraphrase detection)**; **3- (Abstractive + Extractive summarization + Concept detection)**. Moreover, we found that using **(Abstractive + Paraphrase detection)** is consistent across different datasets (CM, CNN-micro, and AY) as well as different models (BERT, BART, and T5). Additionally, we found that paraphrasing and concept detection, which had not been previously examined as auxiliary tasks for abstractive summarization, can be, in general, beneficial tasks given low resource data. Finally, we found that combining any of (paraphrase detection and language modeling) tasks with abstractive summarization would help the model produce more abstractive summaries than summaries generated via training only with abstractive summarization task.

7.0 Multitask Learning and Data Augmentation for Abstractive Summarization (Combining Data-Based and Model-Based Directions)

7.1 Introduction

In chapters 4, 5, and 6, we explored two main lines of work to improve how abstractive summarization models perform in low resource situations when a handful of samples are available to train the model, and both gathering and annotating new data is a time consuming and costly task. The two lines are 1- Data based solutions (Chapters 4, and 5), and 2- Model-based solutions (chapter 6). We introduced performing data synthesis and/or data augmentation to enrich the training data. Additionally, we proposed integrating it with another data augmentation approach based on mixing more than one input sample at a time.

Moreover, we explored using curriculum learning to improve the domain transfer and fine tuning pipeline by weighting training samples differently based on some difficulty criteria. In the latter, we explored training the abstractive summarization task in a multitask setting. We proposed a multitask model to train the abstractive summarization task with different (generative and predictive) tasks. We carried out a comprehensive search across the proposed tasks to find the most likely tasks to improve abstractive summarization when performed in low resource settings, even with different base models and/or data from different domains. In this chapter, however, we propose to combine the two approaches into one single framework, which can benefit from the best of the two worlds. The proposed framework would include training with multitask learning using (Language modeling, Paraphrase detection, and Concept detection) as auxiliary tasks[1], and it would also include the augmentation/synthesis steps that proved to be more effective according to chapter 5 (i.e., Synthesis with paraphrasing and Curriculum learning). Our experiments in this chapter aims to answer the following research questions:

[1]Extractive summarization is not reported as AY data doesn't have extractive annotation, and extractive summarization wasn't a helpful task in general (refer to chapter 6)

- **Question 1 (Q1)** : Would combining data synthesis with multitask learning outperform using each method in isolation ?
- **Question 2 (Q2)** : Can combining curriculum learning with multitask learning outperform using each method in isolation, even if the difficulty metric is not suitable for all tasks?
- **Question 3 (Q3)** : Finally, would combining all techniques (i.e., data synthesis, curriculum learning, and multitask learning) from the two approaches (data-based and model-based) lead to a better-performing model than using each approach in isolation?

To answer these questions, we perform a set of experiments using the combination of the two directions and compare the outcome to prior reported results. In the following sections, we provide the details of the models, and then we discuss the experimental design.

7.2 Summarization Models

In our work we perform experiments using the abstractive summarization model that achieved best scores in previous chapters (i.e. BART [41]) from chapters 5 and 6. In this chapter, we integrate multiple combinations of tasks with one or more of the augmentation techniques (curriculum learning and data synthesis) and fine tune the BART model similar to chapter 6. We fine tune the BART model using multiple tasks in parallel. The data for each task is either augmented with synthesized data, fed to the model through a curriculum, or both.

7.3 Datasets

In this chapter, we perform experiments using two different low resource summarization datasets (i.e., Amazon/Yelp (AY) and CourseMirror(CM)). For CM, we use the (train/ dev/ test) configuration, and for AY, we use the original (train/ dev/ test) splits.

7.4 Experimental Setup

This section describes the experiments we plan to do using the integrated model. We combine experiments done in chapters 5 and 6. We ultimately performed two sets of experiments (set 1 and set 2) spanning the integration of multiple techniques.

7.4.1 Set 1: Multitask Learning with Single Augmentation Technique (answering questions Q1, and Q2)

The first set of experiments focuses on integrating multiple augmentation techniques in isolation with multitask learning. We first perform experiments of data synthesis with multitask learning. We similarly then perform experiments of curriculum learning with multitask learning. To synthesis data, we used synthesis with paraphrasing approach (refer to chapter 5). We varied the number of synthesized samples to (5, 10) synthetic samples for each original sample within the experiments. We used GPT-2 fine tuned language model for paraphrasing trained by Krishna et al. in [39].[2]

Regarding curriculum learning, similar to chapter 5 we used two difficulty metrics to construct the curriculum (i.e., ROUGE and specificity). Unlike single-task learning, we need to sort data from multiple tasks using the curriculum. However, the two proposed difficulty metrics are suited for only abstractive summarization and paraphrase detection tasks (as both tasks are the only tasks that use input document and human summary). Thus, for other tasks, we presented the samples in random order.

7.4.2 Set 2: Multitask Learning with Multiple Augmentation Techniques (answering question Q3)

The second set of experiments focuses on integrating both augmentation techniques (i.e., data synthesis and curriculum learning) simultaneously with multitask learning. We construct a curriculum for both synthetic data and original data, and we then perform two steps of fine tuning. We fine tune using synthetic data ordered via curriculum for the first

[2]https://drive.google.com/drive/folders/1RmiXX8u1ojj4jorj_QgxOWWkryDIdie-?usp=sharing

step. We then do another fine tuning step using only the original data, similarly ordered via curriculum.

7.4.3 Model Training

In all the experiments, we use the DistilBART model trained with CNN/DM data, consisting of 6 layers on both encoder and decoder sides and a hidden size of 4096. We fine tune the model using 6 Nvidia Quadro RTX 5000 GPUs. We use a batch size of 1 sample per GPU. We use a maximum input length of 1024 tokens with right-side padding for shorter inputs. We generate an output of a maximum length of 128 tokens on the decoder side. The generation is done using beam search with a beam size of 10 beams. The first fine tuning step is done for one epoch using multiple tasks followed by ten epochs with only abstractive summarization task. The epoch with the highest ROUGE score on the validation set is used later for testing.

7.5 Results

In this section we report results obtained for different experiments. We report results for the two sets of experiments in tables 37, 38, 39, and 40 for set 1 of experiments and tables 41 and 42 for set 2.

7.5.1 *Set 1* - Results (answering questions Q1, and Q2)

First, we would like to observe the effect of combining data synthesis with multitask learning. According to tables 37, and 38 we can see that integrating synthesized data in small numbers (5 samples) can introduce improvements for both AY and CM data, especially for CM data compared to training with only original data. However, increasing the number of synthetic data diminishes the improvements (10 samples VS. 5 samples). Additionally, in our experiments on multitask learning, we observed consistent improvements through training with the paraphrase detection task. However, when we combined data synthesis with

	Original			Synthetic 5			Synthetic 10		
Tasks	R1	R2	RL	R1	R2	RL	R1	R2	RL
A L	48.54	22.34	34.36	50.54	24.87	35.82	44.53	19.09	31.43
A P	49.87	22.59	35.6	43.6	18.45	30.53	43.99	18.65	30.44
A C	47.81	20.32	32.96	51.3	24.43	36.32	43.1	17.47	28.64
A L P	47.11	23.04	33.58	45.75	21.76	32.4	44.38	19.05	29.47
A C P	47.76	21.5	33.43	43.05	16.47	28.33	47.02	19.62	31.98
A L C	48.35	21.44	33.43	50.27	23.35	35.73	43.93	18.25	29.87
A L P C	47.94	21.08	32.94	43.2	18.43	29.23	45.14	18.92	28.93

Table 37: ROUGE results of BART with both multitask only and multitask with data synthesis on CourseMirror data (highlighted means better than original)

	Original			Synthetic 5			Synthetic 10		
Tasks	R1	R2	RL	R1	R2	RL	R1	R2	RL
A L	37.85	10.66	24.03	38.01	10.99	24.16	32.6	7.83	20.83
A P	37.86	10.39	23.95	33.83	8.82	21.18	32.48	7.75	20.84
A C	38.04	10.81	24.15	38.59	10.69	24.2	33.53	8.5	21.65
A L P	36.96	9.45	22.96	32.67	7.95	21.07	32.41	7.77	20.56
A C P	37.52	10.42	23.55	34.2	8.26	21.67	34.16	8.31	21.2
A L C	37.32	10.39	23.57	37.98	10.57	23.74	32.39	7.83	20.97
A L P C	36.46	10.32	24.08	33.07	7.91	21.11	31.29	7.6	20.18

Table 38: ROUGE results of BART with both multitask only and multitask with data synthesis on Amazon/Yelp data (highlighted means better than original)

	Original			Curr (R)			Curr (S)		
Tasks	R1	R2	RL	R1	R2	RL	R1	R2	RL
A L	48.54	22.34	34.36	44.51	19.6	31.45	44.1	19.5	31.36
A P	49.87	22.59	35.6	45.47	20.01	32.23	45.86	20.65	32.56
A C	47.81	20.32	32.96	44.61	19.83	31.4	48.31	19.94	32.33
A L P	47.11	23.04	33.58	44.46	20.1	32.9	43.36	19.04	30.72
A C P	47.76	21.5	33.43	46.4	20.87	32.3	43.34	20.43	32
A L C	48.35	21.44	33.43	45.93	16.93	30.1	45	17.8	30
A L P C	47.94	21.08	32.94	46.1	18.26	29.9	45.2	18.73	31.01

Table 39: ROUGE results of BART with both multitask only and multitask with curriculum learning on CourseMirror data (highlighted means better than original)

	Original			Curr (R)			Curr (S)		
Tasks	R1	R2	RL	R1	R2	RL	R1	R2	RL
A L	37.85	10.66	24.03	36.9	10.23	24.4	35.59	8.9	23
A P	37.86	10.39	23.95	36.58	10.65	23.84	37.25	9.66	23.02
A C	38.04	10.81	24.15	36.51	10.84	23.83	37.84	10.26	23.71
A L P	36.96	9.45	22.96	36.78	10.3	24.32	35.66	8.98	22.8
A C P	37.52	10.42	23.55	36.13	10.5	23.86	37.04	10.1	23.26
A L C	37.32	10.39	23.57	36.12	9.97	24	35.63	9.35	23.3
A L P C	36.46	10.32	24.08	36.63	10.2	24.45	35.25	9.31	22.91

Table 40: ROUGE results of BART with both multitask only and multitask with curriculum learning on Amazon/Yelp data (highlighted means better than original)

multitask learning, we found improvements with two different tasks (language modeling and concept detection), and the paraphrase detection task failed to introduce any improvements. In the next section, we perform some analysis to understand the discrepancy in helpful tasks between the two cases (i.e., training with only multitask learning and combining multitask learning with data synthesis).

We now observe the effect of integrating curriculum learning with multitask learning. Tables 39, and 40 show the results obtained through integrating curriculum learning with multitask learning for both CM and AY datasets respectively. According to both tables, we can see that some improvements can be realized on AY data using the ROUGE curriculum metric. The improvements are neither for all ROUGE scores nor consistent across tasks. Additionally, no improvements can be seen by using specificity metric on AY data. Moreover, there are almost no improvements when combining curriculum with multitask learning on CM data either by using ROUGE or specificity metrics. Thus, we can see that combining curriculum learning with multitask learning is not very helpful. This agrees with our earlier hypothesis that the proposed curriculum metrics are unsuitable for all tasks.

7.5.2 *Set 2* - Results (answering question Q3)

Now we observe the effect of combining all the techniques we have explored so far. We combine multitask learning with two data manipulation techniques (i.e., data synthesis and curriculum learning). First, table 41 shows the results obtained through combining curriculum learning using ROUGE and specificity metrics with data synthesis and multitask learning for CM dataset. Second, table 42 show the results obtained through combining curriculum learning using ROUGE and specificity metrics with data synthesis and multitask learning for AY dataset.

According to table 41 we can observe similar behavior for both constructing the curriculum using ROUGE and specificity metrics on CM dataset. We can see that integrating curriculum to sort synthetic data (5 samples) introduced improvements for the task combinations involving paraphrase detection tasks (i.e., AP, ALP, ACP, ALPC). The improvements observed for tasks involving paraphrase detection strengthen our hypothesis that proposed

	Original			Synthetic 5								
				No Curriculum			+ Curriculum (R)			+ Curriculum (s)		
Tasks	R1	R2	RL	R1	R2	RL	R1	R2	RL	R1	R2	RL
A L	48.5	22.3	34.3	**50.5**	**24.8**	**35.8**	47.1	21.1	33.2	47.7	21.6	34.5
A P	**49.8**	22.5	**35.6**	43.6	18.4	30.5	48.4	**22.6**	35.1	46.8	21.6	34
A C	47.8	20.3	32.9	**51.3**	**24.4**	**36.3**	46.8	22.5	32.8	48.3	22.4	34.7
A L P	**47.1**	**23**	33.5	45.7	21.7	32.4	46.5	19.6	31.5	46.3	21.1	**33.8**
A C P	47.7	21.5	33.4	43	16.4	28.3	**48.6**	**22.6**	**34.2**	48	22.3	33.7
A L C	48.3	21.4	33.4	**50.2**	**23.3**	**35.7**	45.7	19	31.5	44.2	19.5	30.4
A L P C	**47.9**	**21**	**32.9**	43.2	18.4	29.2	46	19.4	31.8	44.5	20.2	32.1
	Original			Synthetic 10								
				No Curriculum			+ Curriculum (R)			+ Curriculum (s)		
Tasks	R1	R2	RL	R1	R2	RL	R1	R2	RL	R1	R2	RL
A L	**48.5**	**22.3**	**34.3**	44.5	19.1	31.4	47.1	22.1	32.3	46.2	20.1	32.6
A P	**49.8**	22.5	**35.6**	44	18.6	30.4	48.6	21.5	33.6	47.2	**23.3**	34.4
A C	**47.8**	20.3	32.9	43.1	17.4	28.6	46.8	20.9	**33.8**	46.5	**22**	33.7
A L P	**47.1**	**23**	33.5	44.3	19	29.4	46.2	21	**34.2**	44.8	20.3	31.4
A C P	47.7	21.5	**33.4**	47	19.6	31.9	**48**	**21.8**	**33.4**	46.7	21.6	**33.4**
A L C	**48.3**	**21.4**	**33.4**	43.9	18.2	29.8	44.6	20.3	31.2	47	21.3	32.3
A L P C	**47.9**	21	**32.9**	45.1	18.9	28.9	44.3	19.9	31	46.3	**21.5**	32.7

Table 41: Results of BART with multitask, Synthesis, and curriculum learning on CM data (highlighted means better than no curriculum. **Bold** means best ROUGE scores across each combination of tasks)

	Original			Synthetic 5								
				No Curriculum			+ Curriculum (R)			+ Curriculum (S)		
Tasks	R1	R2	RL	R1	R2	RL	R1	R2	RL	R1	R2	RL
A L	37.8	10.6	24	**38**	**10.9**	**24.1**	35.8	8.5	22.5	34.6	7.6	21.1
A P	**37.8**	**10.3**	**23.9**	33.8	8.8	21.1	36.7	9.1	23	36	8.5	23
A C	38	**10.8**	24.1	**38.5**	10.6	**24.2**	37	9	23.3	36	9	23
A L P	**36.9**	**9.4**	**22.9**	32.6	7.9	21	33.9	7.7	21.9	32	7	20.6
A C P	**37.5**	**10.4**	**23.5**	34.2	8.2	21.6	36.9	8.9	23.2	35.4	8.6	22.5
A L C	37.3	10.3	23.5	**37.9**	**10.5**	**23.7**	34.6	8.4	22.1	33.2	7.6	21.2
A L P C	**36.4**	**10.3**	**24**	33	7.9	21.1	33.1	7.1	20.8	32.1	6.8	20.7
	Original			Synthetic 10								
				No Curriculum			+ Curriculum (R)			+ Curriculum (S)		
Tasks	R1	R2	RL	R1	R2	RL	R1	R2	RL	R1	R2	RL
A L	**37.8**	**10.6**	**24**	32.6	7.8	20.8	34.9	8.3	22.2	33.5	7.7	21.1
A P	**37.8**	**10.3**	**23.9**	32.4	7.7	20.8	36.7	9.1	23	35.7	8.4	22.5
A C	**38**	**10.8**	**24.1**	33.5	8.5	21.6	37.3	9.5	22.9	33.5	7.7	21.1
A L P	**36.9**	**9.4**	**22.9**	32.4	7.7	20.5	33.3	7.2	21.2	33.3	7.1	21.1
A C P	**37.5**	**10.4**	**23.5**	34.1	8.3	21.2	37.3	9.7	23.2	35.7	8.7	22.8
A L C	**37.3**	**10.3**	**23.5**	32.3	7.8	20.9	34.1	8.1	21.6	34.2	7.2	21.4
A L P C	**36.4**	**10.3**	**24**	31.2	7.5	20.1	33.3	7.8	21.1	32	6.8	20.7

Table 42: Results of BART with multitask, Synthesis, and curriculum learning on AY data (highlighted means better than no curriculum. **Bold** means best ROUGE scores across each combination of tasks)

metrics used in the curriculum are more suited to abstractive summarization and paraphrase detection tasks. Additionally, we can see that using a curriculum on data synthesis with more samples (10 samples) introduced improvements almost across all tasks and all ROUGE scores. This proves that curriculum learning is beneficial, as it helps use the more noisy data more efficiently, thus improving the performance across all tasks. We didn't observe any tangible difference between the two difficulty metrics when used on the CM dataset.

Now we move to results obtained on the AY dataset. Table 42 show results for both constructing the curriculum using ROUGE and specificity metrics on AY dataset. Unlike results obtained on the CM dataset, we can see that constructing a curriculum using ROUGE introduces more improvements for synthesizing data with 5 and 10 samples for the AY dataset. Moreover, similar to observations on the CM dataset, curriculum learning introduces improvements for combinations that include paraphrase detection on AY. Additionally, it similarly introduces improvements for all combinations of tasks on AY when applied to the more noisy synthetic data (10 samples).

Finally, we observed improvements by adding curriculum learning to the scores obtained through using data synthesis with multitask learning. Specially, on task combinations involving paraphrase detection which unlike training with multitask learning only, didn't introduce improvements. However, even with the improvements achieved by integrating the curriculum, the performance is still worse compared to training with multitask learning only except for ACP on CM data. Thus, combining the two manipulation techniques with multitask learning in most cases is not very helpful compared to using only data synthesis with the multitask learning.

7.6 Analysis

7.6.1 Named Entities

A recent line of work by Liu and Chen [48] introduced controlling summary generation via named entities. They showed that summarization quality and factual consistency could

benefit from the inclusion of named entities. Additionally, Liu and Liu [46] performed analysis on generated summaries using named entities. With that said, influenced by the prior work, we decided to perform further analysis to find if the number of named entities that appear in both input and reference summary and appear in generated summary relates to the ROUGE scores. We want to investigate if the named entities produced by the model are somehow related to auxiliary tasks used in multitask learning. Additionally, we investigate how the model behavior changes when using original or synthetic data. Moreover, we want to investigate if models focus more on named entities or regular words and how that affects ROUGE scores.

7.6.1.1 Named entity extraction

		CM				AY			
Data	Tasks	R1	R2	RL	F1	R1	R2	RL	F1
Original	AP	49.9	22.6	35.6	61.7	37.8	10.4	24	56.1
	AL	48.5	22.3	34.4	59.3	37.8	10.6	24	58.9
	AC	47.8	20.3	33	57.4	38	10.8	24.2	57.2
Syn. 5	AP	43.6	18.5	30.5	47.2	33.8	8.8	21.1	58.1
	AL	50.5	24.9	35.8	58.3	38	11	24.1	59.2
	AC	51.3	24.4	36.3	55.7	38.6	10.7	24.2	58
Syn. 10	AP	44	18.7	30.4	57.9	32.4	7.7	20.8	57.7
	AL	44.5	19.1	31.4	59.3	32.6	7.8	20.8	66.1
	AC	43.1	17.47	28.6	56.3	33.5	8.5	21.6	61.7
Correlation between F1 and ROUGE									
		CM				AY			
		F1-R1	F1-R2	F1-RL		F1-R1	F1-R2	F1-RL	
Original		0.99(0.08)	0.86(0.3)	0.99(0.06)		-0.11(0.92)	0.63(0.55)	-0.11(0.92)	
Syn. 5		0.95(0.2)	0.98(0.09)	0.95(0.19)		0.35(0.76)	0.49(0.67)	0.51(0.65)	
Syn. 10		0.99(0.07)	0.97(0.15)	0.99(0.07)		0.14(0.98)	-0.02(0.98)	-0.02(0.98)	

Table 43: NER F1 Pearson correlation with ROUGE for CM and AY. P-value is shown between parentheses

We extracted named entities from the input documents using spaCy[3] open-source python package. We then marked named entities that appear in reference summaries as 1 and those

[3]https://spacy.io/

that don't appear as 0. We then measured the F1 for extracted named entities in generated summaries for 3 task combinations (AP, AL, AC). We chose these three tasks as we don't have extractive annotation for the AY dataset. We also used one auxiliary task instead of multiple ones to analyze how each task affects named entities on its own. Table 43 shows the F1 achieved for different task combinations using original and synthetic data on both CM and AY datasets.

Additionally, table 43 shows the correlation[4] between F1 on all tasks and ROUGE scores (i.e., R1, R2, and RL) for original and synthetic data on both CM and AY datasets. We can see that there is a very high correlation between all ROUGE scores and F1 for CM, both original and synthetic. The high correlation means that on CM, the more named entities the model produces, the higher the ROUGE score the model gets. On the other hand, we see a low or negative correlation between ROUGE scores and F1 on AY. The negative correlation, in turn, indicates that there is almost no relation between the ROUGE performance and the number of produced named entities on AY.

7.6.1.2 Most frequent named entities

We perform additional experiments to analyze the discrepancy between AY and CM datasets regarding the correlation between named entity and ROUGE score and further analyze how important named entities can be for summaries. We extracted the most frequent named entities for both CM and AY. The most frequent named entities are the ones that appear most across all summaries. Table 44 shows the most frequent named entities that appear in original and synthetic data for both CM and AY datasets. According to table 44 we can see that for CM, a lot of the most frequent named entities either produced through different models or in reference summaries are words referring to the number of students (e.g., 60%, over half, and approximately 20%). We found that the focus on these named entities decreases as we synthesize more data, which is expected as we are introducing new paraphrases replacing these named entities. On the other hand, for AY, the most frequent

[4]Unfortunately, most of the calculated correlation values are not significant due to the small sample size (i.e., only three samples, as we calculate the correlation between F1 and R1, R2, and RL using three F1 values for (AP, AL, AC) and three R1, R2, and RL values for (AP, AL, AC)).

named entities vary significantly with no apparent effect of data synthesis on the retrieved named entities. This, in turn, can justify the discrepancy we observed in the correlation between AY and CM, as models on CM tends to produce named entities such as *60%, over half, and approximately 20%, etc.* to match the reference summary. In order to verify our argument, we do a similar analysis after performing a filtering step to remove some of the most frequent named entities from CM data.

7.6.1.3 Filtering named entities

To analyze the effect of the presence of the most frequent named entities such as *60%, over half, and approximately 20%, etc.*, we removed these named entities from both produced summaries and reference summaries and re-calculated ROUGE scores and NER F1. Table 45 shows F1, ROUGE scores, and correlation[5] for CM data before and after filtering the most frequent words. Additionally, table 46 shows the correlation between F1 and ROUGE score for each task combination on both CM and AY datasets. According to table 45 we can see that removing the most frequent words led to a huge drop in the correlation between F1 and ROUGE scores for results obtained from original data. On the other hand, we can see a less impact or a very slight impact on correlation for synthetic data. This shows that synthesizing new data helped the model avoid focusing on a small subset of named entities such as (60%, over half, and approximately 20%, etc.). Furthermore, we can see from table 46 that summaries generated through AP trained models for the CM dataset tend to focus on named entities from reference summaries, thus achieving a high correlation between ROUGE and F1. Even when filtering the most frequent named entities, AP's correlation is slightly affected. This means that summaries produced from training with paraphrasing as an auxiliary task contain a lot of named entities that appear in reference summaries. However, the named entities are not focused just on the most frequent ones. As for the AY dataset, while the correlation is negative, we can still see that AP has the highest correlation compared to the other tasks.

[5]Similarly, most of the calculated correlation values are not significant due to the small sample size (i.e., only three samples, as we calculate the correlation between F1 and R1, R2, and RL using three F1 values for (AP, AL, AC) and three R1, R2, and RL values for (AP, AL, AC)).

CM					
Original	Ref	Almost half	milestone 3	a little under half	a great year
	A P	60%	approximately 20%	summer	ssr
	A L	approximately 20%	ssr	m3	one
	A C	over half	approximately 20%	one	ssr
Syn. 5	Ref	almost half	a little under half	a great year	milestone 3
	A P	more than a third	3-year-old	4th	one
	A L	approximately 20%	100 years	one	ssr
	A C	approximately 20%	over half	another 20%	one
Syn. 10	Ref	almost half	a little under half	mile stone 3	a great year
	A P	ssr	two	m3	2
	A L	ssr	third	one	first
	A C	ssr	two	4	mile stone 3
AY					
Original	Ref	one	bennett	many years	mexican
	A P	only one	3	today	70-300mm
	A L	two	six-years-old	$10 00	the morning
	A C	1963	100	toronto	first
Syn. 5	Ref	one	bennett	many years	mexican
	A P	the day of the day	3-year-old	70-300mm	24/7/
	A L	mexican	one	years	24 hours
	A C	mexican	45 minutes	24 hours	$10 00
Syn. 10	Ref	one	bennett	many years	mexican
	A P	a couple of days	a minute	hour	the weekend
	A L	mexican	one	30-day	ages 4 to 100
	A C	a couple of years	$10 00	100	mexican

Table 44: Most frequent named entities in CM and AY

7.6.1.4 Named entities distribution

Another aspect we analyzed is the number of named entities that appear in train, test, and generated summaries for different tasks on both CM and AY datasets. Table 47 shows the percentage of ratio of named entities to all words that appear in train (input/output), test(input/output), and generated summaries for CM, CM after filtering named entities, and AY datasets. From table 47 we can see that for CM (original and filtered), the number of generated named entities increase for AL when moving from original to synthetic-5 data, this correlates with increase in ROUGE scores as well (refer to tables 43, and 45). We can

		CM				CM - Filtering			
Data	Tasks	R1	R2	RL	F1	R1	R2	RL	F1
Original	AP	49.9	22.6	35.6	61.7	49.71	21.6	35.1	61.1
	AL	48.5	22.3	34.4	59.3	48.1	21.6	33.8	54
	AC	47.8	20.3	33	57.4	45.3	15.7	29	59.3
Syn. 5	AP	43.6	18.5	30.5	47.2	43.2	18	30.1	47.2
	AL	50.5	24.9	35.8	58.3	50.2	24.4	35.3	54
	AC	51.3	24.4	36.3	55.7	51.3	24	36	55.7
Syn. 10	AP	44	18.7	30.4	57.9	43.7	18.1	30.3	57.9
	AL	44.5	19.1	31.4	59.3	44.4	18.9	31.4	59.3
	AC	43.1	17.4	28.6	56.3	48.7	21.7	32.1	59.3
Correlation (p-value)									
		CM				CM - Filtering			
		F1-R1	F1-R2	F1-RL		F1-R1	F1-R2	F1-RL	
Original		0.99(0.08)	0.86(0.3)	0.99(0.06)		0.15(0.9)	-0.21(0.8)	-0.01(0.9)	
Syn. 5		0.95(0.2)	0.98(0.09)	0.95(0.19)		0.99(0.04)*	0.97(0.15)	0.99(0.04)*	
Syn. 10		0.99(0.07)	0.97(0.15)	0.99(0.07)		0.61(0.57)	0.67(0.53)	0.92(0.25)	

Table 45: NER F1 Pearson correlation with ROUGE for CM after filtering, CM values without filtering from table 43. P-value is shown between parentheses (* means statistically significant)

		No Filtering			Filtering		
Data	Tasks	F1-R1	F1-R2	F1-RL	F1-R1	F1-R2	F1-RL
CM	A P	0.74(0.47)	0.73(0.47)	0.68(0.51)	0.74(0.46)	0.72(0.48)	0.71(0.49)
	A L	-0.74(0.47)	-0.82(0.38)	-0.73(0.48)	-0.93(0.23)	-0.86(0.33)	-0.92(0.24)
	A C	-0.98(0.12)	- 0.93(0.24)	-0.98(0.12)	-0.82(0.38)	-0.7(0.5)	-0.9(0.29)
AY	A P	-0.89(0.29)	-0.85(0.34)	-0.95(0.17)	NA		
	A L	-0.99(0.03)*	-0.99(0.07)	-0.99(0.05)	NA		
	A C	-0.96(0.17)	-0.98(0.13)	-0.99(0.05)	NA		

Table 46: NER F1 Pearson correlation with ROUGE for each task using values from all data variants (original, synthetic 5, and synthetic 10) (Recall that we didn't perform filtering for AY as the named entities varied significantly, unlike CM) (* means statistically significant)

Data	Train		Test		Generated		
	Input	Reference	Input	Reference	A P	A L	A C
CM	2.354	1.978	2.332	1.550	1.818	1.118	1.893
CM Syn 5		1.605			0.944	1.525	1.792
CM Syn 10		1.565			0.531	0.480	1.047
CM (Filtered)	2.354	1.839	2.332	1.408	1.621	1.011	1.683
CM Syn 5 (Filtered)		1.582			0.896	1.450	1.321
CM Syn 10 (Filtered)		1.549			0.531	0.480	1.078
AY	1.869	0.615	1.843	0.657	0.814	0.632	0.548
AY Syn 5		0.562			0.555	0.749	0.793
AY Syn 10		0.623			0.342	0.281	0.494

Table 47: Percentage of named entities in train, test and generated summaries for CM and AY datasets

also see that the decrease in number of named entities correspond to decrease in ROUGE score as seen in AP when moving from original to synthetic-5 to synthetic-10, and also for AL and AC when moving from synthetic-5 to synthetic-10. Similarly, we can see that on AY the number of generated named entities increase for both AL and AC when moving from original to synthetic-5 data. This correlates with the increase in ROUGE scores. Similar to CM, we can see that the number of generated named entities decreased for AP when moving from original to synthetic-5 to synthetic-10 which also correlates with the decrease in ROUGE scores we saw earlier in table 43.

7.6.1.5 Conclusion of analysis

In conclusion, we found a high correlation between copying named entities from reference summaries and ROUGE score for the CM dataset. However, the correlation is not clearly present in the AY dataset, except for synthetic data. We extracted and examined the most frequent named entities from CM data. We then filtered these named entities to see if the correlation would change. We found that we can still get a very high correlation with synthetic data. Moreover, we found that the AP task has a good correlation between ROUGE and F1 before and after filtering. Thus, we can conclude that using synthetic data can help produce summaries with more diverse named entities. To further analyze the impact of named entities in summaries, we counted the ratio of named entities to regular words for different datasets. We found that for both CM (original/Filtered) and AY datasets, there is a correlation between increasing the ratio of named entities and getting higher ROUGE scores. We also found that AL and AC combinations tend to generate more named entities when trained with synthetic data, thus leading to a higher ROUGE score. In the end, we can see that named entities can give us a good indication of how different tasks would perform across different datasets and of why there is a discrepancy between tasks that performed well on original data (AP) and ones that performed well on synthetic data (AL and AC).

7.7 Conclusion

In this chapter, we integrate the two directions (data manipulation from chapter 5 and multitask learning from chapter 6) into one framework to see if that would improve the use of each direction in isolation. We integrated multitask learning with the two best performing augmentation techniques (Synthesis with paraphrasing and Curriculum learning) either one at a time or both simultaneously. We observed that integrating each technique in isolation with multitask learning led to improvements. However, we found a discrepancy between the combination of tasks that introduced improvements in training with original data and training with synthetic data. We then performed further analysis using named entities to find the cause of this discrepancy and get an insight into how different tasks would perform on different datasets. We ultimately found that tasks that produce more named entities and produce a distribution of named entities close to the distribution in reference summaries can lead to higher scored summaries. Finally, we found that combining curriculum learning with multitask learning didn't help the model, and we argue that this is due to the lack of a general difficulty metric that suits all tasks. We also found that combining both augmentation approaches (synthesis and curriculum) simultaneously with multitask learning didn't introduce improvements over training with only synthetic or original data.

8.0 Conclusions

In this work, we focus on automatic text summarization for low resource domains which is an under explored area. We also focus our work on less studied domains of student reflections and reviews. *Unlike, prior work which mainly focused on extractive summarization solutions, we on the other hand focused on abstractive summarization solutions which are more challenging.* In the course of this work, we explored two different directions to improve the performance of neural abstractive summarization models for low resource domains. **The first direction** focused on improving model performance through manipulating the data. First, we explored improving the performance of neural abstractive summarization models using three approaches: domain transfer, data synthesis, and the combination of both. To tackle *data synthesis that is rarely explored in the domain of text summarization, we proposed a new template based synthesis model to synthesize new summaries. We then showed the utility of the under explored data synthesis in improving model performance.* Despite improving model performance, the proposed synthesis model depends heavily on templates extracted from in-domain training data. This can limit the model from generalizing to domains with diverse forms of summaries such as news. Motivated by the improvement data synthesis introduced and the limitations of template based data synthesis, we decided to further investigate other potentially more generalizable data synthesis approaches and additional data augmentation/manipulation techniques. We then proposed data synthesis using paraphrasing and data augmentation using sample mixing (MixGen). Moreover, we introduced training the model with the aid of curriculum learning to help the model learn gradually which is also an under explored technique in the domain of text summarization. *We found that the proposed paraphrasing data synthesis approach is able to improve the performance and to generalize to more complex data such as news. We found that MixGen introduced slight improvements when mixing is done few number of times and fails otherwise. With various parameters to tune we argue that MixGen needs further analysis and that it can be a good candidate for future work. Finally, we showed that the two proposed difficulty metrics (i.e., ROUGE and specificity) are able to improve the model training. However, using sum-*

marization related ROUGE metric for curriculum learning most leads to more improvements compared to generic metric such as specificity.

In our second direction we focused on improving the model itself through multitask learning. In this direction, we explored the utility of training a multitask model for abstractive summarization using low resources summarization data. Prior work showed that certain auxiliary tasks such as extractive summarization and language modeling can be helpful with large data. We additionally proposed two new tasks (i.e. paraphrase detection and concept detection). We performed several experiments to find if the proposed tasks would be helpful and if the previously studied tasks would transfer well to low resource domains. *surprisingly we found that the very relevant extractive summarization task didn't offer consistent improvement across all experiments. We also found that the proposed paraphrasing and concept detection tasks, which had not been previously examined as auxiliary tasks for abstractive summarization, can be very helpful given low resource data.*

In the last step, we merged the two directions into one framework. We showed that combining multitask learning with data synthesis would introduce additional improvements. However, the improvements are more prevalent in combinations of tasks that include language modeling. This can be due to the human summaries' variability introduced via paraphrasing data synthesis, which can help the language modeling task that depends only on summary rather than reflections for training, with a richer set of summaries. Moreover, we found that to integrate curriculum learning with multitask learning, a more general difficulty metric that suits all tasks is needed. This can be another interesting direction for future work. Finally, we showed that combining multiple augmentation techniques with multitask learning would harm the model performance, which is another side effect of the absence of a generic difficulty metric for curriculum learning.

9.0 Future Directions

9.1 Evaluation

ROUGE is the most common evaluation metric used in the summarization community. Unfortunately, ROUGE can't capture many aspects of produced summaries that are important to judge the summarization quality, such as coherency, factual consistency, diversity, etc. In the course of our work, we focused on using ROUGE as the fundamental evaluation metric. However, we tried to enrich the evaluation with human evaluation in order to overcome the shortcoming of ROUGE. Recently there has been a growing interest in conducting evaluation using different evaluation metrics (e.g., question answering, factual consistency, semantic distance using BART, etc.). Thus, a future direction of this work can integrate multiple evaluation metrics and analyze how different evaluation metrics correlate with different auxiliary tasks, augmentation techniques, and human decisions.

9.2 Results' Significance

During the course of this work, we performed a large number of experiments. Many of these experiments produced positive results. However, the differences in numbers in some cases are minuscule. We performed statistical significance tests to verify if the differences are significant or not (refer to section 4.5.1 and section 6.6.1). Unfortunately, we found that most of the numbers are not significant due to the small sample size (four in our case due to leave one course out setting). In the future, we plan to replicate experiments that produced small differences, using more samples by performing experiments in a cross-validation setting. Additionally, we plan to explore how differences in ROUGE score can affect other metrics such as factual consistency, question answering capability, etc. We plan to analyze how the observed conclusions would transfer to other different metrics and how strong or weak these conclusions transfer.

9.3 Data Synthesis

We showed the utility of under-explored data augmentation, especially data synthesis for abstractive summarization. In our experiments, we performed synthesis using two different techniques, template-based and paraphrasing-based. Moreover, we showed that paraphrasing-based synthesis is effective with domains such as reflections and reviews and more structured domains such as news. However, in our analysis, we found that the synthesis was harmful to the model when we used synthesis on CNN data with 20k training samples. This raises a couple of questions "1- What can be the breaking point ?", "2- Does the breaking point depend on the domain ?". In the future, we plan to investigate the amount of data that the synthesis might fail after. We also plan to investigate if this amount of data can be different for each domain (i.e., news, reviews, scientific papers, reflections) or it can be generic regardless of the domain.

9.4 Domains

In this work, we focused on understudied domains such as student reflections and reviews, which unlike news articles or scientific papers, fundamentally consist of independent pieces of text. To overcome this limitation, we analyzed how generalizable our findings would be on different domains such as news. In future work, we would like to expand our analysis to more structured and more difficult data such as scientific papers, emails, and medical notes. We think that simulating low resource settings for these different domains would help judge how generalizable the findings of this work are.

9.5 MixGen

In our experiments on sample mixing using MixGen, we found that increasing the number of times we perform mixing more than three times for each sample leads to performance

degradation. Moreover, we found that we could only get slight improvement compared to other augmentation techniques. Our experiments varied the number of times we performed mixing while fixing the remaining parameters (i.e., number of layers to propagate samples through, α that controls the β distribution). In future work, we think that tuning these parameters is worth exploring and that it might lead to a different conclusion regarding the sample mixing utility.

9.6 Curriculum Learning

Difficulty metric is a significant component of curriculum learning. In our work on curriculum learning, we focused on difficulty metrics pre-computed prior to model training, which can neglect the model's potential to be better at learning certain aspects. Rather than using a pre-computed difficulty metric, some research [98] used a smaller version of the model, trained it with a portion of the training data, and then used it to judge other training samples' difficulty. In the future, we plan to investigate calculating difficulty using the actual model and compare the performance with pre-computed metrics. Moreover, we would explore different difficulty metrics for constructing a curriculum and see if we can apply different ones for different tasks and if there is a general metric that can positively affect all tasks. Additionally, we plan to explore different types of datasets that contain more structure (news, papers, etc.) in the future, and we might see different findings than the ones we observed in this work. Finally, we followed prior work and fixed the number of buckets to split data into to be 10. In the future, we plan to change the number of buckets and observe its effect on model performance.

9.7 Extractive Summarization

In prior work targeting richer domains, extractive summarization proved to be a helpful auxiliary task. On the other hand, in our experiments, surprisingly, we didn't find improve-

ments by integrating extractive summarization as an auxiliary task in most cases. This might be due to the fact that in this work, we represented the auxiliary extractive summarization task as a simple non-auto-regressive classification task. This, in turn, introduced limitations to the task modeling. Human annotators also selected only five out of all input reflections as the extractive summary. We then used these reflections as positive samples while considering all other reflections as negative samples. We argue that this naive solution neglects a fundamental property of reflections/opinions, which is redundancy. For the CM dataset, multiple reflections are very similar, and training the model to accept a reflection and reject a similar one if not selected by humans as part of the summary can lead the model to perform poorly. Thus, we plan to perform a clustering step before training the extractive summarization task in the future. Clustering similar reflections might help train the model with more coherent decisions.

Appendix A Summarization Annotation

In the course of this work we used CourseMirro data [52, 54, 51]. The data was annotated as part of work done by Luo [51]. According to Luo the process of annotating student reflections with reference summaries is done by following a set of instructions. Table 48 shows the instructions provided to annotators to annotate reflections with (abstractive, extractive, and phrase) reference summaries.

In creating each summary, you should keep in mind the following scenario for its use. Imagine you are a TA for this course, what do you want to present to the instructor after reading the students' responses for each of the following two prompts? Prompt1: "Describe what you found most interesting in today's class?" Prompt2: "Describe what you found most confusing in today's class?"
Task1: Phrase Summarization. Create a summary using 5 phrases together with how many students semantically mentioned each phrase. You can use your own phrases.
Task2: Abstract Summarization. Given the students' responses, create a short summary using your own words (40 words) of it. The summary needs to be a coherent paragraph and should include the major points. The summary should only contain information about reflections, and avoid adding irrelevant sentences or suggestions such as " Make sure to bring this up in next class", or "Consider this for future lectures" , etc..
Task3: Extractive summary. Select five most representative sentences in order as the summary. (Use the sentence index number.)

Table 48: Instruction provided to annotators during summarization annotation process.

Appendix B Additional Multitask Learning Scores

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

Table 49: 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. *Italics* indicates improving scores across different datasets. Underlining indicates improving scores across different datasets and different models.

Tasks	R1	R2	RL	AVG	Δ	R1	R2	RL	AVG	Δ
	CS0445					**ENGR**				
A	34.62	9.46	29.84	24.64	*	35.43	9.93	31.07	25.47	*
AE	30.01	8.21	22.92	20.38	-	32.04	8.11	27.10	22.41	-
AC	34.42	9.71	29.31	24.48	-	35.84	10.14	31.38	25.78	+
AP	***34.56***	***9.81***	***30.11***	***24.82***	+	***36.79***	***12.64***	***32.62***	***27.35***	+
ACP	34.70	9.47	30.2	27.79	+	36.16	11.46	31.74	26.45	+
AEC	27.43	7.54	24.63	19.86	-	29.41	7.63	26.15	21.06	-
ALL	28.34	8.31	26.72	21.12	-	30.11	8.45	28.98	22.51	-
	STAT 2015					**STAT 2016**				
A	36.87	12.03	32.34	27.08	*	37.41	12.33	33.02	27.58	*
AE	27.65	7.96	22.74	19.45	-	30.25	10.93	26.45	22.54	-
AC	34.49	10.40	30.12	25	-	37.09	12.77	32.42	27.42	-
AP	***36.78***	***12.64***	***32.62***	***27.34***	+	***38.86***	***13.41***	***33.84***	***28.71***	+
ACP	35.63	11.14	30.85	25.87	-	38.64	14.27	33.52	28.81	+
AEC	28.25	7.97	23.15	19.79	-	31.65	11.60	26.86	23.37	-
ALL	31.21	10.66	28.99	23.62	-	31.57	10.99	27.20	23.25	-

Table 50: ROUGE results of T5 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. *Italics* indicates improving scores across different datasets. Underlining indicates improving scores across different datasets and different models.

Appendix C Specificity

C.1 Annotation Chart

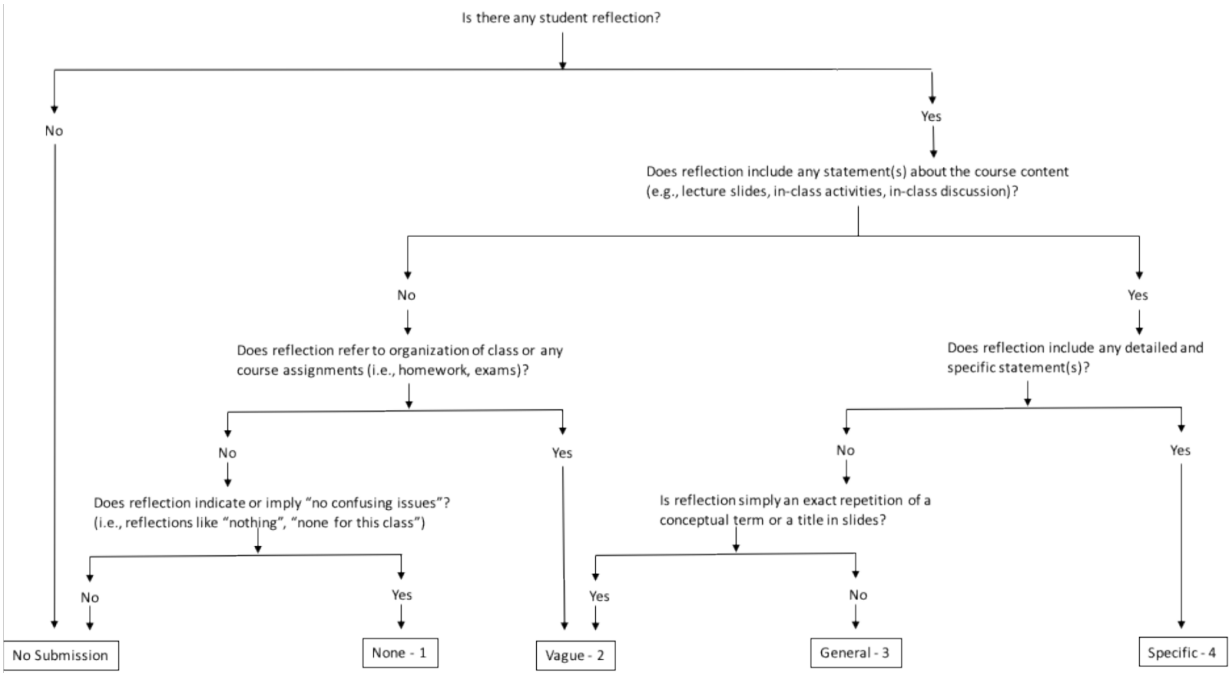


Figure 8: Flow chart of specificity annotation guidelines.

Points Of Interest	Score	Confusing Points	Score
Matlab vectors	2	Matrix operations	2
Test fan	2	Test fan	2
testets	2	testetest	2
Seeing the way the program can solve complex mathematical equations.	3	The most confusing part is finding what was the error in a fail command. It is a very sensitive process.	4

I found vector operations interesting.	3	Explanation of the MATLAB "matrix dimension" error message.	4
The people coming in to discuss the app and having us download it. Seems like a cool tool to use for our class. However, what is more interesting is the ability to get extra credit from using this app which I personally feel is amazing. Like who doesn't love extra credit? Extra credit is the greatest.	4	Everything was well described so I was not confused or in need of more detail.	1
Doing the math equations in matlab	3	I dont really understand the significance of the scripts	3
the coursemirror presentation	2	how to omit numbers from a vector or matrix that doesn't go in order	4
What I found interesting is how I could actually write a program and make it work on my computer.	4	Certain functions should be described throughout the questions instead of the end (example I didn't know what % did so I had to search for it).	4
The most interesting thing was the ability to compress the information presented to the user in the command window.	4	The difference between a matrix and a vector is confusing. Also, when to use the dot and when to not use it.	4
Using vectors in Matlab	2	Following the assignment guidelines.	3
getting to start using MATLAB	2	creating a script for the first time.	3

I found the calculations exercuse interesting and refreshing. It helped me refresh my concepts and also learnt new terms and concepts. I'm looking forward to learning more calculation concepts.	4	I was confused with script file part of the problem set 1. Also, unlike ENGR 132, This did not have unlimited attempts in quizes, so that might be slightly challenging and tough. Some MATLAB concepts were confusing.	4
Learning to code is very interesting. I have not coded much before but I am enjoying it.	3	I find using MATLAB help to be confusing. It does not seem to have a very intuitive search engine and most of the time I find it difficult to find what I'm looking for.	4
The most interesting thing about today's class was learning how to change the format on a script, making it more compact.	4	I was confused on how to write natural logs and log functions in MATLAB.	4
I found learning learning how to use a script in MATLAB as interesting, also, the format compact was very helpful.	4	The user interface if Matlab is confusing. I'm not really sure if what I'm doing is correct, and I don't know what I don't know.	4
learning how to use the script files	3	using the correct syntax	2
The most interesting topic in today's class was getting to know to solve the problem 3,4,5 on problem set 01.	3	The way the problem needs to be submitted needed more detail.	3
Talking to new people. And listening to the speaker.	3	The script writting and how to save properly was very dificult and not well explained.	4
When Dr. Hynes showed how to use script properly	3	I didn't understand what exactly fprintf did for the program	4

The most interesting thing in today's class was how to program using MATLAB. It is cool how we can program a machine to do work for us.	4	The most confusing thing about today's class was the matrix multiplication vs period character multiplication.	4
Hearing about the using CourseMIRROR to share our thoughts about our class was quite interesting. I like the idea of instructors receiving regular feedback on their teaching.	4	Learning to use MATLAB was rather confusing, but having TAS nearby was helpful.	3
Learning about this app	2	Understanding MATLAB and how to code correctly	3
Learning MATLAB further and doing calculations was interesting.	3	The comma and suppression was not quite clear for me.	4
I though courseMIRROR was the most interesting thing.	3	the most confusing topic was changing the number of decimal places	4
Getting used to coding and MATLAB and discovering the app is both challenging and interesting for me in today's class.	4	-	1
I liked solving mathematical problems using matlab	3	There was nothing that needed more illustration	1
MATLAB vector calculation	2	None	1
I enjoyed learning about the semicolons use in the matlab program. As well as the commands showing how to change the decimals shown and the format command.	4	I did not find anything confusing about today. I enjoy class when we work on our assignments, because the lectures seem to be redundant or do not have much value to our assignments. Todays lecture was nuch better because it helps us complete our given work.	1

The uses of computations in MATLAB	2	How element by element operations can be learned quickly	4
Course mirror and its innovative way to facilitate Professor/student interaction.	3	The free version of MATLAB and it's inconsistent and perplexing performance.	3
I thought the most interesting part of today's class was working on problem 3 of PS01.	3	I did mot know how to use the "fprintf" command in MATLAB.	4
I like how he picked out who will answer his question at the beginning of the class (the one with the longest hair in the group).	4	The last part of Problem 3 was sorta confusing what with the decimal places and also the early part of Problem 4 where rows were involved.	4
I have never coded before so writing lines of data and having them create answers for us is pretty exciting.	4	Saving files is difficult for me as I'm not always sure how to get files into matlab itself vs. my documents on my computer.	4
The thing I found most interesting was the discussion about common issues faced by everyone.	3	The thing I needed more detail on was the process of zipping a folder/file.	4
We can get extra credit	2	How to change how many decimal places are displayed	4
The Matrix calculations in Matlab and how to suppress it	3	How exactly the calculations work because I don't understand how matrix dimension change via calculations	4
The most interesting topic in today's class was understanding notation to denote element by element operations in matlab.	4	The most confusing part of this class is all the formatting needed to submit problem sets.	4

The continued help in matlab has helped me feel more comfortable with the program.	3	Confusing functions in matlab like e^x	4
This new feedback app	2	More practice with scribe	3
I liked how we were taught the thing to remove the extra space in the MATLAB command window.	4	I could not get my Purdue career drive open.	3
Script demonstration	2	Zipping files all together	3
I found the use of MATLAB and the codes very interesting as a first time user. The codes are interesting.	4	The manual was very self explanatory so I wasn't really confused on anything.	1
The most interesting thing in class today was to be able to create script programs on Matlab for the first time.	4	The most confusing part of the day was trying to find the perfect place to submit your files: whether it was on your drive or the Purdue drive.	4
learning how to use matlab and writing codes	3	also how to use matlab since I haven't had any previous experience with it.	3
Course mirror introduction	2	Nothing really	1
Learning how to use MATLAB to do simple and complex calculations.	3	The class today was mostly independent and cpuld have been done outside of class. I didnt feel like i learned much in class today.	4
professor gave more insight about mathlab and learned more about mathlab	3	how to round up the numbers on the answers (the information was given in the class 3 folder)	4

Table 51: Sample of specificity human annotation of an ENGR lecture.

C.2 Model

The specificity prediction model (figure 9) is trained using the CourseMirror specificity data. The model 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.

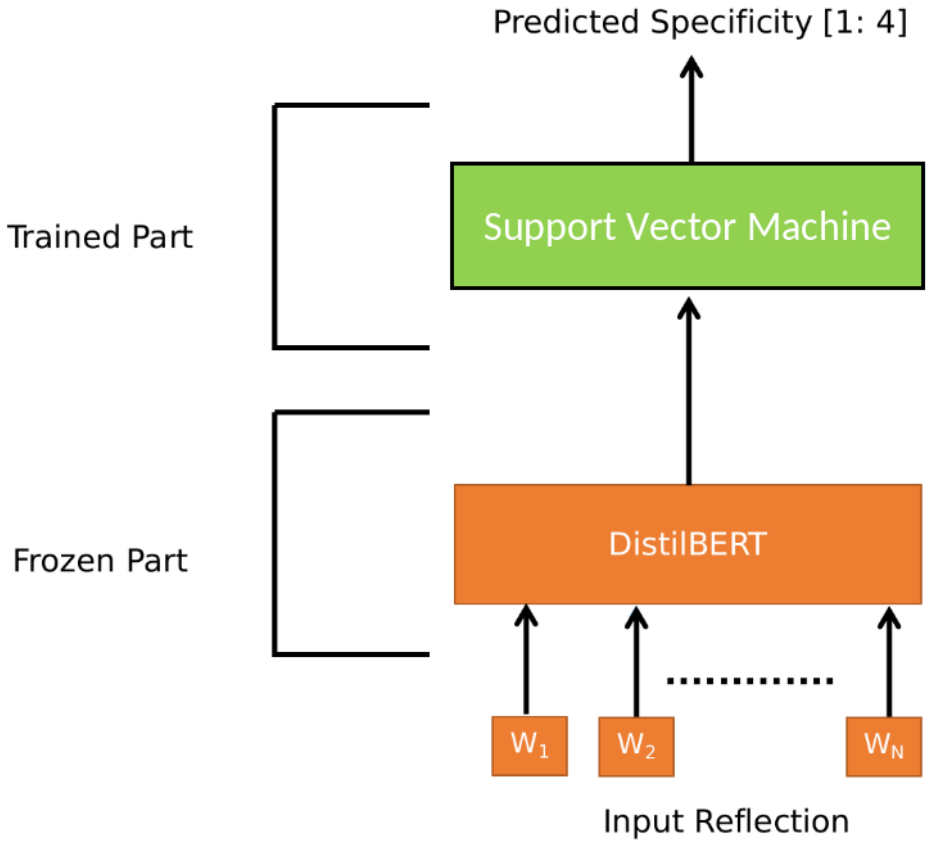


Figure 9: Specificity prediction model used.

C.3 Evaluation

In order to evaluate the performance of the used specificity model, we performed two sets of experiments. First, we evaluated the model using Leave-One-Course-out (LOCO) configuration. We trained the model using 3 out of the four courses while performing testing using the last course. Additionally, we performed an evaluation using a 10-fold cross-validation configuration over all the four courses data. We compared the model with a GloVe baseline line to verify the utility of the used model. Table 52 shows the results of both models using the two configurations.

Model (CM data)	**10-Fold**				**Leave-One-Course-Out**			
	QWK	MSE	MAE	R^2	QWK	MSE	MAE	R^2
GloVe (Baseline)	0.75	0.53	0.38	0.44	0.66	0.73	0.5	0.21
DistilBERT + SVM	**0.84**	**0.33**	**0.26**	**0.66**	**0.79**	**0.42**	**0.33**	**0.56**

Table 52: Predictive performance results (best in **bold**). Lower is better for regression Mean Square Error (MSE) and Mean Absolute Error (MAE), while higher is better for regression R^2 and classifier Quadratic Weighted Kappa (QWK).

Appendix D Human Evaluation

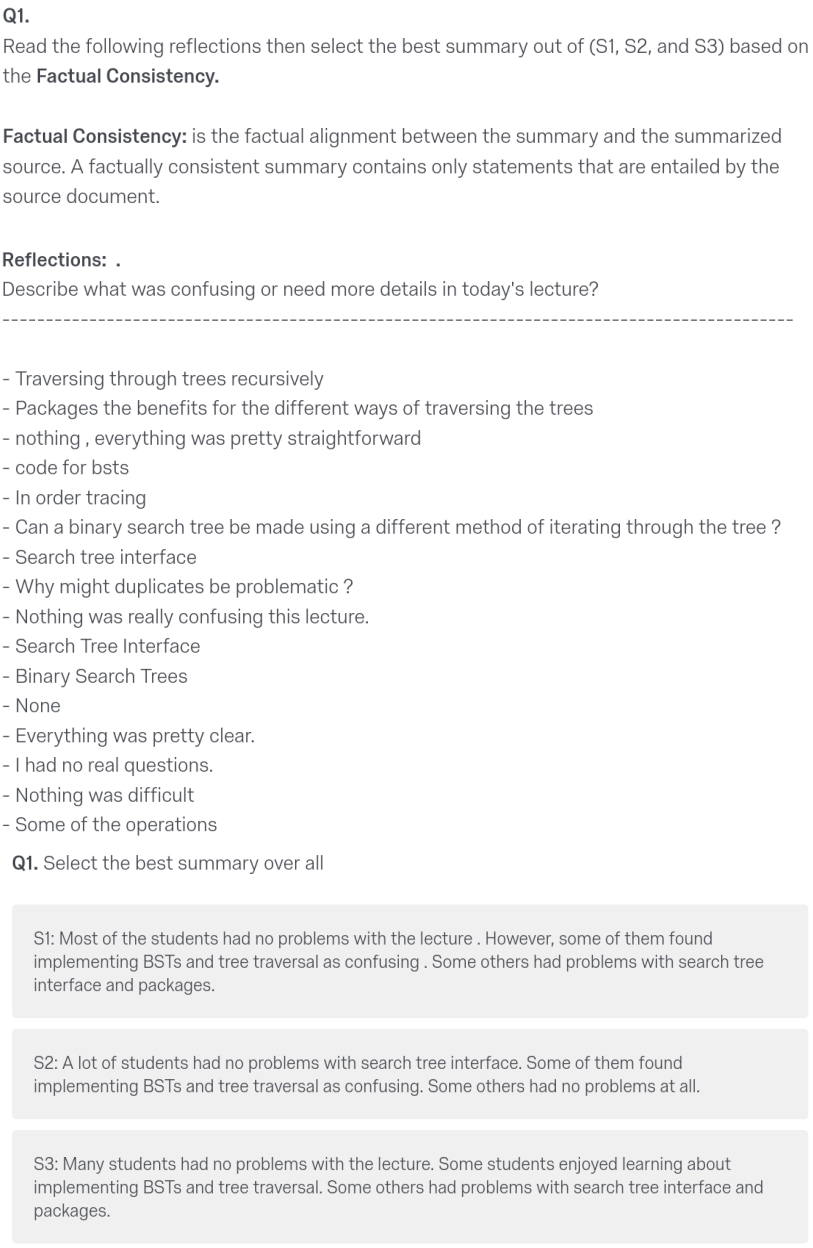


Figure 10: Example of pre human evaluation test for factual consistency aspect

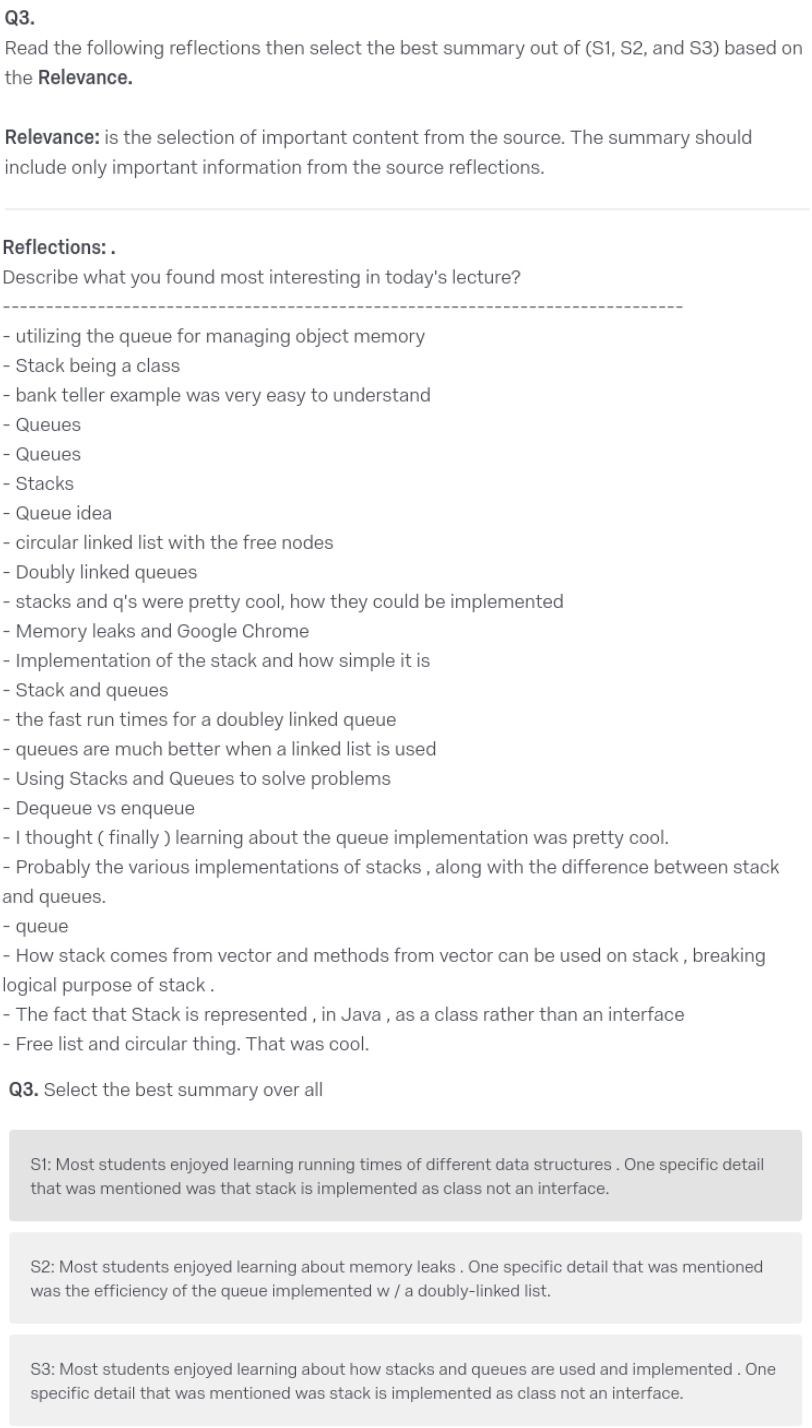
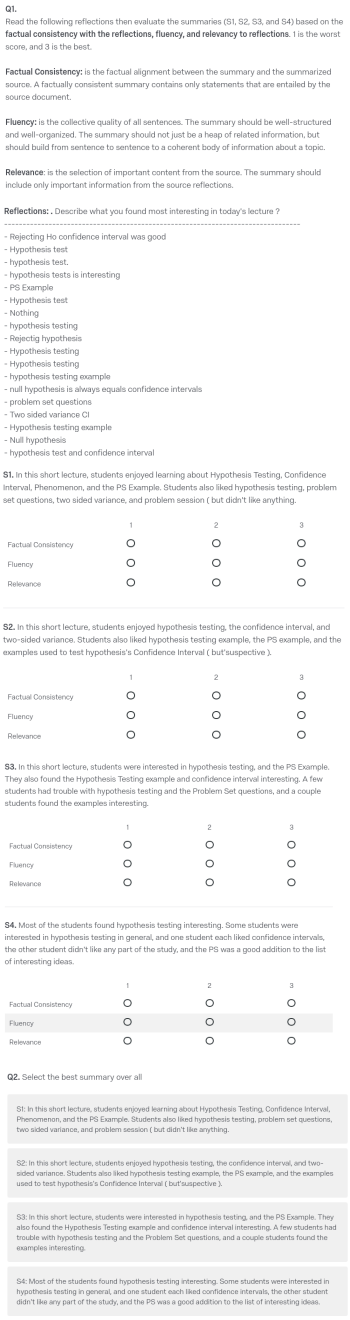


Figure 11: Example of pre human evaluation test for relevancy aspect



Q1.
Read the following reflections then evaluate the summaries (S1, S2, S3, and S4) based on the **factual consistency with the reflections, fluency, and relevancy to reflections**. 1 is the worst score, and 3 is the best.

Factual Consistency: is the factual alignment between the summary and the summarized source. A factually consistent summary contains only statements that are entailed by the source document.

Fluency: is the collective quality of all sentences. The summary should be well-structured and well-organized. The summary should not just be a heap of related information, but should build from sentence to sentence to a coherent body of information about a topic.

Relevance: is the selection of important content from the source. The summary should include only important information from the source reflections.

Reflections: . Describe what you found most interesting in today's lecture ?

- Rejecting Ho confidence interval was good
- Hypothesis test
- hypothesis test.
- hypothesis tests is interesting
- PS Example
- Hypothesis test
- Nothing
- hypothesis testing
- Rejectig hypothesis
- Hypothesis testing
- Hypothesis testing
- hypothesis testing example
- null hypothesis is always equals confidence intervals
- problem set questions
- Two sided variance CI
- Hypothesis testing example
- Null hypothesis
- hypothesis test and confidence interval

S1. In this short lecture, students enjoyed learning about Hypothesis Testing, Confidence Interval, Phenomenon, and the PS Example. Students also liked hypothesis testing, problem set questions, two sided variance, and problem session (but didn't like anything.

	1	2	3
Factual Consistency	○	○	○
Fluency	○	○	○
Relevance	○	○	○

S2. In this short lecture, students enjoyed hypothesis testing, the confidence interval, and two-sided variance. Students also liked hypothesis testing example, the PS example, and the examples used to test hypothesis's Confidence Interval (but'suspective).

	1	2	3
Factual Consistency	○	○	○
Fluency	○	○	○
Relevance	○	○	○

S3. In this short lecture, students were interested in hypothesis testing, and the PS Example. They also found the Hypothesis Testing example and confidence interval interesting. A few students had trouble with hypothesis testing and the Problem Set questions, and a couple students found the examples interesting.

	1	2	3
Factual Consistency	○	○	○
Fluency	○	○	○
Relevance	○	○	○

S4. Most of the students found hypothesis testing interesting. Some students were interested in hypothesis testing in general, and one student each liked confidence intervals, the other student didn't like any part of the study, and the PS was a good addition to the list of interesting ideas.

	1	2	3
Factual Consistency	○	○	○
Fluency	○	○	○
Relevance	○	○	○

Q2. Select the best summary over all

S1: In this short lecture, students enjoyed learning about Hypothesis Testing, Confidence Interval, Phenomenon, and the PS Example. Students also liked hypothesis testing, problem set questions, two sided variance, and problem session (but didn't like anything.

S2: In this short lecture, students enjoyed hypothesis testing, the confidence interval, and two-sided variance. Students also liked hypothesis testing example, the PS example, and the examples used to test hypothesis's Confidence Interval (but'suspective).

S3: In this short lecture, students were interested in hypothesis testing, and the PS Example. They also found the Hypothesis Testing example and confidence interval interesting. A few students had trouble with hypothesis testing and the Problem Set questions, and a couple students found the examples interesting.

S4: Most of the students found hypothesis testing interesting. Some students were interested in hypothesis testing in general, and one student each liked confidence intervals, the other student didn't like any part of the study, and the PS was a good addition to the list of interesting ideas.

Figure 12: Example of CM annotation sample.

Q1.
Read the following reflections then evaluate the summaries (S1, S2, S3, and S4) based on the **factual consistency with the reflections, fluency, and relevancy to reflections**. 1 is the worst score, and 3 is the best.

Factual Consistency: is the factual alignment between the summary and the summarized source. A factually consistent summary contains only statements that are entailed by the source document.

Fluency: is the collective quality of all sentences. The summary should be well-structured and well-organized. The summary should not just be a heap of related information, but should build from sentence to sentence to a coherent body of information about a topic.

Relevance: is the selection of important content from the source. The summary should include only important information from the source reflections.

Reflections: . Very different from what I'm used to which is a regular freezer ice maker. It's kind of a process if you use all the time, which I do, you develop a system. Makes ice fast. Simple to operate.

Gave this as a gift to my daughter and son-in-law for their deck. The have numerous pool parties and according to my daughter, this works great. It prevents wet feet from going in the house to get ice. It fits nicely on their deck and they seem to be enjoying it.

I thought this was exactly like my current ice maker but when it got here it wasn't. Once seeing it I wondered what size it was going to make but was pleasantly surprised at the size of the large cubes. It ended up being all it was advertised to be and I love it. I would recommend it to all.

This thing works like a charm. We use it in our bar at home and in our RV when traveling. You just have to remember to empty it when not in use and keep it clean. It does the job for parties! Very fast and efficient.

absolutely love it!!! this is the 2nd one we have. The 1st one lasted 8 years!!! I love having ice made all the time. I have it made by the refrigerator also but I love this machine better.

It worked as advertised although I didn't check to see if it make the actual 27 pounds per day. I bought it for the camper and seemed to satisfy the demand for ice. It sure beats paying $3 / bag at the camp store. It won't take long to pay for it.

So far I love this device. It cranks out ice in minutes. Perfect to have for entertaining. We actually bought this as a back up for our unreliable built in ice maker in our fridge. This has saved the day.

We have had this machine for 2 years. It sits outside on our patio in Arizona. It hits 115 degrees in the summer. It has been cleaned one a month and has NEVER stopped or had any kind of problems. We just keep filling it with water and it keeps making ice 24 / 7 / 365.

S1. This ice maker is very fast and efficient. It cranks out ice in minutes. It does the job of making ice for parties. It is easy to operate and easy to empty the ice. It can be used in bar, on the bar or in the RV. Overall, it is recommended.

	1	2	3
Factual Consistency	○	○	○
Fluency	○	○	○
Relevance	○	○	○

S2. This ice maker is very fast and efficient. It cranks out ice in minutes and makes ice 24 times a day. It is a great tool for entertaining and is great for making ice on a patio. It does the job of making ice for pool parties, bar and RV parties.

	1	2	3	4
Factual Consistency	○	○	○	○
Fluency	○	○	○	○
Relevance	○	○	○	○

S3. This ice maker is very fast and efficient. It cranks out ice in minutes and keeps making ice 24 / 7 / 365. It is a great tool for entertaining and prevents wet feet from going in the house to get ice. It sure beats paying $3 / bag at the camp store.

	1	2	3
Factual Consistency	○	○	○
Fluency	○	○	○
Relevance	○	○	○

S4. This ice maker is very fast and efficient. It cranks out ice in minutes. It does the job for parties and is easy to use. It is a great way to make ice. It can be used as a back up for our unreliable built in ice maker in our fridge.

	1	2	3
Factual Consistency	○	○	○
Fluency	○	○	○
Relevance	○	○	○

Q2. Select the best summary over all

S1: This ice maker is very fast and efficient. It cranks out ice in minutes. It does the job of making ice for parties. It is easy to operate and easy to empty the ice. It can be used in bar, on the bar or in the RV. Overall, it is recommended.

S2: This ice maker is very fast and efficient. It cranks out ice in minutes and makes ice 24 times a day. It is a great tool for entertaining and is great for making ice on a patio. It does the job of making ice for pool parties, bar and RV parties.

S3: This ice maker is very fast and efficient. It cranks out ice in minutes and keeps making ice 24 / 7 / 365. It is a great tool for entertaining and prevents wet feet from going in the house to get ice. It sure beats paying $3 / bag at the camp store.

S4: This ice maker is very fast and efficient. It cranks out ice in minutes. It does the job for parties and is easy to use. It is a great way to make ice. It can be used as a back up for our unreliable built in ice maker in our fridge.

Figure 13: Example of AY annotation sample.

Bibliography

[1] Amplayo, R. K., Angelidis, S., and Lapata, M. (2021). Aspect-controllable opinion summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6578–6593.

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

[3] Bajaj, A., Dangati, P., Krishna, K., Kumar, P. A., Uppaal, R., Windsor, B., Brenner, E., Dotterrer, D., Das, R., and McCallum, A. (2021). Long document summarization in a low resource setting using pretrained language models. *arXiv preprint arXiv:2103.00751*.

[4] Berg-Kirkpatrick, T., Gillick, D., and Klein, D. (2011). Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 481–490.

[5] Böhm, F., Gao, Y., Meyer, C. M., Shapira, O., Dagan, I., and Gurevych, I. (2019). Better rewards yield better summaries: Learning to summarise without references. 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 3101–3111.

[6] Bražinskas, A., Lapata, M., and Titov, I. (2020). Few-shot learning for opinion summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4119–4135.

[7] Cao, Z., Li, W., Li, S., and Wei, F. (2018). Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 152–161.

[8] Chan, H. P., Chen, W., and King, I. (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 '20, page 1191–1200, New York, NY, USA. Association for Computing Machinery.

[9] Chen, J., Yang, Z., and Yang, D. (2020). Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. pages 2147–2157.

[10] Chen, P., Wu, F., Wang, T., and Ding, W. (2018). A semantic qa-based approach for text summarization evaluation. In McIlraith, S. A. and Weinberger, K. Q., editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI*

Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 4800–4807. AAAI Press.

[11] Chen, Y., Ma, Y., Mao, X., and Li, Q. (2019). Multi-task learning for abstractive and extractive summarization. *Data Science and Engineering*, 4(1):14–23.

[12] Chen, Y.-C. and Bansal, M. (2018). Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proc. of ACL*, pages 675–686.

[13] Clark, E., Celikyilmaz, A., and Smith, N. A. (2019). Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2748–2760.

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

[15] Dolan, W. B. and Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

[16] Du, J., Grave, E., Gunel, B., Chaudhary, V., Celebi, O., Auli, M., Stoyanov, V., and Conneau, A. (2020). Self-training improves pre-training for natural language understanding. *arXiv e-prints*, pages arXiv–2010.

[17] Durmus, E., He, H., and Diab, M. T. (2020). Feqa: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *ACL*.

[18] Edmundson, H. P. (1969). New methods in automatic extracting. *Journal of the ACM (JACM)*, 16(2):264–285.

[19] Fabbri, A. R., Kryscinski, W., McCann, B., Xiong, C., Socher, R., and Radev, D. (2021). Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

[20] Fadaee, M., Bisazza, A., and Monz, C. (2017). Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573.

[21] Fan, X., Luo, W., Menekse, M., Litman, D., and Wang, J. (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.

[22] Ganesan, K., Zhai, C., and Han, J. (2010). Opinosis: A graph based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 340–348, Beijing, China. Coling 2010 Organizing Committee.

[23] Gehrmann, S., Deng, Y., and Rush, A. M. (2018). Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109.

[24] Gerani, S., Mehdad, Y., Carenini, G., Ng, R., and Nejat, B. (2014). Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1602–1613.

[25] Guo, H., Pasunuru, R., and Bansal, M. (2018a). Soft layer-specific multi-task summarization with entailment and question generation. *arXiv preprint arXiv:1805.11004*.

[26] Guo, S., Huang, W., Zhang, H., Zhuang, C., Dong, D., Scott, M. R., and Huang, D. (2018b). Curriculumnet: Weakly supervised learning from large-scale web images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150.

[27] Hacohen, G. and Weinshall, D. (2019). On the power of curriculum learning in training deep networks. In *International Conference on Machine Learning*, pages 2535–2544.

[28] Hasan, K. S. and Ng, V. (2014). Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1262–1273.

[29] He, R. and McAuley, J. (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*, pages 507–517.

[30] Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701.

[31] Hsu, W.-T., Lin, C.-K., Lee, M.-Y., Min, K., Tang, J., and Sun, M. (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. Association for Computational Linguistics.

[32] Hua, X. and Wang, L. (2017). A pilot study of domain adaptation effect for neural abstractive summarization. *EMNLP 2017*, page 100.

[33] Isonuma, M., Fujino, T., Mori, J., Matsuo, Y., and Sakata, I. (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.

[34] Jadhav, A. and Rajan, V. (2018). Extractive summarization with swap-net: Sentences and words from alternating pointer networks. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 142–151.

[35] Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. (2018). Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pages 2304–2313.

[36] Keneshloo, Y., Ramakrishnan, N., and Reddy, C. K. (2019). Deep transfer reinforcement learning for text summarization. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 675–683. SIAM.

[37] Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

[38] Koehn, P. and Knowles, R. (2017). Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

[39] Krishna, K., Wieting, J., and Iyyer, M. (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.

[40] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

[41] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (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.

[42] Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

[43] Liu, L., Lu, Y., Yang, M., Qu, Q., Zhu, J., and Li, H. (2018). Generative adversarial network for abstractive text summarization. In *AAAI*.

[44] Liu*, P. J., Saleh*, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

[45] Liu, Y. and Lapata, M. (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 3721–3731.

[46] Liu, Y. and Liu, P. (2021). Simcls: A simple framework for contrastive learning of abstractive summarization. *arXiv preprint arXiv:2106.01890*.

[47] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[48] Liu, Z. and Chen, N. (2021). Controllable neural dialogue summarization with personal named entity planning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 92–106.

[49] Liu, Z., Li, P., Zheng, Y., and Sun, M. (2009). Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 257–266.

[50] Lu, Y., Liu, L., Jiang, Z., Yang, M., and Goebel, R. (2019). A multi-task learning framework for abstractive text summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9987–9988.

[51] Luo, W. (2017). *Automatic Summarization for Student Reflective Responses*. PhD thesis, University of Pittsburgh.

[52] Luo, W. and Litman, D. (2015). Summarizing student responses to reflection prompts. In *Proc. of EMNLP*, pages 1955–1960.

[53] Luo, W. and Litman, D. (2016). Determining the quality of a student reflective response. In *The twenty-ninth international FLAIRS Conference*.

[54] Luo, W., Liu, F., and Litman, D. (2016a). An improved phrase-based approach to annotating and summarizing student course responses. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 53–63.

[55] Luo, W., Liu, F., Liu, Z., and Litman, D. (2016b). Automatic summarization of student course feedback. In *Proceedings of NAACL-HLT*, pages 80–85.

[56] Ma, S., Sun, X., Lin, J., and Ren, X. (2018). A hierarchical end-to-end model for jointly improving text summarization and sentiment classification. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4251–4257.

[57] Magooda, A. and Litman, D. (2020a). Abstractive summarization for low resource data using domain transfer and data synthesis.

[58] Magooda, A. and Litman, D. (2020b). Abstractive summarization for low resource data using domain transfer and data synthesis. In *The Thirty-Third International Flairs Conference*.

[59] Magooda, A. and Litman, D. (2021). Mitigating data scarceness through data synthesis, augmentation and curriculum for abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2043–2052.

[60] Magooda, A., Litman, D., and Elaraby, M. (2021). Exploring multitask learning for low-resource abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1652–1661.

[61] Magooda, A. and Marcjan, C. (2020). Attend to the beginning: A study on using bidirectional attention for extractive summarization. *arXiv preprint arXiv:2002.03405*.

[62] Martins, A. F. and Smith, N. A. (2009). Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 1–9.

[63] Medelyan, O., Frank, E., and Witten, I. H. (2009). Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 1318–1327.

[64] Menekse, M., Stump, G., Krause, S. J., and Chi, M. T. (2011). The effectiveness of students' daily reflections on learning in engineering context. In *ASEE Annual Conf. and Exposition*.

[65] Miller, G. A. (1998). *WordNet: An electronic lexical database*. MIT press.

[66] Nallapati, R., Zhai, F., and Zhou, B. (2017). Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[67] Nallapati, R., Zhou, B., dos Santos, C., glar Gulçehre, Ç., and Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. *CoNLL 2016*, page 280.

[68] Napoles, C., Gormley, M., and Van Durme, B. (2012). 'annotated english gigaword. *Linguistic Data Consortium, Philadelphia*.

[69] Nenkova, A. and McKeown, K. (2011). *Automatic Summarization*, volume 5.

[70] Ouyang, Y., Li, W., Wei, F., and Lu, Q. (2009). Learning similarity functions in graph-based document summarization. In *International Conference on Computer Processing of Oriental Languages*, pages 189–200. Springer.

[71] Oya, T., Mehdad, Y., Carenini, G., and Ng, R. (2014). A template-based abstractive meeting summarization: Leveraging summary and source text relationships. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 45–53.

[72] Parida, S. and Motlicek, P. (2019). Abstract text summarization: A low resource challenge. In *Proc. of (EMNLP-IJCNLP)*, pages 5996–6000.

[73] Pasunuru, R., Guo, H., and Bansal, M. (2017). Towards improving abstractive summarization via entailment generation. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 27–32.

[74] Paulus, R., Xiong, C., and Socher, R. (2018). A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

[75] Pruksachatkun, Y., Phang, J., Liu, H., Htut, P. M., Zhang, X., Pang, R. Y., Vania, C., Kann, K., and Bowman, S. R. (2020). Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work? *arXiv preprint arXiv:2005.00628*.

[76] Radev, D. R., Jing, H., Styś, M., and Tam, D. (2004). Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6):919–938.

[77] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

[78] Rose, S., Engel, D., Cramer, N., and Cowley, W. (2010). Automatic keyword extraction from individual documents. *Text mining: applications and theory*, pages 1–20.

[79] Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

[80] Sachan, M. and Xing, E. (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.

[81] Sachan, M. and Xing, E. (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.

[82] Sandhaus, E. (2008). The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

[83] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

[84] Sarkhel, R., Keymanesh, M., Nandi, A., and Parthasarathy, S. (2020). Interpretable multi-headed attention for abstractive summarization at controllable lengths. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6871–6882.

[85] Scialom, T., Dray, P.-A., Lamprier, S., Piwowarski, B., Staiano, J., Wang, A., and Gallinari, P. (2021). QuestEval: Summarization asks for fact-based evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*,

pages 6594–6604, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[86] See, A., Liu, P. J., and Manning, C. D. (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.

[87] Sennrich, R., Haddow, B., and Birch, A. (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.

[88] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[89] Tan, C., Wei, F., Yang, N., Du, B., Lv, W., and Zhou, M. (2018). S-net: From answer extraction to answer synthesis for machine reading comprehension. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[90] Tay, Y., Wang, S., Luu, A. T., Fu, J., Phan, M. C., Yuan, X., Rao, J., Hui, S. C., and Zhang, A. (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.

[91] Thaker, K., Brusilovsky, P., and He, D. (2019). Student modeling with automatic knowledge component extraction for adaptive textbooks. In *iTextbooks@ AIED*, pages 95–102.

[92] Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700.

[93] Wang, W., Tian, Y., Ngiam, J., Yang, Y., Caswell, I., and Parekh, Z. (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.

[94] Wang, W. Y. and Yang, D. (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.

[95] Wang, X., Pham, H., Dai, Z., and Neubig, G. (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.

[96] Wieting, J. and Gimpel, K. (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.

[97] Wu, Y.-f. B., Li, Q., Bot, R. S., and Chen, X. (2005). Domain-specific keyphrase extraction. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 283–284.

[98] Xu, B., Zhang, L., Mao, Z., Wang, Q., Xie, H., and Zhang, Y. (2020). Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6095–6104.

[99] Yan, Y., Qi, W., Gong, Y., Liu, D., Duan, N., Chen, J., Zhang, R., and Zhou, M. (2020). Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *arXiv preprint arXiv:2001.04063*.

[100] Yao, J.-g., Wan, X., and Xiao, J. (2015). Phrase-based compressive cross-language summarization. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 118–127.

[101] Yu, N., Huang, M., Shi, Y., and Zhu, X. (2016). Product review summarization by exploiting phrase properties. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1113–1124.

[102] Yu, T., Liu, Z., and Fung, P. (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.

[103] Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

[104] Zhong, M., Liu, P., Chen, Y., Wang, D., Qiu, X., and Huang, X. (2020). Extractive summarization as text matching. In *ACL*, pages 6197–6208.