Argument Summarization: Enhancing Summary Generation and Evaluation Metrics

by

Mohammad Khosravani

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science with Specialization in Artificial Intelligence

Faculty of Science and Environmental Studies Department of Computer Science
Lakehead University

Abstract

In the current era of mass digital information, the need for effective argument summarization has become paramount. This thesis explores the domain of argument summarization, focusing on the development of techniques and evaluation metrics to improve the quality of summarization models. The study first investigates the task of key point analysis, and the challenges associated with previous approaches to it, emphasizing the significance of coverage of the summary. To address these challenges, we propose a novel clustering-based framework that leverages the inherent semantics of arguments to identify and group similar arguments. The proposed approach is evaluated on the benchmark dataset and compared with previous state-of-the-art methods, demonstrating its effectiveness. In addition to the proposed framework, this thesis also presents an analysis of the previous evaluation metric for argument summarization. Commonly used metric, ROUGE is evaluated, revealing its limitation in capturing the nuanced aspects of argument quality. To this end, we introduce new evaluation metrics and methods that consider the coverage and redundancy of the generated summaries, providing more accurate and informative assessments of summarization models. We further show that our evaluation metric has a better correlation with actual summary quality, whereas previous metrics fail to capture this correlation. The implementation is available online[1].

[1]https://github.com/b14ck-sun/arg-sum

Preface

This thesis integrates sections from papers that have been previously published or are about to be submitted. The paper represents original work conducted by me, under the guidance of Professor Amine Trabelsi. I am the lead author responsible for defining the problem, carrying out the implementations, conducting the experimental evaluations, and writing the papers.

This thesis has used ChatGPT (GPT-3.5) for the purpose of rephrasing and clarifying paragraphs. No information or results were generated by AI. The prompt used for rephrasing was "Rephrase the following paragraph" in addition to the text. The generated response was reviewed and edited, if necessary, by the author.

1. M Khosravani, C Huang, A Trabelsi, "Enhancing Key Point Generation via Clustering: Prioritizing Exhaustiveness and Introducing an Automatic Coverage Evaluation Metric", NAACL 2024, Main Conference [1] (To Appear)

2. M Khosravani, A Trabelsi, "Recent Trends in Unsupervised Summarization", arXiv 2023 [2]

Table of Contents

List of Tables

List of Figures

Abbreviations

AI Artificial Intelligence.

BERT Bidirectional Encoder Representations from Transformers.

BLEU BiLingual Evaluation Understudy.

GKP Generated Key Points.

HDBSCAN Hierarchical Density-Based Spatial Clustering of Applications with Noise.

KP Key Point.

KPA Key Point Analysis.

KPG Key Point Generation.

KPM Key Point Matching.

ML Machine Learning.

NLP Natural Language Processing.

NN Neural Networks.

RKP Reference Key Points.

RoBERTa Robustly Optimized BERT Pretraining Approach.

ROUGE Recall-Oriented Understudy for Gisting Evaluation.

SBERT Sentence-BERT.

TSNE t-distributed stochastic neighbor embedding.

UMAP Uniform Manifold Approximation and Projection.

Chapter 1

Introduction

In the current digital landscape, vast amounts of information are generated and disseminated daily, and the need for efficient information processing and comprehension has become more critical than ever before. Online arguments and debates on social media platforms are an example of such data, and summarizing them is a pivotal yet relatively understudied area of research in the realm of natural language processing (NLP) and text mining. The objective of argument summarization is to automatically identify the main points and generate a summary from debates, discussions, and arguments, presenting a condensed and coherent representation of the underlying aspects. While previous approaches have made some progress in this domain, the complexity and diversity of arguments presents ongoing challenges, which limits the effectiveness of previous works. This thesis delves into the exploration of an innovative approach for argument summarization in addition to a novel semi-supervised approach for evaluating generated summaries.

1.1 Text Summarization

Summarization is the task of generating a concise and fluent summary of the input text, where the main points are covered. Summarization methods can be classified into different categories based on their characteristics such as the technique, number of inputs, summary type, and the domain. The first category, technique, is usually

either abstractive or extractive, where the abstractive methods generate their summary based on their understanding of the text, often done using a hidden representation. On the other hand, extractive methods find and select the salient sentences and phrases from the input and present them as the summary. The second category classifies summarization models based on the number of inputs namely single document summarization and multi-document summarization. Summary type is often either general or aspect-based. General summaries aim to summarize the main points of a text, whereas aspect-based summaries focus on a specific angle and are often used in review or opinion summarization. Lastly, domain describes the domain of input text, most summarization methods focus on News as there is an abundance of datasets and benchmarks available for it. However, recently there has been a number of works and datasets in other domains such as review, scientific paper, social media summarization.

1.2 Motivation

In this thesis we aim to perform summarization on the argumentative domain. Automatic summarization of online debates and arguments is an important, yet understudied part of automatic text summarization. Over the past decade the main focus of research on text summarization was on News, mainly due to the availability of various datasets and benchmarks. As the number of online arguments increases due to the popularity of social media, summarizing online debates becomes more important since the ability to accurately summarize an online debate on a particular topic provides insights on different viewpoints about the topic. This could be could benefit a wide range of audience, from governments gauging people's opinions towards a policy, companies receiving feedback on a service or product, and even ordinary citizen by helping them understand different view points on a recent topic. However, summarizing online debates introduces some additional challenges compared to traditional News summarization such as use of informal language, and working with

multiple documents. Over the recent years, there has been an increase in argument summarization research with the introduction of the ArgKP [3] dataset, and the Key Point Analysis (KPA) shared task [4].

1.3 Key Point Analysis

Key Point Analysis is concerned with generating a concise summary using arguments on a specific topic. It was first introduced by BarHaim [3] alongside the ArgKP dataset, and was later a shared task on the Argument Mining Workshop at EMNLP 2021 [4]. In key point analysis the input is a collection of arguments on different topics with pro and against stances, where each argument could belong to a key point. Key points are the main talking points on a specific topic, and can be used as a summary of the topic. For example, for the topic of "Child vaccination should be mandatory", the key point "Child vaccination is proven to be effective" is a key point with a pro stance on the topic. Table 1.1 shows an example of a topic in the dataset alongside its reference key points and the number of arguments that discuss it.

The task of key point analysis consists of two tracks, key point matching and key point generation. In key point matching, the goal is to match arguments to key points, and both the arguments and key points are given as the input. In key point generation, the arguments are provided and the key points are generated from them. Specifically, the goal is to generate the summary in a short bullet-point style list format. The generated key points should ideally cover the main talking points of all arguments. Key points should be concise, informative, and not too general or specific. In the KPA task, the term "generated key points" refers to the summary or output generated by the model and the model itself is referred as a KPG model. The KPG model can be an extractive, abstractive, or hybrid summarization model. Moreover the terms key point generation and argument summarization are used interchangeably throughout the text as the goal of the KPG model is to summarize a group of arguments.

Homeschooling should be banned	#Args
Pro	
Mainstream schools are essential to develop social skills.	65
Parents are not qualified as teachers.	20
Homeschools cannot be regulated/standardized.	15
Mainstream schools are of higher educational quality.	9
Con	
Parents should be permitted to choose the education of their children.	28
Homeschooling is often the best option for catering for the needs of exceptional/ religious/ill/disabled student.	25
Homeschools can be personalized to the child's pace/needs.	21
Mainstream schools have a lot of violence/bullying.	21
The home is a good learning environment.	13
Parents will have more ability to pay-attention/educate their child.	7

Table 1.1: A sample key point-based summary, extracted from the ArgKP dataset.

1.3.1 Problem Specification

The goal of KPG task is to generated a bullet point-style list of sentence, called generated key points, given a collection of input arguments on a topic. Formally, given input collection of documents D on topic T, where D_T={Argument 1, Argument 2, Argument 3, ..., Argument n}, the generated key points (GKP), G_T={GKP 1, GKP 2, ..., GKP k}. For the ArgKP dataset, the ideal value of k is often between 8 to 12, depending on the topic.

1.4 Previous Approaches & Challenges

During the Argument Mining Workshop, a total of 17 submissions were made for key point matching, and 6 submissions were for key point generation. Furthermore, there have been some prior works dedicated to this task in the past two years. We conducted a thorough analysis of previous approaches to this task, assessing the quality of their

outputs. Our experiments revealed several limitations in these prior approaches.

Firstly, the top-performing methods in the workshop relied on a quality evaluation model, which assessed the quality of input arguments, and ignored low quality arguments. However, this approach does not effectively assess the quality of argument, thereby introducing error and is proved to be time-consuming, taking approximately 8 seconds for each arguments.

Secondly, our experiments showed that the generated summaries had low coverage, failing to encompass many key points from their reference summaries. This was primarily because these methods tended to select multiple similar arguments for a few popular key points instead of choosing arguments that covered various key points.

Thirdly, the methods assumed the availability of stance information about an argument, which is not readily accessible when working with real-world data. Lastly, the leading methods removed sentences beginning with pronouns to enhance the quality of the generated summary. In Section 3 Methodology, we delve deeper into the issues with previous approaches.

In addition to reviewing methods, we also examined the evaluation metrics used in the summarization task. The main challenge in evaluation of summarizes is the lack of reliable automatic metrics, that effectively correlate a good summary with a high score. The key point analysis task employed human judges for evaluation, which is not easily reproducible or scalable due to the substantial cost and human resources required, particularly as the number of works in this area increases. Conversely, other summarization methods employed ROUGE score [5] for evaluating summaries. ROUGE score is a is a set of metrics that measure the overlap between the generated summary and the reference summary. Our experiments demonstrated that ROUGE is not an ideal metric, as it cannot effectively distinguish between summaries with different coverage and quality. We show that the ROUGE score was incapable of differentiating between summaries that cover all the key points in the reference summary and summaries that only cover half the key points. Further details on our

experiments can be found in Section 4 Evaluation Metrics.

1.5 Contribution

Our approach to **argument summarization** and key point generation is designed to address the shortcomings of previous methods. We employ a method based on extractive clustering. Through semantic-based clustering, we group similar arguments, and a matching model is then utilized to identify the argument that best exemplifies each cluster. The chosen representatives of the clusters serve as the summary. This approach enables the representation of less popular key points, specifically smaller clusters. Additionally, the use of a matching model in argument selection eliminates the need for a filtering step, which not only restricts the extracted arguments but also introduces additional errors and significantly hampers the process. Furthermore, we showcase the versatility of our model by comparing it to previous approaches on datasets beyond the traditional ArgKP, which was the primary focus of earlier methods.

In addition to the framework itself, we introduce new **evaluation metrics** for key point generation, namely coverage and redundancy. We also experiment with various techniques that aim to accurately predict these metrics in unsupervised settings. For coverage evaluation, we use a KPM model that determines whether a given generated key point matches with a reference or ground truth key point. The coverage score quantifies the proportion of matched reference key points in the summary. The redundancy model, computes a similarity score between different all possible generated key point pairs to find similar outputs. Redundancy score represents the percentage of similar pairs to all possible pairs. Our experiments confirm the efficacy and appropriateness of the proposed metric compared to traditionally employed ones.

In short our contributions are as follows:

1. Introducing a novel clustering-based approach for argument summarization and

key point generation with a focus on coverage

2. Proposing two new automatic evaluation metrics for evaluating coverage and redundancy

3. A set of datasets for assessing the effectiveness of evaluation metrics, based on ArgKP dataset

1.6 Structure

The structure of this thesis is as follows: Section 2 serves as a background to models and techniques employed in the thesis alongside the previous work on argument mining and summarization. Section 3 focuses on the proposed methodology and framework. Section 4 introduces the new evaluation metric and its variations. Section 5 discusses the experiments and results.

Chapter 2

Background & Related Works

2.1 Overview

Overview: In this chapter we cover the basic the concept and tasks in natural language processing that is used in our method and evaluation metrics. Next, we examine the research in argument mining and the previous approaches to key point analysis task.

2.2 Background

2.2.1 Artificial Intelligence

Artificial Intelligence or AI refers to the simulation of human intelligence in machines, enabling them to perform tasks that typically require human intelligence. AI encompasses a wide range of technologies, approaches, and applications designed to enable machines to think, reason, learn, perceive, and interact with their environment in ways that mimic human intelligence.

2.2.2 Machine Learning

Machine learning is a subset of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computer systems to improve their performance on a specific task through learning and experience. In machine learning, computers are trained to recognize patterns, make predictions, or solve problems based on data rather than being explicitly programmed for each specific

task.

2.2.3 Natural Language Processing

NLP, or Natural Language Processing, is a subfield of artificial intelligence that focuses on the interaction between computers and human language. Its primary goal is to enable computers to understand, interpret, and generate human language in a valuable way. NLP encompasses a wide range of tasks and applications related to language understanding and generation.

2.2.4 NLP Tasks: Paraphrase Detection & Entailment

Paraphrase detection and entailment are two of the popular NLP tasks. Both tasks are binary classification tasks where the goal is to predict a label (0 or 1), given a pair of sentences as input. In paraphrase detection, the goal is to predict whether the input sentences are paraphrased or not, i.e. conveying the same information with different wording and structure. In entailment, the goal is to predict if the first sentence entails the second second. We use entailment and paraphrase detection in the proposed method and the evaluation metric.

2.2.5 Neural Networks

Neural networks are computational models inspired by the structure and functioning of biological neural networks in the human brain. They are a fundamental component of artificial intelligence and machine learning. Neural networks consist of interconnected nodes, often referred to as neurons, organized in layers. These neurons process and transform input data by performing weighted sums of their inputs, applying activation functions to introduce non-linearity, and passing the results to subsequent layers. Through a process called training, neural networks adjust the weights of their connections to learn and recognize complex patterns and relationships within data. This enables them to be used for a wide range of tasks, including image and

speech recognition, natural language processing, and various other machine learning applications.

Deep Learning

Deep learning is a sub-field of machine learning that focuses on artificial neural networks with multiple layers, often referred to as deep neural networks. These deep neural networks are designed to automatically learn and represent data in increasingly abstract and hierarchical ways. They are called "deep" because they have more layers (depth) than traditional neural networks, allowing them to model complex patterns and relationships in data. Deep learning has gained significant attention and popularity due to its remarkable success in various applications, including image and speech recognition, natural language processing, and reinforcement learning. Deep learning has the ability to discover intricate features and patterns in large datasets through their training process.

The training in deep learning models is comprised of multiple blocks. First a model architecture (e.g. transformer) is selected and weights of neurons are randomly initialized. Second, a loss function is selected to calculate the error between the model's predictions and the actual target. Third, an optimizer is selected to update the weights of neurons based on the loss function. The optimizer adjusts the model's parameters during training in a way that reduces the loss function's error.

Combining these components, a single training loop (i.e. epoch) of a deep learning model includes the following steps. First, the input data is fed into the neural network, and feedforward propagation occurs in order to generate the predictions based on model's current parameters. Second, the loss function computes the error between the predictions and the actual labels. Third, backpropagation happens where gradients of the loss with respect to the model's parameters (i.e. weights) are computed using the chain rule of calculus. Fourth, in gradient descent the optimizer uses the gradients to update the model's parameters.

Loss Function

A loss function, also known as a cost function or objective function, quantifies the error or discrepancy between the model's predictions and the actual target values in the training dataset. The goal during training is to minimize this loss function. The choice of the appropriate loss function depends on the nature of the problem and the data available for training. Different loss functions used in this thesis are:

- Cross-entropy loss: Cross-entropy loss is frequently used for classification tasks, especially in the context of binary or multiclass classification. It quantifies the dissimilarity between the predicted class probabilities and the true class labels. The formula for binary cross-entropy loss is:

$$loss = -[Y * log(Y_{pred}) + (1 - Y) * log(1 - Y_{pred})]$$

- Cosine similarity loss: Cosine similarity is a measure of similarity between two non-zero vectors in an n-dimensional space. In cosine similarity loss, the similarity of two input vectors is computed using cosine similarity and the result is compared to the gold similarity score.

$$loss = \frac{Y.Y_{pred}}{|Y|.|Y_{pred}|}$$

- Contrastive loss: Contrastive loss, is a loss function used in deep learning for tasks involving similarity learning, such as recommendation systems. It is designed to encourage a neural network to learn embeddings (vector representations) in such a way that similar items or examples are embedded closer together in the feature space while dissimilar ones are farther apart.

$$loss_{i,j} = -\log \frac{exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N}[i \neq k]exp(sim(z_i, z_k)/\tau)}$$

2.2.6 Embeddings

In natural language processing (NLP), an "embedding" refers to a numerical representation of words, phrases, or documents in a continuous vector space. These embeddings are designed to capture semantic relationships between words and enable machine learning models to work with text data in a more meaningful way. Some common approaches to creating word embeddings are: Word2Vec [6], GloVe [7], and BERT [8].

2.2.7 Softmax

Softmax is a mathematical function that converts a vector of real numbers into a probability distribution. It can be applied on the output layer of a neural network for multi-class classification tasks. The softmax function takes as input a vector of arbitrary real numbers (logits) and transforms them into a probability distribution over multiple classes.

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

2.2.8 Transformers

Transformers [9] are a class of deep learning model architectures that have revolutionized natural language processing (NLP) and other sequence-based machine learning tasks. Introduced in 2017, they rely on the self-attention mechanism, allowing them to efficiently capture context and relationships between elements in sequences, making them particularly effective for NLP applications. Transformers consist of multiple layers of multi-head self-attention, feedforward neural networks, and position encodings. They have led to remarkable advancements in NLP and have been adapted for various domains, setting new standards in tasks ranging from text classification and machine translation to sentiment analysis and question answering. Popular models like BERT, BART, GPT-3, and RoBERTa are based on transformer architectures.

Transformers are often used in a paired encoder-decoder architecture for sequence-to-sequence tasks, such as machine translation. The encoder processes the input sequence, while the decoder generates the output sequence.

2.2.9 BERT

BERT [8], or "Bidirectional Encoder Representations from Transformers," is a natural language processing model developed by Google AI based on transformers. BERT excels in representing the meaning of words and phrases in context, thanks to its bidirectional architecture and transformer-based design BERT only uses the encoder part of transformers by stacking multiple encoders. This architecture enables BERT to generate embedding of words, and can also be used for classification tasks. BERT is pre-trained on two tasks, next sentence prediction and masked language modeling (predicting masked words in a sentence).

2.2.10 RoBERTa

RoBERTa [10], or "A Robustly Optimized BERT Pretraining Approach," is a state-of-the-art natural language processing model introduced by Facebook AI in 2019. It is built upon the BERT (Bidirectional Encoder Representations from Transformers) architecture but incorporates several key improvements and optimizations, making it one of the most powerful language understanding models. RoBERTa employs larger datasets for pretraining, more training steps, dynamic masking of text, and other enhancements, allowing it to outperform its predecessors on a wide range of NLP tasks.

2.2.11 SBERT

Sentence-BERT [11] or SBERT is a framework developed for creating sentence embeddings, which are vector representations of sentences. SBERT is designed to capture the semantic meaning of sentences and their contextual relationships in a continuous

vector space. It achieves this by training on a large amount of sentence pairs and applying siamese network and triplet loss. SBERT has proven highly effective in various NLP tasks, such as information retrieval, semantic search, and document clustering, where understanding sentence-level semantic similarity and dissimilarity is crucial.

2.2.12 Clustering

Clustering is a machine learning technique used in unsupervised learning, where the goal is to group similar data points or objects together based on their inherent similarities or patterns. The objective of clustering is to identify natural structures or clusters within a dataset, allowing data to be organized into subsets with similar characteristics. Common clustering algorithms include K-means [12], hierarchical clustering [13], and DBSCAN [14]. Clustering is used in various applications, such as data analysis, image segmentation, customer segmentation, and recommendation systems, to discover patterns, explore data, and make it more manageable for further analysis or decision-making.

In the context of NLP, we use sentence embeddings as inputs for different clustering algorithms. Ideally the embeddings of similar sentences are more similar to each other, enabling clustering algorithms to group them with each other.

Agglomerative Clustering

Agglomerative clustering is a hierarchical clustering method in machine learning and data analysis that starts with each data point as its own cluster and progressively merges the closest clusters, forming larger clusters until all data points belong to a single cluster or a predefined stopping criterion is met. This bottom-up approach creates a hierarchical structure, allowing for different levels of granularity in cluster formation.

Rand Index Score

The Rand index score [15] is a measure used in data analysis and clustering to assess the similarity between two different clusterings or to evaluate the quality of a

clustering solution in comparison to a known reference (ground truth). It quantifies the agreement between the two clusterings by calculating the number of data point pairs that are correctly clustered together or correctly placed in different clusters, relative to all possible pairs. The score ranges from 0 to 1, where a higher score signifies a higher level of agreement, with 1 indicating perfect agreement between the clusterings. It is a valuable tool for cluster evaluation and validation, providing insight into the quality of clustering results. The Rand Index is computed using the formula:

$$RI = \frac{CorrectSimilarPairs + CorrectDissimilarPairs}{TotalPossiblePairs}$$

2.3 Evalutation Metrics

2.3.1 ROUGE Score

ROUGE [5] (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics and algorithms used for the automatic evaluation of the quality of machine-generated text, such as summaries, machine translation outputs, and other natural language processing tasks. ROUGE measures the similarity between a reference (human-generated) text and the machine-generated text by comparing n-grams (contiguous sequences of n words) and other linguistic units. The primary focus of ROUGE is on recall, which means it assesses how much of the reference text is captured by the generated text. Some common ROUGE metrics include ROUGE-N (measuring overlap in n-grams) and ROUGE-L (computing the longest common subsequence). ROUGE scores are often used in research and development to assess the quality and performance of various natural language generation tasks.

2.3.2 BLEU Score

BLEU [16] (Bilingual Evaluation Understudy) is a metric used to evaluate the quality of machine-generated translations by comparing them to one or more reference

translations. It was proposed as a reference-based evaluation metric to address the limitations of simpler metrics like precision and recall. The BLEU score is calculated based on the precision of n-grams (contiguous sequences of n words) in the candidate translation compared to the reference translations. The precision is computed for different n-gram orders (unigrams, bigrams, trigrams, etc.), and the scores are combined using a weighted average. The resulting BLEU score ranges from 0 to 1, with higher scores indicating better agreement between the candidate and reference translations.

2.3.3 Transformer-based Evaluation Metrics

Transformer based evaluation metrics aim to overcome the shortcomings of traditional metrics such as ROUGE and BLEU score. Traditional metrics evaluate text based on n-grams, not taking into account the semantics of words with respect to the context. Additionally, they have no way of evaluating whether the generated texts are fluent, accurate, or meaningful. Transformer based metrics solve these issues by making use of pre-trained language models such as BERT and BART [17].

BLEURT [18] is a metric for evaluation of natural language generation tasks based on BERT. It takes a pair of candidate and reference as inputs, and returns a similarity score indicating the extent of candidate's fluency and similarity to the reference. Similarly, BARTScore [19] is an evaluation based on BART, that evaluates the quality and similarity of a generated text compared to the reference.

2.4 Argument Summarization

2.4.1 Argument Mining

Prior to the introduction of the key point analysis (KPA) task, the field of argument summarization was rather underdeveloped, both in terms of available datasets and techniques. However, several research focused on related experiments, such as clustering or extraction of arguments. Misra et al. [20] aimed to extract different aspects of arguments, called argument facets, similar to key points. The proposed framework

first extracted the sentences that include arguments, and further ranked extracted sentences by their similarity to each other, with similar arguments representing an argument facet. Egan et al. [21] proposed a multi-step argument summarization, where first "points", a verb and its syntactic arguments, were extracted and the final summary was generated by connecting points. Ajjour et al. [22] focused on frame identification using clustering, where a frame refers to arguments that cover the same aspect. The method first clusters the arguments into topics then removes topic-specific features from the arguments, and lastly clusters the "topic-free" arguments into frames. Reimers et al. [23] similarly focuses on argument clustering and classification, however unlike the previous approach they use BERT [24] and ELMo [25] instead of TF-IDF and LSA [26]. The authors experimented with different classification and clustering methods in both supervised and unsupervised settings.

2.4.2 Key Point Analysis

According to the organizers of the KPA task, a total of 17 and 5 models were submitted for the matching and generation track, respectively. The top two performing models based on the evaluation of the organizers were BarHaim [27] and Alshomary [28].

BarHaim, which was developed by the curator of the dataset used for the task, uses an extractive approach. The method scores all arguments in the input with respect to their quality, and selects all the arguments scored above a threshold as candidates. Then the rest of the arguments in the input are matched to every candidate using a key point matching model, where an argument-key point match is made with the highest scoring candidate if its matching score is above a certain threshold. Lastly, candidates are sorted by the number of arguments they cover and the top k candidates are selected as summary, if their similarity score with previously selected candidates is below a certain threshold.

Alshomary uses a variation of page rank [29] for extractive summarization. After

filtering out the arguments starting with pronouns and low quality, an undirected graph is constructed where the nodes are the remaining arguments. Next, the importance score between every two nodes is calculated using a key point matching method, and nodes with a high confidence score are connected. The nodes with the highest importance score are selected as the summary, as long as they don't have a high similarity score with previously selected arguments.

Sultan et al. [30] experimented with the approaches proposed by BarHaim [27] and Alshomary [28] in the legal domain, in addition to experimenting with their approach. Authors first cluster the arguments and input each cluster into different extractive and abstractive summarization methods such as LexRank [31], LSA [26], Luhn [32], BART [17], PEGASUS [33], and Legal-PEGASUS. The authors evaluate the effectiveness of each method by manually comparing the output of each method.

Li et al. [34] is the most similar to ours, as it uses a clustering based approach and introduces new evaluation metrics. After clustering all the input arguments, the authors input all arguments belonging to a cluster into a language model, prompting it for summarization. The proposed method uses BERTopic [35] for clustering, specifically it uses a pre-trained language model [36] for getting contextualized embeddings, performs dimensionality reduction using UMAP [37], and applies HDBSCAN [38] to cluster embeddings. In the second step, it uses a fine-tuned language model [39] for generating the reference key points given the arguments in each cluster, to generate the final summary. The collection of generated key points for all clusters is presented as the final summary. For evaluation metric, the authors propose a precision and recall based approach, called soft precision and soft recall. However as the true labels for generated key points and reference key points are not known (i.e. which reference key point belongs to which generated key point), the authors propose using a similarity score instead. For soft precision the reference key point with the highest similarity score for each generated key point is found. For soft recall the generated key point with the highest similarity score for each reference key point is found. These

similarity scores are averaged and are presented as the soft precision and recall. In order to calculate the similarity score between generated and reference key points, BLEURT [18] and BARTScore [40] is used. Finally a soft F1 score is calculated from the soft precision and recall.

2.5 Conclusion

This chapter covered the basic concepts and background needed for our work. We first explored basics of natural language processing (NLP), deep learning, and recent popular models used in NLP. Next we discussed the evaluation metrics and methods used for evaluating generated text. Lastly, we focused on reviewing the related works in our field, argument mining and summarization. We discussed the idea of earlier approaches in argument mining and the recent key point analysis task, which our approach is based upon. We also included a survey in unsupervised summarization that surveys the recent trends in the field A.

Chapter 3

Methodology

3.1 Overview

This chapter first explains the structure of the dataset used in key point analysis (KPA), the limitations of previous approaches to KPA, and how the proposed approaches addresses them. Next, it focuses on the proposed approach that consists of a clustering and a selection step.

This chapter outlines our proposed framework and approach to Key Point Generation (KPG). The input corpus comprises a collection of arguments centered around specific topics, such as "Should routine child vaccinations be mandatory?". The output, also called generated key points or the summary are a collection bullet point style sentences that summarize the input arguments. Our approach, alongside the previous approaches we compare our work to, are extractive. Therefore the generated key points are extracted arguments from the input.

To create summaries, our model starts by clustering the input arguments and selecting the most suitable sentence within each cluster as a candidate. Figure 3.1 provides an overview of this approach.

3.2 Dataset

The dataset proposed for the task includes information in the form of "key point, argument, label, stance, topic" data point, where the label determines if the argument

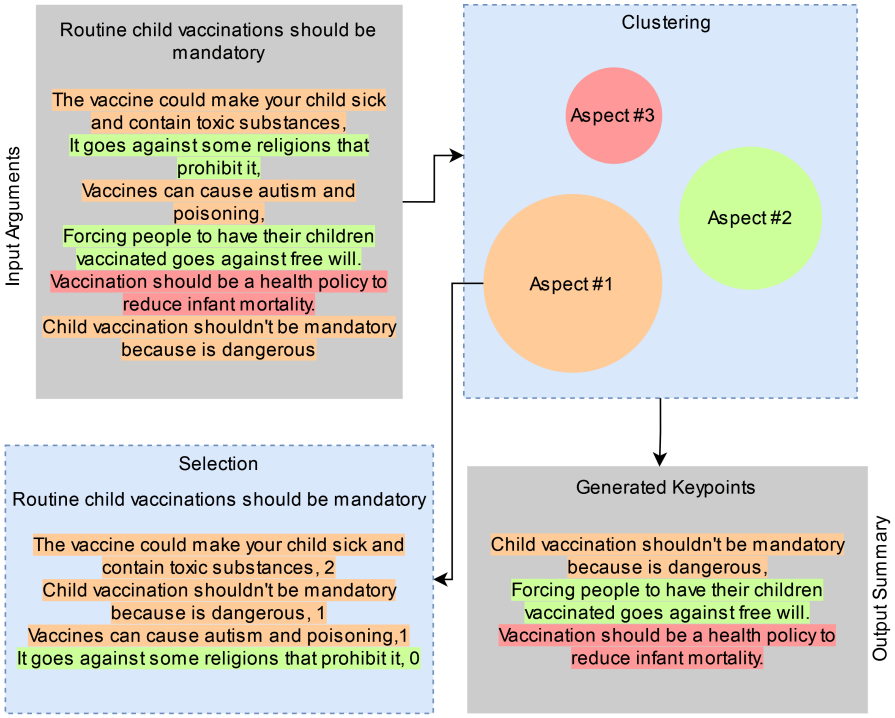

Figure 3.1: Visual Depiction of the Proposed KPG Method. The color of arguments represents their key point or aspect. First the clustering step groups similar arguments. Next, the selection step chooses the argument with the highest score within each cluster as the cluster representative. The final summary aggregates the representatives from each cluster

supports the key point. The dataset includes information about the topic and stance of an argument as well. Topic can be any arbitrary subject, but the stance value is either zero or one, representing "against" or "pro". For example, on the topic of "Homeschooling should be banned", the key point "Parents are not qualified as teachers." is pro the topic, and the key point "Mainstream schools have a lot of violence/bullying." is an example of a key point against the topic. Arguments are the inputs for our models, they are collected from users data online and are paired with key points. The label determines if an argument-key point are matching or not. A pair is matched if both argument and key point cover the same aspect. For example, for the key point "Parents are not qualified as teachers.", and the argument "Children learn more how to socialize in schools.", the label is set to 0 since the argument and the key point cover different aspects (i.e. parent not qualified and schools enable socializing).

The key points in the dataset are generated by an expert on the topic. The number of key points is different for each topic, but is between 3 to 8 key points per stance per topic. The arguments are collect from users on the internet. The authors use human judges to label and stance values to key point-argument pairs.

The ArgKP dataset has 24 topics in the training set and 4 in the validation set, containing 24,000 data points in total. Additionally, the test set includes 3 new topics with 3,400 data points. The topics in the test set are as follows:

- Topic 1: Routine child vaccinations should be mandatory
- Topic 2: Social media platforms should be regulated by the government
- Topic 3: The USA is a good country to live in

We implemented two modifications to the task, first we assume the stance information is not initially available and second we restructure the data.

- For the first modification, we combined data points from both stances to re-

semble real-world data where stance information is not readily available. Consequently, the dataset is now categorized solely by topics.

- For the second modification, we have restructured the dataset to better suit our needs. We transitioned from the format of {key point, argument, label, stance, topic} data point to a format where each topic is represented as a dictionary. In this new structure, the dictionary's keys correspond to key points, and their respective arguments are the associated values. This adjustment was made to streamline the process of assessing coverage and identifying arguments related to the same key point. Below is an example of the new structure.

- Topic 1: Routine child vaccinations should be mandatory
  - Key point 1: Routine child vaccinations, or their side effects, are dangerous
    - Argument 1: Routine child vaccinations should not be mandatory because children may not bear the side effects of it.
    - Argument 2: Child vaccination should not be mandatory as there are often side effects from them
    - (. . .)
  - Key point 2: Child vaccination saves lives
    - Argument 3: Child vaccination saves lives and keeps children from suffering from preventable illness. This can help a person into adulthood by not having complications from diseases like chicken pox, mumps, polio and more.
    - Argument 4: The use of child vaccines saves lives
    - (. . .)
  - Key point3: (. . .)
- Topic 2: (. . .)

It is also important to note that we propose a modification to the key point analysis task. The original key point analysis track was designed with distinguished stances, we propose a modification to the task where the stance information is not known. Having such stance information would be advantageous as it could assist in initially categorizing arguments based on their stance, making it easier to distinguish between lexically similar key points with opposing viewpoints. We proposed this change to make the methods more applicable to real data. In a real world scenario, arguments about a certain topic could be found easily, e.g. by using popular hashtags in twitter. However, the stance of each argument cannot be accurately determined using automatic methods.

3.3 Limitations of Previous Methods

Previous approaches, specifically BarHaim and Alshomary (as explained in 2.4.2), exhibit weaknesses that hinder the quality of their output. This is partly due to their reliance on hyperparameter fine-tuning and usage of biased approaches. Additionally, both of these methods assume that the stance of each input is known, which is often not the case in real-world scenarios where additional metadata about inputs are generally unavailable. Specifically, both methods employ an initial filtering step in their process. While this step is intended to remove low-quality arguments, it also diminishes the pool of available inputs, which is crucial for an extractive summarization algorithm. In addition, to make the final output more akin to a summary format, both methods remove sentences that start with pronouns, further reducing the available inputs. Furthermore, both methods employ a fixed similarity threshold for detecting duplicate arguments in the generated summary, which applies uniformly to all topics when eliminating similar candidates. This approach is problematic because the actual similarity threshold for one topic is likely to differ from that of another topic, given that key points in some topics are more semantically related than in others. In addition to their shared limitations, both methods have their own unique constraints.

BarHaim heavily relies on two thresholds for matching arguments to candidates and for eliminating similar candidates. These thresholds are tailored to a specific dataset, limiting the method's generalizability to other datasets. Moreover, BarHaim only selects short arguments for inclusion in a summary. This often yields only a few arguments from an input of around two hundred and produces no summary for inputs of a hundred or fewer. The other method, Alshomary, determines the best values for hyperparameters (e.g. "d," "qual," "match", standing for minimum distance, minimum quality, and minimum matching score) for a specific dataset and also employs a threshold similar to BarHaim for identifying similar candidates.

Beyond the technical constraints imposed by their methodologies, the quality of outputs produced by BarHaim and Alshomary are negatively impacted by their strategy for summary. The generated summary of these models lack comprehensive coverage of the reference key points. Both methods tend to extract multiple arguments for popular aspects or key points while overlooking key points with fewer associated arguments. This phenomenon happens because there often is an imbalance between the number of arguments discussing different aspects or key points of a topic. For example, on the topic of "Routine child vaccinations should be mandatory", there is a total of nine reference or ground truth key points, in the ArgKP dataset, where the most popular key point, has fifty arguments discussing it, whereas the least popular key point has only four. As a result BarHaim is more likely to extract from popular topics, as candidates from popular topics are more likely to get a higher match score. Also in Alshomary's work, nodes representing arguments from popular key points get higher importance scores, as there are more edges connecting them. This falls short of the desired summarization objective, which is to encompass all key points, not just the most popular ones.

3.4 Addressing the Limitations

Our approach effectively addresses the aforementioned limitations by leveraging clustering and matching techniques.

First, our method operates without the need for fine-tuning hyperparameters specific to a dataset, making it readily applicable to diverse datasets without requiring any modifications.

Second, our method doesn't depend on prior knowledge of stance information, allowing it to be used with real-world data where such information is often unavailable.

Third, we refrain from employing input quality filtering or the removal of arguments commencing with pronouns. This means we retain all arguments that could potentially be valuable, eliminating the necessity for an external model to assess argument quality.

Fourth, the clustering step groups similar candidates into the same cluster, and since only one argument from each cluster is chosen, there is no need for duplicate removal, thereby obviating the similar candidate elimination step and its associated threshold.

Lastly, by using clustering, our method excels in providing wider coverage of key points, prioritizing but not favoring popular ones. This generates a summary with a higher coverage of reference key points and a lower redundancy between generated arguments, as demonstrated by our experiments, encompassing less popular key points as well.

3.5 The Proposed Method

The proposed method can be divided into two main components: clustering and argument selection. The clustering step, aims to cluster arguments that discuss the same aspect. The argument selection, selects a candidate argument in each cluster that best represents the entire cluster. The collection of candidates from all clusters

is the summary (i.e, generated key points). In the subsequent sections, we explore various options and configurations for our proposed method.

3.5.1 Clustering

Prior methods for Key Point Generation (KPG), such as the works by BarHaim and Alshomary, often generate summaries by giving preference to popular aspects. This results in summaries that tend to overemphasize certain key points while neglecting others. Our approach aims to provide equal representation for all key points, prioritizing them without showing favoritism to popular ones. This approach results in summaries that better cover the reference key points and exhibit reduced redundancy among generated arguments. To achieve this objective, we group similar arguments, or sentences, that pertain to the same key points by employing Sentence-BERT (SBERT) [11]. Specifically, we first get the sentence **embeddings** for input arguments using SBERT and further **cluster** them using Agglomerative Clustering [13] as seen in Algorithm 1.

In order to evaluate the effectiveness of our approaches in **embedding** and **cluster**, we use Rand index score [15]. Rand index score is a metric for clusters that evaluates the similarity between the predicted clusters and the actual clusters. A score of 1 represents perfect accuracy.

Algorithm 1 Overview of the Clustering Approach

Input: Arguments ARG,
Embdding Model (SBERT): $embed$,
Clustering Technique: $cluster$,
Output: Clusters C

for arg *in* ARG **do**
    $embeddings[arg] \leftarrow embed(arg)$
end for
$C \leftarrow cluster(embeddings)$
$C.sort()$
return C

Embedding

We observed that even the best-performing pre-trained SBERT model (all-mpnet-base-v2) for generating embeddings led to low Rand index scores, indicating that the clustered embeddings did not accurately represent the actual clusters, see Table 3.2. To address this, we improved the embedding generation model (SBERT) by fine-tuning it on the ArgKP dataset using both contrastive loss (see Section 2.2.5), following Alshomary, and Cosine Similarity loss. We used argument-key point pairs from the ArgKP train set as input pairs and the matching labels as output for fine-tuning the SBERT model.

Fine tuning	Topic 1	Topic 2	Topic 3
Before FT	0.184	0.190	0.152
FT Cosine Similarity Loss	0.434	0.453	0.388
FT Contrastive Loss	0.378	0.446	0.317

Table 3.1: Rand scores using Agglomerative Clustering before and after fine-tuning with two different methods. Using "all-mpnet-base-v2" SBERT base models and distance threshold of 1.5.

Language Model	Topic 1	Topic 2	Topic 3
"MiniLM w/ distance threshold"	0.103	0.144	0.156
"mpnet w/ distance threshold"	0.184	0.190	0.152
"MiniLM w/ cluster number"	0.105	0.144	0.168
"mpnet w/ cluster number"	0.186	0.185	0.188

Table 3.2: Rand scores using Agglomerative Clustering with two different settings, with number of clusters equal to reference key points and 1.5 distance threshold. Rand scores are averaged on the three test set topics. MiniLM represents "all-MiniLM-L6-v2" and mpnet represents "all-mpnet-base-v2" SBERT base models. MiniLM model is faster and smaller version of mpnet.

We observed that fine-tuning the language model improves the Rand index score in the clustering step by more than a 100%, and that the model fine-tuned using Cosine Similarity outperformed the one fine-tuned using Contrastive loss, Table 3.1.

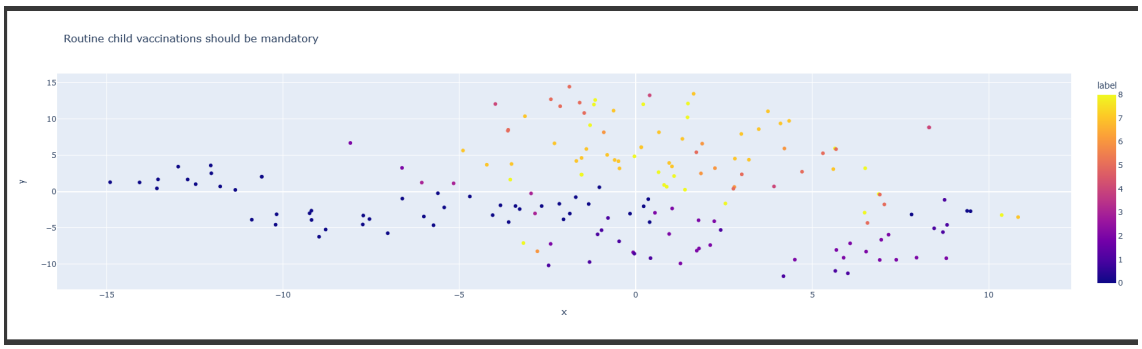

Figure 3.2: Embedding plots before fine-tuning the embedding model on topic 1

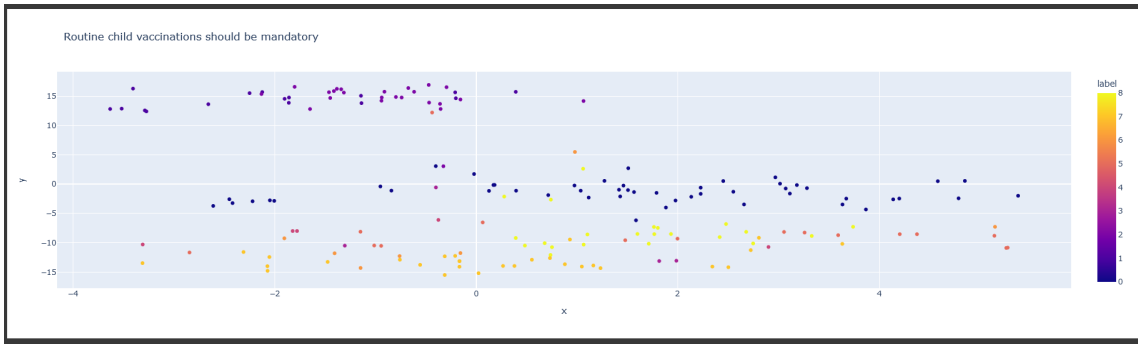

Figure 3.3: Embedding plots after fine-tuning the embedding model on topic 1

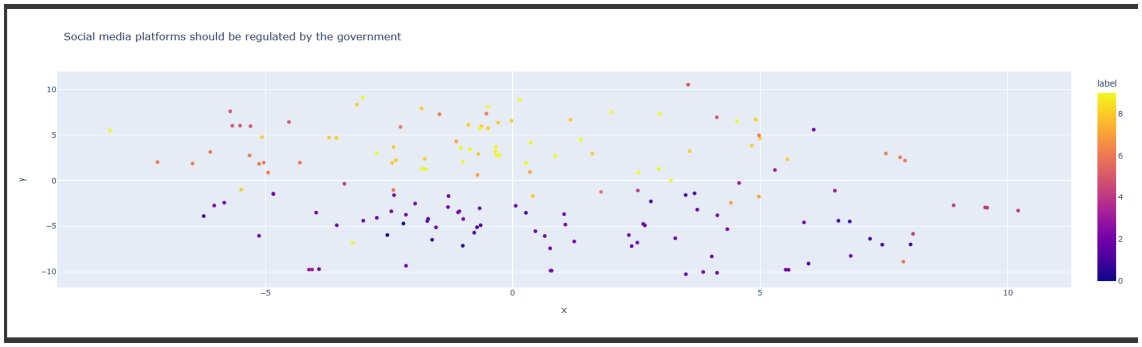

Figure 3.4: Embedding plots before fine-tuning the embedding model on topic 2

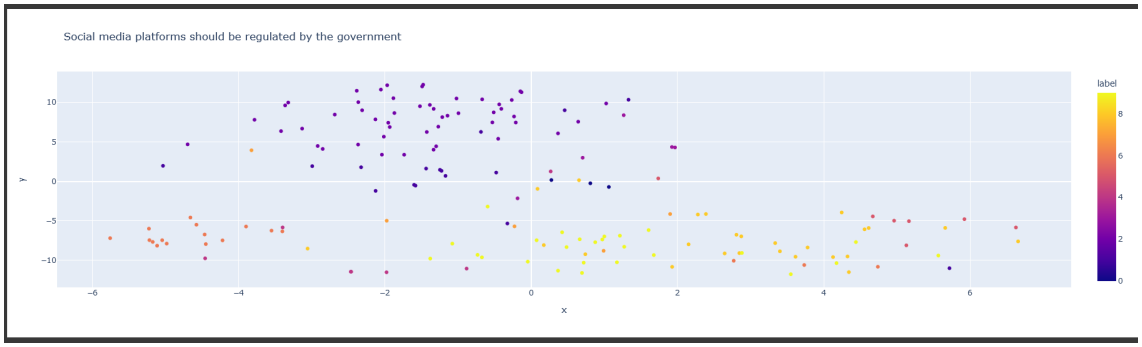

Figure 3.5: Embedding plots after fine-tuning the embedding model on topic 2

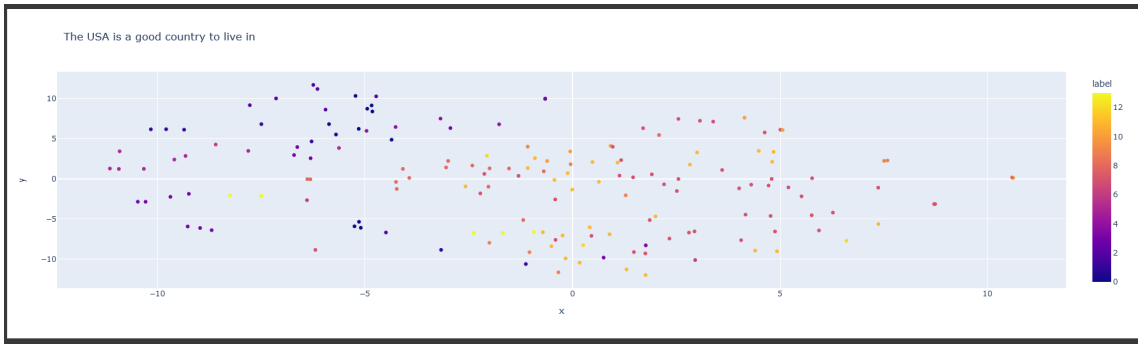

Figure 3.6: Embedding plots before fine-tuning the embedding model on topic 3

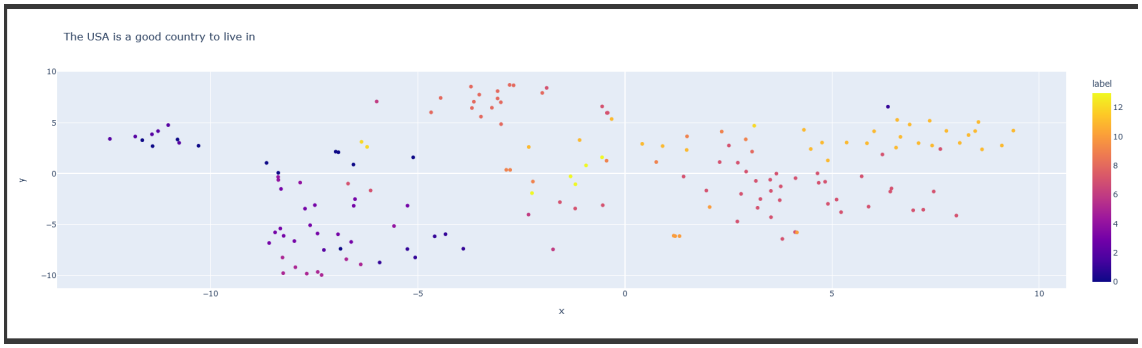

Figure 3.7: Embedding plots after fine-tuning the embedding model on topic 3

To further demonstrate the effectiveness of fine-tuning, we depict TSNE plots of arguments for the three topics in ArgKP test set before and after fine-tuning SBERT in Figures 3.2 through 3.7. Figures 3.2 and 3.3 show the embedding plots of arguments on the topic of "Routine child vaccinations should be mandatory" before and after fine-tuning. As seen in the plots, the embeddings of arguments related to the same aspect are grouped next to each other after fine-tuning, and are less spread over the distribution. The same phenomenon happens in Figures 3.4 and 3.5, and Figures 3.6 and 3.7 that showcase arguments on the topic of "Social media platforms should be regulated by the government" and "The USA is a good country to live in" respectively.

Cluster

In the second clustering step, we conducted experiments using various clustering algorithms on both the training and test sets. The clustering algorithms employed in our experimentation included Agglomerative Clustering, K-Means [12], Affinity Propaga-

tion [41], Mean Shift [42], Spectral Clustering [43], DBSCAN [14], BIRCH [44], and Bisecting K-Means [12]. Based on the Rand Index scores obtained from each method as shown by Table 3.3, Agglomerative Clustering, BIRCH, and Bisecting K-Means exhibited the best overall performance, followed by K-Means, Affinity Propagation, and Spectral Clustering. However, DBSCAN and Mean Shift yielded unsatisfactory results. It's worth noting that these clustering methods have distinct hyperparameters. For instance, while some methods like Spectral Clustering and K Means require specifying the number of clusters as input, others like DBSCAN depend on a distance hyperparameter. Moreover, methods such as BIRCH and Agglomerative Clustering can accept either of these parameters as input. We used the default hyperparameters for each clustering method to ensure a fair comparison, and we refrained from using external information, such as the number of clusters, or fine-tuning hyperparameters to achieve the best scores.

Rand Index Score	Training set	Test set
Agglomerative Clustering	0.4736	0.4300
K Means	0.4510	0.3990
Affinity Propagation	0.4448	0.4072
Spectral Clustering	0.4183	0.4162
DBSCAN	0.2416	0.1042
BIRCH	0.5321	0.4284
Bisecting K Means	0.4731	0.4208

Table 3.3: Rand index score using different clustering methods. The scores averaged across all topics for both the training (28 topics) and test (3 topics) sets.

We selected Agglomerative Clustering due to its superior Rand index scores, consistent with the findings of Reimers et al. [23], who also identified it as the optimal clustering algorithm for argument clustering and classification. An additional advantage of Agglomerative Clustering is that it does not require specifying the number of clusters as a hyperparameter, which is beneficial given that this information is often

unknown during inference. However, our approach remains adaptable to algorithms that might necessitate an estimated number of clusters.

Agglomerative Clustering

Agglomerative clustering is a hierarchical clustering method. The process starts by considering each data point as a single cluster. Then, it iteratively merges the closest clusters until a stopping criterion is met, resulting in a hierarchy of clusters.

At each step, the algorithm identifies the two clusters that are closest together and merges them into a single cluster. The distance between clusters is typically defined using a distance metric such as Euclidean distance or Manhattan distance.

Agglomerative clustering proceeds until all data points are in a single cluster, until a predetermined number of clusters is reached, or until the distance threshold is reached. The distance threshold is a parameter used to determine when to stop merging clusters during the clustering process (i.e. threshold at or above which clusters will not be merged). In our implantation, we use a distance threshold of 1.5. The overview of the clustering approach can be seen at 2.

Algorithm 2 Agglomerative Clustering

Input: Emdeddings $emdb$, distance metric $dist(\cdot, \cdot)$, Distance Threshold T
Output: Clusters C

$C \leftarrow \{\{embd_1\}, \{embd_2\}, ..., \{embd_n\}\}$
while $|C| > 1$ **do**
    Find the two closest clusters C_i and C_j:
    $C_i, C_j \leftarrow \arg\min_{C_a, C_b \in C} d(C_a, C_b)$
    Merge C_i and C_j: $C \leftarrow (C \setminus \{C_i, C_j\}) \cup \{C_i \cup C_j\}$
    if $\min_{C_a, C_b \in C, C_a \neq C_b} d(C_a, C_b) > T$ **then**
        break
    end if
end while
Return *Clusters*

BERTopic

Furthermore, we conducted experiments involving neural topic modeling as used by Li et al. [34]. This method employs BERTopic for clustering, where a pre-trained language model (SBERT) is utilized to generate contextualized embeddings. Subsequently, dimensionality reduction using UMAP is applied to these embeddings. Finally, HDBSCAN is employed to cluster the resulting embeddings. Table 3.4 presents the Rand index score achieved using this approach. Although the outcomes demonstrate enhancement compared to the baseline language model, it is evident that the suggested method falls considerably short of the performance achieved by our approach utilizing a fine-tuned language model.

Rand Index Score	Topic 1	Topic 2	Topic 3
BERTopic Modelling	0.2400	0.2518	0.1296

Table 3.4: Rand index scores using BERTopic Modelling approach on three topics for test set

3.5.2 Argument Selection

In the argument selection phase, the method's goal is to choose a single argument that best represents the entire cluster. Given that the clustering step groups related arguments, having just one argument from each cluster is adequate to represent each cluster. To determine the best argument for each cluster, we adopt a multi-faceted approach. Initially, we sort the clusters by their size, giving precedence to larger and more substantial clusters. Subsequently, we experiment with eleven different scoring functions, influenced by BarHaim's work. These methods aim to select the most suitable cluster representative, taking into account argument coverage, argument quality scores, or a combination of both. Algorithm 3 depicts an overview of the process. Table 3.5 depicts an overview of models and techniques used in each function.

Algorithm 3 Argument Selection

Input: $cluster\{c_1\}, cluster\{c_2\}, \ldots, cluster\{c_n\}$
Output: $Summary\{c_1\}, Summary\{c_2\}, \ldots, Summary\{c_n\}$

for $cluster\ c_i = c1, c2, \ldots, cN$ **do**
  for $argument\ arg_i = arg1, arg2, \ldots, argN\ in\ c_i$ **do**
    $Score\{c_i\}\{arg_i\} \leftarrow scoring\_function(arg_i,\ c_i)$
  end for
  $Summary(\{c_i\}) \leftarrow Argmax(Score\{c_i\})$
end for
return $output \leftarrow \{Summary\{c_1\}, Summary\{c_2\}, \ldots, Summary\{c_n\}\}$

Scoring Function	KP Match.	Arg. Qual.	Len. (Static)	Len. (Dynamic)	Splitting
1	■				
2	■	■			
3		■			
4	■				
5		■	■		
6	■				■
7	■				■
8	■		■		■
9	■		■		■
10	■	■			■
11	■			■	■

Table 3.5: Scoring functions categorized by the model and techniques they employ for scoring arguments. The models and techniques are key point matching, argument quality, length (static and dynamic), and splitting.

Key Point Matching Model

All scoring function use the key point matching model as part of their scoring process, fine-tuned on the ArgKP dataset. The matching model takes a a pair of sentences as inputs, and outputs a binary label. The label represents the model prediction on whether the first sentence covers the same aspect as the second sentence (i.e. first sentence entails the second sentence). This enables the argument selection step to find the argument in each cluster that entails the most sentences in that cluster. The

argument that entails the most number of arguments in a cluster is considered as the representative of that cluster. We call this the argument coverage.

The key point matching model is fine-tuned on the ArgKP dataset, where an argument-key point pair are given as inputs and the matching label is given as the label. We use BERT as the key point matching model. During inference, sentence pairs are considered matching if the predicted label is 1. A match between two arguments signifies that one argument entails the other. We format each input to the BERT model as "[CLS] argument [SEP] key point [SEP]", and [logit 1, logit 2], where [1, 0] represents class zero or non-matching and [0, 1] represents matching argument-key point pairs, for the output. A pair is considered matching if the second logit is greater than the first one.

Argument Quality Model

Some of the scoring functions use an argument quality model [45]. The argument quality model, takes an argument as the input, and returns a number between 0 and 1 to determine the quality of an argument. High quality arguments get a score close to 1.

Scoring Functions (SF)

1. In the first SF, we **only utilize the key point matching** model. Each argument in a cluster is compared with every other argument in that cluster, and the key point matching model predicts whether a pair matches or not. After this process is applied to all the arguments in a cluster, we **rank the arguments based on the number of matches** they have, selecting the argument with the **highest number of matches**. If multiple arguments have the same highest number of matches, we opt for the shortest argument to ensure a concise and comprehensive summary.

2. In the second SF, we follow a similar approach and rank arguments by the num-

ber of matches, however we use the **average match number as a threshold to filter** out arguments with matches **below threshold**. We then compute the argument quality score for the top arguments in each cluster using the argument quality model and select the highest-scoring argument. This in theory should enable us to select high-quality arguments with coverage.

3. The third SF **solely relies on** the **argument quality score** in each cluster, choosing the argument with the highest score. This was done as an experiment to measure the effectiveness of our matching model in selecting high-coverage summaries.

4. In the fourth SF, we opt for the argument with the **highest coverage**, akin to the first SF, but we use a **strict threshold** for allowing matches. We only consider a pair as a match if the second logit is greater than the first, and it is above a threshold (0.9). This increases the sensitivity of matching model, only matching arguments that are very similar to each other.

5. The fifth SF selects the **shortest argument** in terms of length, provided its quality score is above a specific threshold (0.8). This SF aimed to extract short high quality summaries, without focusing on coverage.

6. The following methods **focus on splitting multi-sentence arguments into separate, shorter sentences**, as the previous approaches often produced multi-sentence outputs. However, as the sentence splitting step could be applied both before and after clustering, we experimented with both. Method six performs this split after the clustering step, while SF seven does it before the clustering step. Method six outperforms SF seven, proving that using the original longer sentences proves more advantageous in the clustering step.

7. Methods eight, nine, ten, and eleven are designed experiment with generating shorter outputs, while also trying out affects of quality and coverage as criteria

for selection. These methods are a continuation of SF six with sentence splitting after clustering steps.

Method eight segments sentences (breaking multi-sentences to single sentences), selecting sentences with a **maximum of 70 words**. This enforces a strict length limit, selecting only short sentences as summary.

Method nine follows a similar word limit but **splits sentences into phrases** using using Spacy library. Since phrases are often shorter than sentence, we believed this might increase the number phrases that are within the length limit.

Method ten segments sentences and adheres to the maximum word limit, but selects the **highest quality sentences** that are **not similar to previously chosen sentences**, using cosine similarity. This SF is inspired by BarHaim, and aims to choose high-quality sentences while maintaining diversity, i.e., high coverage.

Upon evaluating the output and coverage of SF ten, it becomes evident that neither the quality nor the coverage of the model is enhanced. Method eleven **prioritizes shorter sentences without relying on a static length**. The function scores arguments in each cluster relative to the number of matches and penalizes lengthy sentences. We use

$$Score = \frac{matches^i}{\#ofwords}$$

where i determines the importance of matches.

Evaluation of Scoring Functions

In order to choose the best SF, we have to evaluate the output generated by each one. We use the coverage score as this evaluation metric. The coverage score represents the percentage of reference key points that are covered by a summary. A score of one means that all the reference key points on a topic are covered by the summary.

Additionally, the output generated by the methods generate different number of key points in the summary. As a result, the summaries that have more generated key points, have a higher chance of getting a higher coverage score. To this end, we also report the "limited" version of output summaries, where the number of generated key points is limited to the number of reference key points for each topic. We use the results of limited output for comparison and selection of the best methods.

Table 3.6 shows the coverage score of different methods for each topic in the test set. Based on the coverage results, we selected SF 11 and SF 6 (referred to as V11 and V6) as the best performing methods. Although V6 offers better coverage, V11 offers shorter summaries. The reason for this is that the selection criteria for V6 only considers coverage, whereas V11 considers both coverage and length. We did not select V1 as its coverage score is lower than V6 and it does not consider length unlike V11. We refer to V6 as **Selection with Matching Model** (**SMM**), and SF V11 as **Selection with Scoring Function** (**SSF**).

Coverage Scores	Topic 1	Topic 2	Topic 3	**Averaged**
V1	0.7778	0.5000	0.5714	0.6164
V2	0.5556	0.4000	0.6429	0.5328
V3	0.4444	0.4000	0.5714	0.4720
V4	0.6667	0.4000	0.5714	0.5460
V5	0.4444	0.4000	0.5714	0.4720
V6	0.8889	0.5000	0.6429	**0.6772**
V7	0.6667	0.6000	0.5000	0.5889
V8	0.4444	0.4000	0.6429	0.4958
V9	0.5556	0.3000	0.6429	0.4995
V10	0.5556	0.4000	0.5714	0.5090
V11	0.7778	0.4000	0.6429	**0.6069**

Table 3.6: The coverage score of different scoring functions on three test topics. The averaged column reports the average score for all three topics.

3.6 Conclusion

In this section we explained the structure of dataset used in the KPA task, shortcomings of previous approaches and our solution to them. We also introduced our approach, which consists of two steps, clustering and argument selection. In our approach, the clustering step groups similar arguments in a cluster. The argument selection step extracts one sentence from each cluster as the representative of the cluster. The summary (or generated key points) is the collection of all representative.

For our approach, we first fine-tuned SBERT which is used for generating embeddings. The embeddings are input to our clustering step. We experimented with various clustering techniques and chose Agglomerative Clustering due to its superior performance. We compared the effectiveness of our embedding and clustering steps to BERTopic to show its effectiveness. In argument selection, we score each argument in a cluster to select the best one. We experimented with 11 different scoring functions and selected two, Selection with Matching Model (SMM) and Selection with Scoring Function (SSF), based on our evaluation criteria.

Chapter 4

Evaluation Metrics

4.1 Overview

This section centers around evaluation metrics for key point generation task (KPG) (see 1.3). It starts by discussing the most popular evaluation metric in summarization tasks, ROUGE, and its limitations. We introduce coverage datasets, a set of datasets used for assessing evaluation metrics.

Next, we propose our two evaluation metrics for KPG, coverage and redundancy. We experiment with different methods and models for measuring coverage, including BLEU score, BERT and RoBERTa models, a key point matching model 3.5.2, and BLEURT and BARTScore models. Moreover, we devise a dataset for redundancy task to train and compare different methods. We experiment two approaches, a cosine similarity based method and using BERT for classification. We present the best performing methods for both coverage and redundancy at the end of chapter.

4.2 ROUGE

In the realm of summarization tasks, the prevalent evaluation metric is the "Recall-Oriented Understudy for Gisting Evaluation," abbreviated as ROUGE [5]. ROUGE score measures the overlap of n-grams between a candidate text and a reference text and is employed in various tasks, including translation. The two most frequently used variants in summarization evaluation are ROUGE-N (ROUGE-1, ROUGE-2)

and ROUGE-L. However, despite its widespread use, ROUGE has notable limitations, such as its inability to consider semantic similarity and its inclination toward favoring longer sentences [46]. Furthermore, it is worth noting that according to [47], many reported ROUGE scores in research may be incorrect due to inaccuracies in the software packages used for ROUGE calculation, and because some researchers fail to account for critical evaluation decisions and parameters when reporting ROUGE scores, thereby making their experiments not reproducible.

4.3 ROUGE in Key Point Generation

Moreover, we contend that the ROUGE score, on its own, proves inadequate as a metric for our specific task of argument summarization, primarily because it lacks the capability to distinguish between summaries that exhibit varying degrees of coverage. To illustrate this limitation, we conducted a study in which we sampled diverse datasets, each representing different levels of coverage, and computed their ROUGE scores using the "rouge_score" package in Python. Specifically, we generated nine distinct datasets for each coverage level, 100%, 75%, 50% coverage datasets from the ArgKP training set. Our experiments showed that ROUGE is incapable of differentiating datasets with different levels of coverage (More in 5.3.2).

4.4 Coverage Datasets

To test the performance of different evaluation metrics, we created a set of pseudo summaries from the ArgKP test set, with different levels of coverage, 100%, 75%, and 50%. We named them Coverage Datasets – a dataset for each level of coverage. These pseudo summaries each contain the same number of arguments i.e. 25, sampled from the unseen ArgKP test set. Each coverage dataset corresponds to its respective proportion of reference key points from the topic. For instance, a 100% coverage dataset for a topic with 20 key points encompasses all 20 key points, while

a 75% coverage dataset covers 15 key points, and a 50% coverage dataset includes 10 key points. To ensure a fair comparison across all datasets, we maintained an equal number of arguments within each dataset. For instance, the 100% coverage dataset contained 25 arguments in total, with at least one argument assigned to each key point, while the 50% coverage dataset containing 25 arguments in total, has at least two arguments corresponding to each key point. Furthermore, for each coverage dataset, we conducted random sampling of nine distinct versions and reported the average score across all these samples. This step was implemented to reduce randomness and ensure that different key points and arguments were selected. Additionally, any experiment using different coverage datasets reported the average scores across all nine datasets.

4.5 Proposed Metrics

To address these concerns, we have introduced a pair of novel evaluation metrics, namely, "coverage" and "redundancy", tailored specifically for the task of argument summarization. In both key point analysis and argument summarization, the primary objective is to generate a concise summary that closely aligns with the key points featured in the reference summary. Consequently, we designed our evaluation metrics with these same objectives in mind. Furthermore, we aimed to devise evaluation metrics that are not only intuitive but also straightforward to implement and apply.

Coverage, quantifies the extent to which a generated summary encompasses the key points or reference summary (i.e. how many reference key points are covered by the generated key point). A high coverage score signifies that the generated summary effectively captures most of the key points outlined in the reference summary.

On the other hand, redundancy gauges the number of distinct arguments present in the generated summary (i.e. duplicate generated key points that cover the same reference key point). A low redundancy score indicates that the majority of the arguments in the generated summary are distinct or dissimilar.

Since evaluating these metrics often necessitates a dataset with labeled key points and arguments, which is typically unavailable, we also introduce techniques for calculating these metrics without relying on labeled data.

In terms of the practical implementation of coverage and redundancy prediction methods, we have explored various techniques and approaches. Nevertheless, a key emphasis of these evaluation metrics is ease of implementation and usage, with the intention of enabling other researchers to readily adopt them without any additional complexity. The subsequent two sections provide a comprehensive overview of the diverse approaches we have adopted for evaluating coverage and redundancy.

4.6 Coverage

The coverage metric assesses the quality of a summary based on how effectively it represents the reference summary, a similar concept to ROUGE. In the context of a reference summary comprising reference key points denoted as $RKP = \{kp1, kp2, \ldots, kpn\}$, with each key point containing one or more arguments, and a candidate summary represented as $GKP = \{arg1, arg2, \ldots, argn\}$, coverage quantifies the extent to which the generated key points (arguments) in the candidate summary cover the reference key points.

As an example, the coverage score of a summary or "generated key points" $GKP = \{arg1, arg2, arg3\}$ with respect to the reference key points, $RKP = \{kp1, kp2, kp3, kp4\}$, given that arg1 and arg3 belong to kp1, and arg2 belongs to kp2 is equal to 50%. The coverage score is calculated using the Algorithm 4.

Assessing coverage, given knowledge of the arguments for each key point, is a straightforward task, i.e. when it is clear what reference key points the generated key points belong to. However, it's important to note that this information may not always be readily available. Therefore there is a need for an automatic method of assigning arguments to reference key points. We framed the coverage measure as an entailment task. Consequently, we conducted experiments utilizing four distinct

Algorithm 4 Coverage Metric

Input: $GKP = \{arg1, arg2, \ldots, argn\}$ & $RKP = \{kp1, kp2, \ldots, kpn\}$
Output: $CoverageScore$

$CoveredKeyPoints = Set(empty)$
for $arg_i = arg1, arg2, \ldots, argn$ **do**
    for $kp_i = kp1, kp2, \ldots, kpn$ **do**
        if arg_i *is in* kp_i **then**
            $CoveredKeyPoints.add(kp_i)$
        end if
    end for
end for
return $CoverageScore \leftarrow \frac{\#\ CoveredKeyPoints}{\#\ ReferencePoints}$

automatic evaluation methods for coverage and documented their individual performance.

4.6.1 BLEU Score

Initially, we conducted an experiment to evaluate the applicability of the BLEU Score in determining coverage. In this experiment, we organized the dataset by key points and allocated arguments to their corresponding key points. Subsequently, we calculated the BLEU Score between each candidate argument in the summary and each key point's group of arguments. The BLEU Score between the candidate argument and the group of arguments yields a score between zero to one. A score of one indicates that the candidate argument was identical to an argument within the key point's group, while a score of zero means there is no 1, 2, 3, or 4-grams shared between the two. We assigned the candidate argument to the key point with the highest BLEU Score. To ensure fairness, we removed the candidate argument from the group of arguments associated with the key point. Table 4.1 shows the predicted coverage of different coverage datasets using BLEU score.

The results obtained from the test set indicates that the BLEU Score is not a suitable metric for determining the key point to which a candidate argument belongs. The primary reason for BLEU score's limitation is that it operates at the lexical level,

neglecting the semantic aspects of text. While such an approach may suffice for tasks like summarization or translation, it proves less effective in the context of argument summarization. In argument summarization, arguments with distinct stances and key points can contain shared words, and arguments that pertain to the same key point may be expressed using different phrasings or word choices.

Predicted coverage	Topic 1	Topic 2	Topic 3
100% Coverage	44%	40%	25%
75% Coverage	44%	30%	25%
50% Coverage	22%	20%	16%

Table 4.1: The predicted coverage of different coverage datasets using BLEU metric.

4.6.2 Encoder Model: BERT & RoBERTa

The second proposed approach for determining coverage involves utilizing an encoder-only model to assign arguments to key points. We employed the BERT and RoBERTa (see 2.2.9) base checkpoints from Huggingface as our base models and subsequently fine-tuned them for the task of binary classification. It is important to note that we use the same model and idea as the key point matching model mentioned in Section 3.5.2. Similarly, the input data format for the model is structured as follows: [CLS] argument [SEP] key point [SEP], and the model's output is configured as [logit 1, logit 2]. Here, logit 1 represents the probability of the argument not being associated with the key point, while logit 2 signifies the probability of the argument being related to the key point. The argument-key point pair is labeled matching (1) if logit 2 ¿ logit 1, and non-matching (0) otherwise. Figures 4.1 and 4.2 provide details on the training loss and accuracy, as well as the validation loss and accuracy, for both the BERT and RoBERTa models. These models were fine-tuned for ten epochs using the Adam optimizer with a learning rate of 1e-6 and a batch size of 32. The template

used for fine-tuning the model was taken from [1].

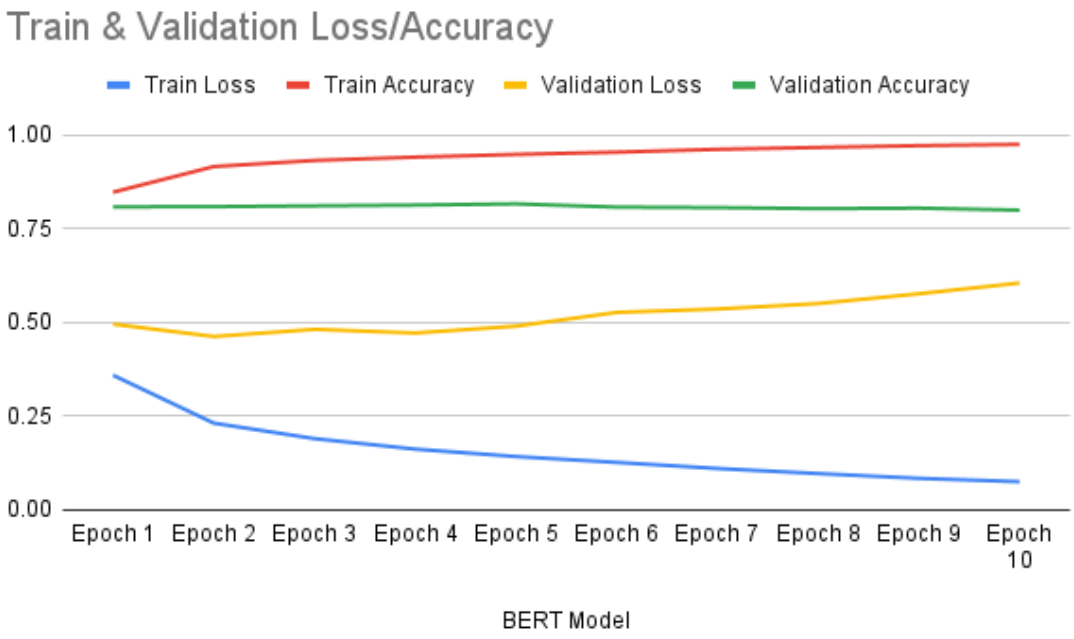

Figure 4.1: Training and validation loss and accuracy of BERT model

Setting Threshold

In the inference phase, we apply a softmax layer on the encoder model's output, which enables us to apply a threshold. The threshold is applied on the second logit, making positive predictions more restrictive. With a threshold, a pair is only labeled as matching when $logit2 > logit1$ and $logit2 > threshold$. Our experiments indicated that setting a threshold for assigning labels to the model's outputs is an effective approach. By default, the model designates non-matching labels to output logits $[logit1, logit2]$ if $logit1 > logit2$. After a thorough examination of the output logits and further experiments, we discovered that setting a threshold of 0.9, following the application of softmax for matching labels (i.e., $logit1 < logit2$ and $logit2 > 0.9$), significantly enhances coverage accuracy.

[1] https://towardsdatascience.com/fine-tuning-pre-trained-transformer-models-for-sentence-entailment-d87caf9ec9db

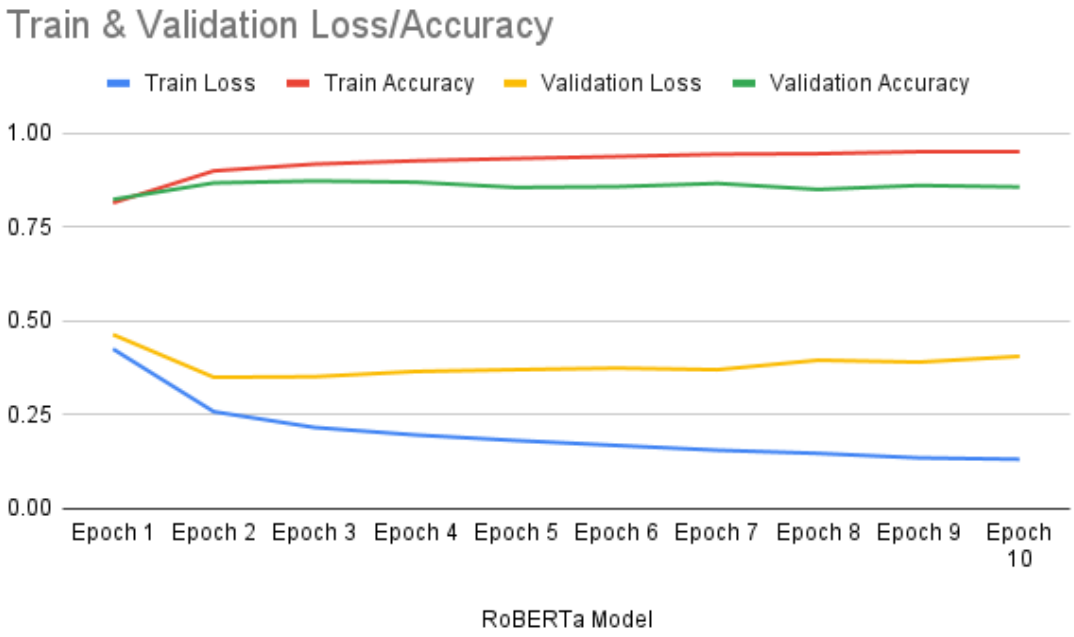

Figure 4.2: Training and validation loss and accuracy of RoBERTa model

To demonstrate efficacy of thresholds, we conducted tests using the model on various coverage datasets. Table 4.2 and Table 4.2 illustrates the accuracy of the model under different conditions, including without a threshold and with the inclusion of softmax in conjunction with a threshold.

BERT Model	100% Coverage	75% Coverage	50% Coverage
Accuracy w/o Threshold	95.97% (0)	83.21% (0.02)	70.37% (0.02)
Accuracy w 0.9 Threshold	92.68% (0.01)	77.96% (0.01)	63.51% (0.02)

Table 4.2: The predicted coverage of different coverage datasets using BERT with and without threshold. The values in parentheses represent standard deviation.

RoBERTa Model	100% Coverage	75% Coverage	50% Coverage
Accuracy w/o Threshold	90.8% (0.01)	82.57% (0.01)	73.57% (0.02)
Accuracy w 0.9 Threshold	78.09% (0.01)	70.37% (0.01)	58.34% (0.02)

Table 4.3: The predicted coverage of different coverage datasets using RoBERTa with and without threshold. The values in parentheses represent standard deviation.

Setting Limit

Additionally, we conducted experiments exploring alternative configurations for the coverage model. By default, whenever the model predicts that a key point is implied by an argument—indicating that the second logit of the model's output is greater than the first (i.e., for output $[logit1, logit2]$, $logit1 < logit2$)—it signifies that the key point is covered. This often results in an argument matching with multiple key points. However, in practice, an argument typically covers one or, at most, two key points, especially given that the arguments in the dataset are in the form of short tweets. To enforce this, we limit the number of key points each argument can cover to a maximum of one or two, assigning the key point with the highest scores (logit 2) to each argument.

Table 4.4 and Table 4.5 presents the model's predicted coverage under various scenarios, including with no limit on the number of key points assigned to a key point, a maximum of one key point, and a maximum of two key points assigned.

BERT Model	100% Coverage	75% Coverage	50% Coverage
No Limit	95.97% (0)	83.21% (0.02)	70.37% (0.02)
Max One KP	82.12% (0.01)	65.29% (0.02)	50.52% (0.01)
Max Two KP	93.82% (0)	79.24% (0.01)	65.15% (0.02)

Table 4.4: The predicted coverage of different coverage datasets using BERT with limit on the number of key points. The values in parentheses represent standard deviation.

RoBERTa Model	100% Coverage	75% Coverage	50% Coverage
No Limit	90.8% (0.01)	82.57% (0.01)	73.57% (0.02)
Max One KP	72.24% (0.02)	61.54% (0.01)	51.53% (0.01)
Max Two KP	87.51% (0.01)	77.64% (0.01)	66.39% (0.02)

Table 4.5: The predicted coverage of different coverage datasets using RoBERTa with limit on the number of key points. The values in parentheses represent standard deviation.

Applying Both Settings

Seeing the improvements in accuracy scores resulting from the inclusion of both thresholds and limitations on the maximum number of key points per argument, we conducted experiments incorporating both these measures. Table 4.6 and Table 4.7 showcases some of the most favorable outcomes we achieved through these combined settings.

The experiments have demonstrated that the optimal configuration for our model involves setting a threshold of 0.9 and allowing for a maximum of two key points per argument.

BERT Model	100% Coverage	75% Coverage	50% Coverage
Max One KP, 0.9 Threshold	80.56% (0.01)	63.92% (0.01)	49.51% (0.01)
Max Two KP, 0.9 Threshold	90.71% (0)	75.26% (0.01)	60.53% (0.02)

Table 4.6: The predicted coverage of different coverage datasets using BERT with threshold and limit on the number of key points. The values in parentheses represent standard deviation.

RoBERTa Model	100% Coverage	75% Coverage	50% Coverage
Max One KP, 0.9 Threshold	64.74% (0.01)	56.1% (0.01)	46.41% (0.01)
Max Two KP, 0.9 Threshold	76.17% (0.01)	67.48% (0.01)	56.01% (0.01)

Table 4.7: The predicted coverage of different coverage datasets using RoBERTa with threshold and limit on the number of key points. The values in parentheses represent standard deviation.

Based on these results, we have decided to adopt the BERT model with a maximum of one key point setting as our evaluation method. We opted for BERT over RoBERTa due to its superior accuracy on the test set. Furthermore, we chose the maximum one key point over two, despite its superior performance, as not to overfit our metric on one dataset. It is also important to note that we only use the threshold when experimenting on ArgKP dataset, and do not use it when experimenting on other

dataset as the threshold is fine-tuned for this dataset. Table 4.8 provides the accuracy scores of both BERT and RoBERTa models on the ArgKP test set.

Model	Accuracy
BERT	90.01%
RoBERTa	82.25%

Table 4.8: The accuracy of BERT and RoBERTa with 0.9 threshold and limit of one on the number of key points on the ArgKP test set.

4.6.3 Key Point Matching Model in the Literature

As per the key point analysis task overview, it was reported that Alshomary presented the top-performing model for the key point matching track. Consequently, we made the decision to conduct experiments with this proposed model and utilize it for predicting coverage. This model was fine-tuned using the key point analysis dataset, employing contrastive loss for training. It calculates embeddings for input data and matches arguments to key points based on the highest cosine similarity score, provided the score exceeds a specified threshold. We conducted experiments with this model across various coverage datasets, allowing for the assignment of a maximum of one or two key points per argument. The table 4.9 presents the predicted coverages achieved through these experiments. The results indicate that while Alshomary's KPM model performs well, it is less accurate than the proposed encoder models.

Alshomary's Model	100% Coverage	75% Coverage	50% Coverage
Max One KP, No Threshold	74.66% (0.02)	65.88% (0.01)	56.65% (0.02)
Max Two KP, No Threshold	92.5% (0.02)	85.5% (0.01)	78.64% (0.01)
Max Two KP, 0.5 Threshold	91.35% (0.01)	83.95% (0)	76.54% (0.01)

Table 4.9: The predicted coverage of different coverage datasets using Alshomary's KPM model. The maximum one key point with no threshold is the original setting used by the authors, the rest are experimental settings. The values in parentheses represent standard deviation.

4.6.4 BLEURT & BARTScore

Li et al. [34] introduced a new evaluation metric for the task of key point generation. The authors suggested computing soft precision and soft recall using the similarity score computed by BLEURT and BARTScore, matching key points and arguments with the highest cosine similarity. Soft-precision finds the reference key point with the highest similarity score for each generated key point, and soft-recall finds the generated key point with the highest similarity score for each reference key point. In order to assess the efficacy of their approach, we conducted tests using the BLEURT and BARTScore models to predict the coverage of different coverage datasets. To do this, we computed scores for every possible argument and key point pairing while allowing for the assignment of a maximum of one or two key points to each argument. This evaluation aimed to determine the precision of BLEURT and BARTScore in correctly assigning arguments to the appropriate key points. Table 4.10 and Table 4.11 displays the predicted coverage achieved using these methods. The results suggest that BLEURT and BARTScore struggle to effectively link arguments with their corresponding key points. The primary reason for this is that these models have not been fine-tuned on text resembling arguments.

BLEURT	100% Coverage	75% Coverage	50% Coverage
Max One KP	66.25% (0)	61.11% (0.01)	56.37% (0.02)
Max Two KP	82.92% (0)	81.68% (0)	75.1% (0.02)

Table 4.10: The predicted coverage of different coverage datasets using BLEURT model using maximum one and two key points. The values in parentheses represent standard deviation.

4.7 Redundancy

Redundancy assesses the number of distinct generated key points present in the generated summary. When considering a summary $GKP = \{arg1, arg2, \ldots, argn\}$,

BARTScore	100% Coverage	75% Coverage	50% Coverage
Max One KP	60.28% (0.04)	59.25% (0.02)	57.61% (0.02)
Max Two KP	79.62% (0.01)	79.62% (0.02)	78.8% (0.02)

Table 4.11: The predicted coverage of different coverage datasets using BARTScore model using maximum one and two key points. The values in parentheses represent standard deviation.

redundancy measures how many arguments within the candidate summary address the same key point.

As an example, the redundancy score of a summary or $GKP = \{arg1, arg2, arg3\}$ with respect to the reference key points, $RKP = \{kp1, kp2, kp3, kp4\}$, given that arg1 and arg3 belong to kp1 and arg2 belongs to kp2, is equal to 1/3 or 33%. The redundancy score is calculated using Algorithm 5.

Algorithm 5 Coverage Metric

Input: $GKP = \{arg1, arg2, \ldots, argn\}$
Output: $RedundancyScore$

$DuplicatePairs = 0$
$AllPossiblePairs = \frac{n*(n-1)}{2}$
for $arg_i = arg1, arg2, \ldots, argn$ **do**
    for $arg_j = (arg1, arg2, \ldots, argn) - arg_i$ **do**
        if arg_i *and* arg_j *cover same key point* **then**
            $DuplicatePairs++$
        end if
    end for
end for
return $RedundancyScore \leftarrow \frac{\#\ DuplicatePairs}{\#\ AllPossiblePairs}$

$$RedundancyScore = \frac{\#Generated\ Key\ Points\ in\ the\ Same\ KP}{\#All\ Possible\ GKP\ Pairs}$$

However, quantifying redundancy becomes challenging when we lack information regarding which generated key points (arguments) belong to the same reference key point. In this regard, we conducted experiments using various methods to automatically gauge the redundancy of an output and presented their performance. We

framed the redundancy measure as a paraphrase detection or classification task, aiming to identify non-redundant, i.e. unique, arguments. Alongside these measures, we explored the creation of datasets for the binary task of paraphrase detection for arguments.

4.7.1 Comparing to Standard Paraphrase Detection

Detecting paraphrases for arguments poses a more intricate challenge than standard paraphrase detection. This increased complexity arises because arguments often employ similar words to convey different stances with opposing meanings. For instance, consider a topic like abortion, where sentences such as "Fetus is a human" and "Fetus is not a human" exhibit substantial lexical similarity but belong to contrasting stances. Additionally, sentences can discuss the same topic, stance, and aspect while utilizing entirely different wording. To validate our assertion, we fine-tuned a BERT model for paraphrase detection using the Quora Paraphrase dataset [2] and subsequently evaluated its performance on a dataset containing argument paraphrases from ArgKP dataset. Table 4.12 displays the training and testing accuracies. The results indicate that even after fine-tuning BERT on a paraphrase detection dataset and achieving an acceptable training accuracy, the model is still incapable of distinguishing paraphrased sentences in the argument domain.

BERT Model	Train Loss	Train Accuracy	Validation Loss	Test Accuracy
Before Fine-Tuning	-	-	3.1997	45.44%
After Fine-Tuning	0.3283	85.02%	2.4972	53.87%

Table 4.12: The accuracy score of a BERT model for paraphrase detection. The model is trained on Quora Paraphrase dataset and tested on a paraphrase dataset generated from ArgKP.

[2] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

4.7.2 Dataset Creation

To facilitate the training and evaluation of paraphrase detection models specific to arguments, we had to construct a dedicated dataset for this purpose. Existing datasets for paraphrase detection did not focus on argumentative and therefore were inadequate for our requirements, necessitating the creation of a new dataset. To compile this dataset, we explored three distinct approaches.

First Approach

Our initial approach involved leveraging pre-trained large language models, where we prompted these models to generate paraphrases for given arguments. We used the prompt "Rephrase the following sentence" and added the argument afterwards. We employed Stanford's Alpaca [3] model as part of this process. Nevertheless, the paraphrases generated by this model exhibited a level of similarity to the original input that was too pronounced. Table 4.13 provides examples of the generated paraphrases.

The provided examples clearly illustrate that paraphrases generated by pre-trained language models tend to be quite simplistic, often limited to the replacement of only a few words with synonyms. In reality, similar arguments may exhibit entirely different phrasing and structural variations.

Original Text	Paraphrased Text
Parents should be permitted to choose the education of their children	Mothers and fathers should be allowed to decide on the education of their offspring
Libertarianism increases happiness/fulfilment	Libertarianism promotes happiness/fulfilment
Space exploration improves science/technology	Space travel advances science/technology

Table 4.13: Paraphrases generated by Alpaca model

[3] https://crfm.stanford.edu/2023/03/13/alpaca.html

Second Approach

To address previous method's limitation, we sought to create a new dataset for paraphrase detection by sampling paraphrases from the ArgKP dataset instead of generating new paraphrases with an LLM. This process involved sorting the dataset by topics and, for each key point, identifying every argument associated with that key point. Arguments pairs from the same key point were then collected as positive samples for our dataset. To provide negative samples, an equal number of arguments were selected and paired with arguments from different topics. A selection of examples from this dataset is presented in the Table 4.14.

Original Text	Paraphrased Text	Label
adopting an austerity regime is the best way for the country to reduce its huge debt	it is a good way to control overspending and accumulating more debt	1
people who have mental illness should have not right to keep and bear arms	an austerity regime would help set our economy back on the right track	0
uniforms create a sense of equality among students, regardless of wealth or background	an austerity regime would save money for the country.	0
school uniform is unaffordable for many single parents and should be abandoned.	school uniform should be abolished because it is very expensive	1

Table 4.14: Paraphrased arguments using the second approach. Label 1 indicates a positive example, where the paraphrase is belongs to the same key point as the original text. Label 0 indicates negative examples, where original text and paraphrase are from different topics.

After fine-tuning BERT on the proposed dataset, we observed that the task is too trivial for the model and does not represent the real world data well enough. The reason for it is that the negative samples are too easy for the model to distinguish

between, as they are from different topics, and mostly use different language. Table 4.15 shows the performance of the model after fine-tuning for one epoch.

BERT Model	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
After Fine-Tuning	0.0562	98.27%	0.4047	87.97%

Table 4.15: The training and validation accuracy of BERT model after fine-tuning the model for 1 epoch on the proposed dataset.

Third Approach

To enhance the quality of the dataset, we opted for an alternative method to generate negative samples. While the positive samples remained the same as in the previous approach, the negative samples now consisted of arguments from the same topic but belonging to different key points. This adjustment ensures that positive samples are real examples, while negative samples share similar keywords since they pertain to the same topic. We used this refined dataset to train the encoder models discussed in subsequent sections. Several examples of both positive and negative samples are provided in Table 4.16.

4.7.3 Cosine Similarity & Embedding

Approach I: Using embeddings from BERT

In our initial approach, we utilized BERT sentence embeddings for paraphrase detection. We applied two distinct methods: firstly, we extracted and normalized the encoding for the CLS token (the first token of the BERT model), followed by calculating the cosine similarity between all pairs of arguments. The second technique involved computing the mean of sentence embeddings and subsequently determining the cosine similarity between all pairs of arguments. For both methods, a manually chosen threshold was employed to classify arguments as either paraphrased or non-paraphrased. These techniques were evaluated on a dataset containing arguments and paraphrases generated by ChatGPT. The results revealed that both methods

Original Text	Paraphrased Text	Label
implementing an austerity regime risks cutting programs that citizens depend on such as welfare.	historically, austerity regimes have resulted in unemployment.	0
it is a good way to instill discipline	School uniforms help unite the kids from each school, much like sports uniforms help unite sports teams.	0
school uniform is necessary to avoid poor students to feel any different	school uniforms make is so that everyone is equal	1
an austerity regime hurts the poorest in our society	austerity regimes disproportionally hit the poorest in society hardest.	1

Table 4.16: Paraphrased arguments using the third approach. The topic of discussion is "School uniforms should be abolished". Label 1 indicates a positive example, where the paraphrase is belongs to the same key point as the original text. Label 0 indicates negative examples, where original text and paraphrase are from different topics.

frequently identified different arguments as paraphrases, which was not the desired outcome.

Approach II: Using embeddings from SBERT

The second approach involved utilizing a SBERT to generate sentence embeddings for the arguments. Cosine similarity, combined with a threshold, was used to classify paraphrases. We utilized the same fine-tuned SBERT model mentioned in Section 3.5.1 for clustering, as fine-tuning the language model demonstrated enhancements in clustering. We extended this idea to duplicate detection, believing it could yield similar improvements. Like the previous approach, we extracted embeddings for sentence pairs from the fine-tuned language model and computed their cosine similarity scores.

However, in this case, we introduced a normalization step for the cosine similarity scores. We applied this approach to the training data and tested various thresholds in 0.1 increments to identify the optimal threshold. Table 4.17 displays the accuracy scores for different thresholds on the training data.

Threshold	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Accuracy	0.5	0.5	0.53	0.64	0.75	0.82	0.86	0.85	0.79	0.64	0.5

Table 4.17: The accuracy scores for duplicate detection on the ArgKP-based dataset training set for different values of threshold.

As shown above the 0.5, 0.6, and 0.7 thresholds had the best accuracies, so we applied these three thresholds to the validation dataset. Table 4.18 shows the accuracy of the proposed method on the validation set.

Threshold	0.5	0.6	0.7
Accuracy	0.68	0.72	0.73

Table 4.18: The accuracy scores for duplicate detection on the ArgKP-based dataset validation set for different values of threshold.

Table 4.19 shows the accuracy for applying the best threshold, 0.7, on the test set.

Cosine Similarity	Test Accuracy
	72.77%

Table 4.19: The accuracy scores for duplicate detection on the ArgKP-based dataset test set for different values of threshold.

4.7.4 Encoder Model: BERT

In addition to the previous approach, we also conducted experiments involving the fine-tuning of BERT for classification, as it proved effective in coverage metric. We also explored the use of RoBERTa model, however, initial experiments with RoBERTa yielded unsatisfactory results, leading us to focus our further experiments solely on

BERT. We used the same fine-tuned BERT model mentioned in Section 3.5.2. We used the Adam optimizer and experimented with two different learning rates, namely 2e-5 and 1e-6, for 8 epochs, employing a batch size of 16. Figures 4.3 and 4.4 illustrate the training and validation loss and accuracy for both learning rates.

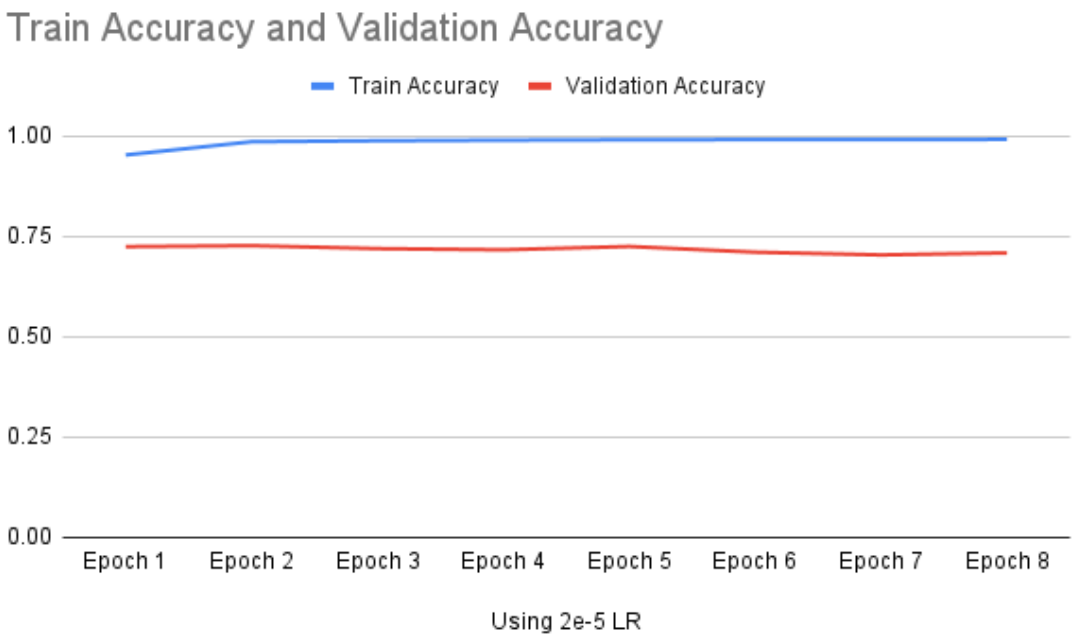

Figure 4.3: Training and validation loss and accuracy of BERT model with learning rate of 2e-5

After evaluating the performance of different learning rates, we have opted to utilize the second model with a learning rate of 1e-6 as the redundancy evaluator model. This decision was driven by its superior validation accuracy. Table 4.20 presents the accuracy of the redundancy model on the ArgKP test set.

BERT Model	Test Accuracy
	67.63%

Table 4.20: The accuracy scores for duplicate detection on the ArgKP-based dataset test set using BERT model.

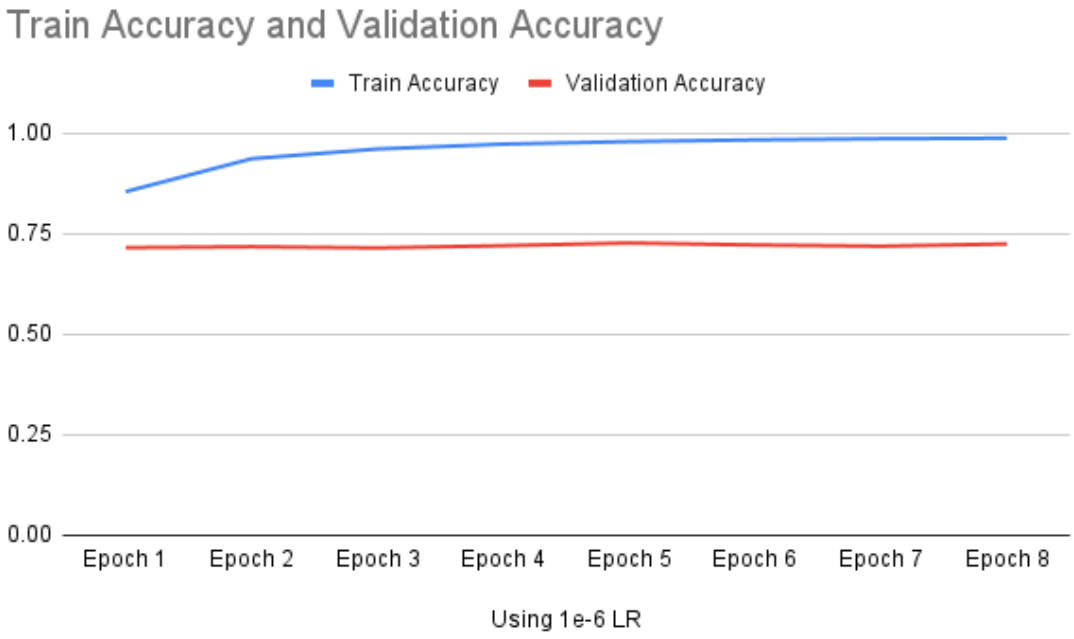

Figure 4.4: Training and validation loss and accuracy of BERT model with learning rate of 1e-6

4.8 Results

4.8.1 Best Performing Evaluation Models

Based on the results evaluations done on the proposed paraphrase detection dataset, we have selected the following methods for evaluating coverage and redundancy. For the coverage metric, we have selected the **BERT model with maximum one key point and threshold**. For redundancy evaluation, we choose C**osine Similarity using SBERT model with a threshold of 0.7**. We further examine the outputs of different models using these two methods and other evaluation metrics to showcase their efficacy.

4.8.2 Comparing Coverage and Redundancy

We have introduced two distinct evaluation metrics for this task, each offering its own set of applications and advantages. While the coverage score provides insights into the comprehensiveness of generated summaries, indicating that a high coverage

score implies that the generated summary adequately encompasses the primary points from the reference summary, it is also likely to yield a low redundancy score when the number of key points in the generated summaries aligns with or is less than the number of key points in the reference summary. Nevertheless, the coverage metric relies on the availability of reference summaries, which is not a prerequisite for the redundancy score. The redundancy measure, on the other hand, assesses the similarity between all pairs of generated key points. However, the limitation of the redundancy metric becomes evident when a model generates unique yet unrelated summaries that fail to cover the reference key points. In such cases, the generated summary may receive a low redundancy score, but it falls short in producing a high-quality summary.

4.9 Conclusion

In this section we propose two measures for evaluating generated key points in key point analysis task, coverage and redundancy. We experiment with different techniques and settings for evaluating these measures in order to find the optimal one.

For measuring coverage, we experiment with four different techniques. We experiment BLEU score, using encoder-based models (BERT and RoBERTa) for classification, a previous SOTA key point matching model, and BLEURT and BARTScore based methods. We also introduced a set of datasets called coverage datasets that are used to evaluate the effectiveness of different approaches. Our initial experiments showcased the effectiveness of encoder-based approaches for coverage. As a result we experimented with different settings for tuning them, such as setting a threshold and a limit. Our final experiments suggest that BERT model with maximum one KP limit and threshold has the best performance (i.e. correlating high coverage with a high score). As for the other approaches, BLEU score did not perform well as a metric; and both the previous key point matching model, and BLEURT and BARTScore based methods, while having a good performance, did not perform as well as the BERT model.

For measuring redundancy, we first created a dataset for paraphrase detection in argument-domain. We experimented with different approaches for creating a dataset that best represents actual paraphrases. We further used the dataset to train and test two different models for paraphrase detection, cosine similarity and BERT-based classifier. For the cosine similarity approach, we calculated the cosine similarity between embeddings, and experimented with two different embedding generation models, BERT and SBERT. Our experiments showed that the cosine similarity method with SBERT for embeddings outperforms the other approaches.

Chapter 5

Experiments

5.1 Overview

This section aims to cover experiments that showcase the efficiency of the proposed approaches for key point generation and evaluation compared to previous approaches. First we evaluate our method by comparing it in terms of coverage, redundancy, argument quality, and length to previous SOTA approaches. Next, we report the accuracy of our evaluation metrics, and compare them to previous evaluation metrics for coverage prediction. Lastly, we extend our experiments for both the method and evaluation metric to another unseen dataset, to ensure the generalizability of both of our approaches.

For the summary (key point generation) model, we selected the selection methods SMM and SSF 3.5.2. As for the evaluation metrics, coverage and redundancy, we selected the best performing BERT [8] model for coverage prediction and the cosine similarity using SBERT [11] method for redundancy, as mentioned in Section 4.8.1.

5.2 Summarization Method Experiments

5.2.1 Coverage and Redundancy

To demonstrate the effectiveness of our approach, we conducted a comparative analysis of key point coverage and redundancy between our method and previously top-performing approaches, as determined by the results of the key point analysis task [4].

We calculated the expected (or actual) coverage of the outputs generated by various methods.

To illustrate this, let's consider a scenario where a method's output summary includes "argument 1," and according to the dataset, "argument 1" is associated with "key point 2". In this case, it means the summary covers "key point 2". In order to compute redundancy, we compute the percentage of duplicate pairs in the output. Two arguments are considered duplicates if they belong to the same key point. However, approximately 27% of the arguments in the dataset, as per [34], are not associated with any key point. In such cases, we grouped all these arguments together, assuming they are either too general to be linked to a key point or represent unique and isolated outliers.

Since each method generates a varying number of output key points, we selected the top k outputs for each method to ensure a fair comparison. We set the value of k equal to the number of reference key points.

In our experiments, we compared our approach to Alshomary's method [28] and BarHaim's method [27]. We evaluated Alshomary's work in two settings. In the first, PageRank w SS, the output provided by the authors of the paper is used, i.e. the original KPA task 3.2. The second setting, PageRank, represents the output of their method applied to input data for both stances on a topic, i.e. the arguments for stances "pro" and "against" were merged and used as input for the model, similar to other approaches. For BarHaim's method, we utilized the API provided by the authors, where the input data was merged for both stances.

It's worth noting that BarHaim's method typically generates fewer key points than the reference key points, usually between 4 to 8, whereas reference key points typically number more than 10. Moreover BarHaim's approach is incapable of generating summaries when the number of inputs is 100 or less, as a result we cannot generate the summary of separate stances similar to PageRank method. Table 5.1 illustrates the actual coverage of each method by topic.

ArgKP	Coverage	Redundancy	R1	R2	RL
Method (SMM)	**59.59%**	2.46%	0.152	0.026	0.132
Method (SSF)	57.67%	**2.27%**	0.158	0.026	0.141
Alshomary w SS	49.09%	3.41%	0.195	**0.032**	0.175
Alshomary	45.45%	3.23%	**0.202**	0.028	**0.186**
BarHaim	37.67%	5.26%	0.153	0.028	0.136

Table 5.1: Actual coverage, redundancy, and ROUGE scores for each model's output. The coverage and redundancy are computed using the labeled data. SS refers to separate stances. Numbers in bold represent the best results.

5.2.2 Length and Quality

To assess the quality of generated arguments, we asked human judges to evaluate arguments produced by different methods with respect to their correct key points. Inspired by the human evaluation conducted by Li et al. [34], we instructed the judges to assign scores to argument-key point pairs (+1 for matches, -1 for non-matches, 0 for uncertainties) based on two criteria: 1. Whether the argument covers the same aspect as the key point. 2. Whether the argument independently presents a clear and understandable perspective on an aspect given the topic (detailed instructions are available in Appendix B).

We selected a total of 20 generated key points from four topics, two from the Debate dataset (Section 5.4) and two from ArgKP, for each model. Three graduate students were tasked with scoring each pair, and we computed the average score across judges for each pair. In addition, we compared the average number of words per generated key point across all topics in the ArgKP test set to highlight the proposed method's capability to produce concise outputs.

To showcase the proposed method's ability to generate quality short sentences, we compare the average word length and the human-evaluated quality scores of generated key points. Table 5.2 shows the scores averaged over all outputs. The results indicate that our method outperforms Alshomary's in terms of conciseness, however, BarHaim

still generates shorter key points as it only extracts short arguments. Moreover, the human judges found the proposed methods' outputs more understandable, as the best method SMM having the highest score and method SSF having a similar score to the Alshomary's method. The human scores also indicate that longer outputs are easier to understand. The Krippendorff's alpha for inter-annotator agreement is 0.47 across all topics, with a score of 0.53 on the ArgKP dataset, and 0.40 on the Debate dataset.

Method	Avg. Words	Arg. Quality
Method (SMM)	18.6	**0.63**
Method (SSF)	12.4	0.4
Alshomary	15.3	0.46
BarHaim	**6.8**	0.23

Table 5.2: Average number of words and quality scores per generated key point, averaged for each model.

5.3 Evaluation Metric Experiments

5.3.1 Accuracy of Coverage and Redundancy Models

Before comparing the performance of our evaluation to other metrics, we report their accuracy on the ArgKP test set, which wsa not seen during training. This test set comprises three topics and resembles the structure of the training and validation sets. For the evaluation, we considered our most effective evaluation metrics, employing BERT and RoBERTa for coverage, and BERT and Cosine Similarity for redundancy, based on their validation accuracy. In the case of coverage, we employed the original dataset for calculating accuracy scores, while for paraphrase detection, we used the generated dataset described in Section 4.7.2. Table 5.3 illustrates the accuracy of the top-performing models in this evaluation. The results indicate that the BERT model with threshold and the cosine similarity method using SBERT have the best performance for coverage and redundancy respectively.

Coverage	BERT (1 KP)	BERT (Threshold) (1 KP)	RoBERTa (1 KP)	RoBERTa (Threshold) (1 KP)
Accuracy	77.72	**90.01**	80.26	82.25
Redundancy	BERT	Cosine Sim w SBERT	-	-
Accuracy	66.93	**72.77**	-	-

Table 5.3: The accuracy of best performing coverage and redundancy models on the ArgKP test set.

ArgKP	Coverage Measure	BLEURT	BARTScore	R1	R2	RL
100%	**83.18%** ± 0.052	66.26% ± 0.006	60.29% ± 0.044	0.166 ± 0.007	0.032 ± 0.005	0.150 ± 0.007
75%	**77.27%** ± 0.056	61.11% ± 0.015	59.26% ± 0.023	0.169 ± 0.010	0.033 ± 0.006	0.153 ± 0.010
50%	64.84% ± 0.055	**56.38%** ± 0.017	57.61% ± 0.029	0.165 ± 0.011	0.033 ± 0.006	0.151 ± 0.011

Table 5.4: Different methods for coverage prediction on datasets with various levels of coverage (100%, 75%, 50%). The numbers in bold represent the best results (closest prediction to the acutal coverage). The values in the subscripts represent the standard deviation.

5.3.2 Evaluation Metrics Comparison: Coverage Prediction

We assess the performance of our coverage measure compared to other metrics for KPG, namely ROUGE and Li et al. [34]. We assessed the metrics by tasking the evaluation metrics and models to predict the coverage of different Coverage datasets (Section 4.4). This allowed us to assess how good the evaluation metrics are at correlating an effective high coverage and low redundancy summary to a high score. First, we computed the ROUGE score on the datasets, with the key points as reference summaries. Second, we used the BLEURT and BARTScore models as used by [34] for computing soft-precision and soft-recall. We assigned every generated key point to the reference key point with the highest matching score, similar to the author's approach for calculating soft-F1 scores, which evaluates the effectiveness of models at correctly assigning key points to arguments. We compared the predicted coverage of each metric to the actual coverage. Table 5.4 illustrates the forecasted coverage of datasets using these evaluation metrics along with their corresponding ROUGE scores.

Our experiments demonstrate that the ROUGE score lacks the ability to distin-

guish between datasets with varying coverage levels. Although BARTScore exhibits only a slight improvement, as the difference in predicted coverages remains negligible. BLEURT, while performing reasonably well on the ArgKP dataset, struggles to generalize to the Debate dataset, as discussed in the subsequent section. We attribute the inefficacy of BLEURT and BARTScore to their lack of training on argumentative text, which is characterized by a distinct word distribution. Conversely, ROUGE's subpar performance can be attributed to its failure to consider the semantic nuances of sentences, a limitation which is especially true in the argument domain. Arguments on the same topic but belonging to different key points often employ similar words, which is not effectively addressed by ROUGE. The inaccurate coverage predictions of BLEURT and BARTScore indicate their inability to assign generated key points to the correct reference key points, rendering them ineffective at predicting the soft-F1 score proposed by the authors. In contrast, our proposed approach's predictions align closer with the actual coverage of datasets.

5.3.3 Evaluation of Generated Outputs

We have also evaluated the outputs of different models using various evaluation metrics. We used the proposed evaluation metrics, the soft-F1 scores, and ROUGE score. To calculate the rouge scores, we utilized the "rouge_score-0.1.2" library in Python. It's worth noting that we've previously demonstrated the limitations of the Rouge score when assessing key point summaries. Nevertheless, it remains crucial to report these scores, as Rouge is a widely recognized metric for evaluating summarization techniques. Table 5.5 shows the results.

These results suggest that our proposed coverage measure often estimates the relative ranking of outputs accurately. On the other hand, alternative evaluation metrics don't consistently link a high coverage/low redundancy output with a top score. Notably, both F1 measures identify Alshomary's output as the best among the evaluated outputs. However, according to Table 5.1 Alshomary's output has a lower coverage of

ArgKP	Coverage	Redundancy	F1 BL	F1 BS	R1	R2	RL
Method (SMM)	35.75% (**59.59%**)	(2.46%)	49.01%	57.94%	0.152	0.026	0.132
Method (SSF)	**39.89%** (57.67%)	(**2.27%**)	52.84%	60.81%	0.158	0.026	0.141
Alshomary	30.70% (45.45%)	(3.23%)	**54.58%**	**63.04%**	**0.202**	**0.028**	**0.186**
BarHaim	23.53% (37.67%)	(5.26%)	N/A	N/A	0.153	**0.028**	0.136

Table 5.5: The predicted coverage of proposed model, alongside the Soft-F1 scores using BLEURT (BL) and BARTScore (BS), and ROUGE score for summaries generated by different models. The numbers in parenthesis represent the actual coverage and redundancy. Numbers in bold represent the best output as evaluated by each metric.

reference key points and higher redundancy in generated key points. Also according to Table 5.2 it has lower argument quality and comparable length to method with SMM. In this case it is not clear how/why a certain summary is better than the others. Consequently, we believe that using a single metric for comparing generated key points is neither effective nor descriptive, since generated KPs have different aspects (coverage, redundancy, length, quality, etc) that require individual evaluation.

5.4 Experiments on Debate Dataset

To further validate the effectiveness of our proposed methods and evaluation metrics, we conducted tests on the Debate dataset [48], which encompasses four distinct topics. Within each topic, there are approximately five aspects per stance. Each input document in the dataset comprises multi-sentence arguments with labeled sentences corresponding to specific aspects. The Debate dataset consists of 3000 argument/key-point pairs. For this experiment, we excluded the "Other" aspect and focused on the remaining aspects, along with their associated arguments.

5.4.1 Debate Dataset: Method Evaluation

We used arguments from the debate dataset as inputs to BarHaim, Alshomary, and our method, and computed the coverage and redundancy of each model's output. Table 5.6 shows the actual coverage and redundancy of each method on the debate

Debate	Coverage	Redundancy	R1	R2	RL
Method (SMM)	**72.38%**	**1.42%**	0.06	0.005	0.052
Method (SSF)	57.98%	1.86%	0.079	0.007	0.069
Alshomary	64.88%	1.68%	0.068	0.003	0.062
BarHaim	56.41%	1.74%	**0.084**	**0.015**	**0.08**

Table 5.6: Actual coverage, redundancy, and ROUGE scores for each model's output. The coverage and redundancy are computed using the labeled data. SS refers to separate stances. Numbers in bold represent the best results.

Debate	Coverage Measure	BLEURT	BARTScore	R1	R2	RL
100%	82.08% ± 0.067	91.67% ± 0.015	**92.71%** ± 0.015	0.064 ± 0.002	0.005 ± 0.000	0.056 ± 0.001
75%	**76.67%** ± 0.040	90.63% ± 0.015	91.67% ± 0.015	0.066 ± 0.003	0.006 ± 0.001	0.057 ± 0.002
50%	**70.42%** ± 0.063	89.58% ± 0.059	94.79% ± 0.044	0.064 ± 0.003	0.006 ± 0.001	0.056 ± 0.003

Table 5.7: Different methods for coverage prediction on datasets with various levels of coverage (100%, 75%, 50%). The numbers in bold represent the best results (closest prediction to the actual coverage). The values in the subscripts represent the standard deviation.

dataset. The results indicate that the method with SMM outperforms all the other approaches, while method with SSF outperforms BarHaim.

5.4.2 Debate Dataset: Coverage Prediction

In order to demonstrate the generalizability of our evaluation metrics, we conducted a comparative analysis using the debate dataset dataset, contrasting our metrics with those proposed by [34]. We curated a series of coverage datasets from the Debate dataset, similar to Section 4.4, with varying coverage levels set at 100%, 75%, and 50%. Each of these datasets consists of 75 arguments, and we randomly sampled each coverage level ten times. The results from Table 5.7 indicate that the proposed metrics are much more generalizable to unseen datasets as shown by its predictions compared to previous approaches.

5.5 Conclusion

In this chapter we aimed to conduct various experiments on the proposed approach and evaluation metric, to show their effectiveness compared to previous works. First, we compared the our proposed approaches for summarization (key point generation) to previous SOTA, in terms of coverage, redundancy, length, and human evaluated quality. Second, we reported the accuracy of our best performing metrics, and compared the performance of our coverage metric to other evaluation metrics at predicting coverage. We also evaluated the summaries generated by different models by previous evaluation metrics and ours. Lastly, we replicate the same experiments for the approach and the metric on a new dataset. This was done to showcase the generalizability of our work on unseen data.

Our experiments on the summarization method show that our approaches outperforms previous SOTA in coverage and redundancy, while also offering high quality and concise summaries. As for our evaluation metrics, we showcase that they offer high accuracy in their respective tasks. Moreover, our coverage metric outperforms other metrics that also aim to measure coverage by larger margin.

Chapter 6

Conclusion and Future Work

In conclusion, this thesis has presented a novel approach to key point generation, addressing the limitations of previous state-of-the-art models such as low coverage and high redundancy. Through the integration of clustering techniques and a novel argument selection approach, the proposed method significantly improves coverage, redundancy, and overall summary quality. Additionally, the introduction of new evaluation metrics, namely coverage and redundancy, provides a more comprehensive assessment of summarization effectiveness compared to traditional metrics like ROUGE.

The experiments conducted demonstrate the superiority of the proposed approach over existing methods, showcasing its ability to generate more informative and concise summaries while minimizing redundancy. Furthermore, the newly introduced coverage evaluation metric proves to be a more accurate predictor of summary coverage, offering a valuable tool for future research in this domain.

6.1 Limitations

However, despite its contributions, this thesis is not without its limitations. Firstly, the proposed approach may still encounter challenges in handling highly nuanced or ambiguous arguments, which could affect the accuracy of key point generation. Additionally, the method does not guarantee generating a balanced number of key

points for both pro and against stances. This could potentially lead to summaries that favor one stance more often than the other.

The evaluation metrics, while more accurate the the previous approaches, are still not accurate enough to rank the models with 100% accuracy, when the actual coverage and redundancy of two generated summaries is insignificant. Additionally, the coverage evaluation metric does not factor in the hierarchical entailment relation between arguments and reference key points. As an example, the model might predict that a general reference key point (e.g. vaccinations may have unwanted side effects) is covered by a specific generated key point (e.g. vaccinations may have a specific side effect).

Additionally the evaluation metric proposed by Li et al [34] shows promising results. However, as shown by our experiments, a single number cannot accurately represent all the different aspects of a summary. Therefore using similar approaches for evaluating different aspects of quality would be more beneficial.

6.2 Future Work

In light of these limitations, several promising avenues for future research emerge. Firstly, exploring techniques for enhancing the model's ability to discern and prioritize salient arguments could lead to further improvements in summarization quality. Additionally, investigating alternative methods for evaluating summarization performance, including more nuanced measures of coherence and informativeness, could provide deeper insights into the strengths and weaknesses of different approaches. Moreover, both the summarization and the evaluation methods could potentially benefit from integrating large language models as they have shown promising results in various natural language processing tasks. Finally, extending the scope of application to other languages and domains could broaden the impact of this research and uncover new challenges and opportunities in argument summarization.

Bibliography

[1] M. Khosravani, C. Huang, and A. Trabelsi, "Enhancing key point generation via clustering: Prioritizing exhaustiveness and introducing an automatic coverage evaluation metric," in *2024 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2024. [Online]. Available: https://openreview.net/forum?id=80tEsVqk5W.

[2] M. Khosravani and A. Trabelsi, *Recent trends in unsupervised summarization*, 2023. arXiv: 2305.11231 [cs.CL].

[3] R. Bar-Haim, L. Eden, R. Friedman, Y. Kantor, D. Lahav, and N. Slonim, "From arguments to key points: Towards automatic argument summarization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 4029–4039. DOI: 10.18653/v1/2020.acl-main.371. [Online]. Available: https://aclanthology.org/2020.acl-main.371.

[4] R. Friedman, L. Dankin, Y. Hou, R. Aharonov, Y. Katz, and N. Slonim, "Overview of the 2021 key point analysis shared task," in *Proceedings of the 8th Workshop on Argument Mining*, Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 154–164. DOI: 10.18653/v1/2021.argmining-1.16. [Online]. Available: https://aclanthology.org/2021.argmining-1.16.

[5] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013.

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

[7] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation.," in *EMNLP*, vol. 14, 2014, pp. 1532–1543.

[8] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. arXiv: 1810.04805. [Online]. Available: http://arxiv.org/abs/1810.04805.

[9] A. Vaswani *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[10] Y. Liu *et al.*, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. arXiv: 1907.11692. [Online]. Available: http://arxiv.org/abs/1907.11692.

[11] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2019. [Online]. Available: https://arxiv.org/abs/1908.10084.

[12] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[13] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967, ISSN: 0033-3123.

[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96, Portland, Oregon: AAAI Press, 1996, 226–231.

[15] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971. DOI: 10.1080/01621459.1971.10482356. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1971.10482356. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1971.10482356.

[16] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, P. Isabelle, E. Charniak, and D. Lin, Eds., Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. [Online]. Available: https://aclanthology.org/P02-1040.

[17] M. Lewis *et al.*, "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*, Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703. [Online]. Available: https://aclanthology.org/2020.acl-main.703.

[18] T. Sellam, D. Das, and A. Parikh, "BLEURT: Learning robust metrics for text generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 7881–7892. DOI: 10.18653/v1/2020.acl-main.704. [Online]. Available: https://aclanthology.org/2020.acl-main.704.

[19] W. Yuan, G. Neubig, and P. Liu, "BARTScore: Evaluating generated text as text generation," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021. [Online]. Available: https://openreview.net/forum?id=5Ya8PbvpZ9.

[20] A. Misra, B. Ecker, and M. Walker, "Measuring the similarity of sentential arguments in dialogue," in *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Los Angeles: Association for Computational Linguistics, Sep. 2016, pp. 276–287. DOI: 10.18653/v1/W16-3636. [Online]. Available: https://aclanthology.org/W16-3636.

[21] C. Egan, A. Siddharthan, and A. Wyner, "Summarising the points made in online political debates," in *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 134–143. DOI: 10.18653/v1/W16-2816. [Online]. Available: https://aclanthology.org/W16-2816.

[22] Y. Ajjour, M. Alshomary, H. Wachsmuth, and B. Stein, "Modeling frames in argumentation," 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)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 2922–2932. DOI: 10.18653/v1/D19-1290. [Online]. Available: https://aclanthology.org/D19-1290.

[23] N. Reimers, B. Schiller, T. Beck, J. Daxenberger, C. Stab, and I. Gurevych, "Classification and clustering of arguments with contextualized word embeddings," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 567–578. DOI: 10.18653/v1/P19-1054. [Online]. Available: https://aclanthology.org/P19-1054.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://aclanthology.org/N19-1423.

[25] M. E. Peters *et al.*, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. [Online]. Available: https://aclanthology.org/N18-1202.

[26] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990. [Online]. Available: https://EconPapers.repec.org/RePEc:bla:jamest:v:41:y:1990:i:6:p:391-407.

[27] R. Bar-Haim, Y. Kantor, L. Eden, R. Friedman, D. Lahav, and N. Slonim, "Quantitative argument summarization and beyond: Cross-domain key point analysis," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 39–49. DOI: 10.18653/v1/2020.emnlp-main.3. [Online]. Available: https://aclanthology.org/2020.emnlp-main.3.

[28] M. Alshomary *et al.*, "Key point analysis via contrastive learning and extractive argument summarization," in *Proceedings of the 8th Workshop on Argument Mining*, Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 184–189. DOI: 10.18653/v1/2021.argmining-1.19. [Online]. Available: https://aclanthology.org/2021.argmining-1.19.

[29] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Stanford Digital Library Technologies Project, Tech. Rep., 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.1768.

[30] O. Sultan, R. Dhahri, Y. Mardan, T. Eder, and G. Groh, *From judgement's premises towards key points*, 2022. arXiv: 2212.12238 [cs.CL].

[31] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *J. Artif. Int. Res.*, vol. 22, no. 1, 457–479, Dec. 2004, ISSN: 1076-9757.

[32] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of Research and Development*, vol. 2, no. 2, pp. 159–165, 1958. DOI: 10.1147/rd.22.0159.

[33] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20, JMLR.org, 2020.

[34] H. Li, V. Schlegel, R. Batista-Navarro, and G. Nenadic, "Do you hear the people sing? key point analysis via iterative clustering and abstractive summarisation," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 14 064–14 080. DOI: 10.18653/v1/2023.acl-long.786. [Online]. Available: https://aclanthology.org/2023.acl-long.786.

[35] M. Grootendorst, "Bertopic: Neural topic modeling with a class-based tf-idf procedure," *arXiv preprint arXiv:2203.05794*, 2022.

[36] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," 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)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992.

DOI: 10.18653/v1/D19-1410. [Online]. Available: https://aclanthology.org/D19-1410.

[37] L. McInnes, J. Healy, N. Saul, and L. Großberger, "Umap: Uniform manifold approximation and projection," *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018. DOI: 10.21105/joss.00861. [Online]. Available: https://doi.org/10.21105/joss.00861.

[38] L. McInnes, J. Healy, and S. Astels, "Hdbscan: Hierarchical density based clustering," *Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017. DOI: 10.21105/joss.00205. [Online]. Available: https://doi.org/10.21105/joss.00205.

[39] 2022. arXiv: 2210.11416 [cs.LG].

[40] W. Yuan, G. Neubig, and P. Liu, "Bartscore: Evaluating generated text as text generation," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 27 263–27 277. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/e4d2b6e6fdeca3e60e0f1a62fee3d9dd-Paper.pdf.

[41] B. J. Frey and D. Dueck, "Clustering by Passing Messages Between Data Points," *Science*, vol. 315, no. 5814, p. 972, Feb. 2007. DOI: 10.1126/science.1136800.

[42] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995. DOI: 10.1109/34.400568.

[43] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds., vol. 14, MIT Press, 2001. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf.

[44] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *SIGMOD*, 1996, pp. 103–114. [Online]. Available: http://delivery.acm.org/10.1145/240000/233324/p103-zhang.pdf?ip=141.51.215.183&acc=ACTIVE%20SERVICE&CFID=109484574&CFTOKEN=55327820&__acm__=1339449941_7d7a4ec8bb626019f7a036183362486d.

[45] A. Toledo *et al.*, "Automatic argument quality assessment - new datasets and methods," 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)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5625–5635. DOI: 10.18653/v1/D19-1564. [Online]. Available: https://aclanthology.org/D19-1564.

[46] P. Liu, C. Huang, and L. Mou, "Learning non-autoregressive models from search for unsupervised sentence summarization," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 7916–7929. DOI: 10.18653/v1/2022.acl-long.545. [Online]. Available: https://aclanthology.org/2022.acl-long.545.

[47] M. Grusky, "Rogue scores," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 1914–1934. DOI: 10.18653/v1/2023.acl-long.107. [Online]. Available: https://aclanthology.org/2023.acl-long.107.

[48] K. S. Hasan and V. Ng, "Why are you taking this stance? identifying and classifying reasons in ideological debates," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 751–762. DOI: 10.3115/v1/D14-1083. [Online]. Available: https://aclanthology.org/D14-1083.

[49] Y. Zou, X. Zhang, W. Lu, F. Wei, and M. Zhou, "Pre-training for abstractive document summarization by reinstating source text," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 3646–3660. DOI: 10.18653/v1/2020.emnlp-main.297. [Online]. Available: https://aclanthology.org/2020.emnlp-main.297.

[50] W. Xiao, I. Beltagy, G. Carenini, and A. Cohan, "PRIMERA: Pyramid-based masked sentence pre-training for multi-document summarization," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 5245–5263. DOI: 10.18653/v1/2022.acl-long.360. [Online]. Available: https://aclanthology.org/2022.acl-long.360.

[51] T. Goodwin, M. Savery, and D. Demner-Fushman, "Flight of the PEGASUS? comparing transformers on few-shot and zero-shot multi-document abstractive summarization," in *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 5640–5646. DOI: 10.18653/v1/2020.coling-main.494. [Online]. Available: https://aclanthology.org/2020.coling-main.494.

[52] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *CoRR*, vol. abs/1910.10683, 2019. arXiv: 1910.10683. [Online]. Available: http://arxiv.org/abs/1910.10683.

[53] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20, JMLR.org, 2020.

[54] T. Yu, Z. Liu, and P. Fung, "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*, Online: Association for Computational Linguistics, Jun. 2021, pp. 5892–5904. DOI: 10.18653/v1/2021.naacl-main.471. [Online]. Available: https://aclanthology.org/2021.naacl-main.471.

[55] A. Fabbri *et al.*, "Improving zero and few-shot abstractive summarization with intermediate fine-tuning and data augmentation," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online: Association for Computational Linguistics, Jun. 2021, pp. 704–717. DOI: 10.18653/v1/2021.naacl-main.57. [Online]. Available: https://aclanthology.org/2021.naacl-main.57.

[56] Y. Wang, Z. Zhang, and R. Wang, "Element-aware summarization with large language models: Expert-aligned evaluation and chain-of-thought method," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 8640–8665. DOI: 10.18653/v1/2023.acl-long.482. [Online]. Available: https://aclanthology.org/2023.acl-long.482.

[57] S. Syed, D. Schwabe, K. Al-Khatib, and M. Potthast, "Indicative summarization of long discussions," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 2752–2788. DOI: 10.18653/v1/2023.emnlp-main.166. [Online]. Available: https://aclanthology.org/2023.emnlp-main.166.

[58] G. Adams, A. Fabbri, F. Ladhak, E. Lehman, and N. Elhadad, "From sparse to dense: GPT-4 summarization with chain of density prompting," in *Proceedings of the 4th New Frontiers in Summarization Workshop*, Y. Dong, W. Xiao, L. Wang, F. Liu, and G. Carenini, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 68–74. DOI: 10.18653/v1/2023.newsum-1.7. [Online]. Available: https://aclanthology.org/2023.newsum-1.7.

[59] Y. Tang, R. Puduppully, Z. Liu, and N. Chen, "In-context learning of large language models for controlled dialogue summarization: A holistic benchmark and empirical analysis," in *Proceedings of the 4th New Frontiers in Summarization Workshop*, Y. Dong, W. Xiao, L. Wang, F. Liu, and G. Carenini, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 56–67. DOI: 10.18653/v1/2023.newsum-1.6. [Online]. Available: https://aclanthology.org/2023.newsum-1.6.

[60] J. Wang *et al.*, "Zero-shot cross-lingual summarization via large language models," in *Proceedings of the 4th New Frontiers in Summarization Workshop*, Y. Dong, W. Xiao, L. Wang, F. Liu, and G. Carenini, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 12–23. DOI: 10.18653/v1/2023.newsum-1.2. [Online]. Available: https://aclanthology.org/2023.newsum-1.2.

[61] K. M. Hermann *et al.*, "Teaching machines to read and comprehend," *CoRR*, vol. abs/1506.03340, 2015. arXiv: 1506.03340. [Online]. Available: http://arxiv.org/abs/1506.03340.

[62] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 1797–1807. DOI: 10.18653/v1/D18-1206. [Online]. Available: https://aclanthology.org/D18-1206.

[63] H. D. Lasswell, "The structure and function of communication in society," vol. 37(1):136–139, 1948. [Online]. Available: https://www.scinapse.io/papers/2290526371.

[64] Y. Tang *et al.*, "Multilingual translation from denoising pre-training," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, Online: Association for Computational Linguistics, Aug. 2021, pp. 3450–3466. DOI: 10.18653/v1/2021.findings-acl.304. [Online]. Available: https://aclanthology.org/2021.findings-acl.304.

[65] P. He *et al.*, "Z-code++: A pre-trained language model optimized for abstractive summarization," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 5095–5112. DOI: 10.18653/v1/2023.acl-long.279. [Online]. Available: https://aclanthology.org/2023.acl-long.279.

[66] A. Pagnoni, A. Fabbri, W. Kryscinski, and C.-S. Wu, "Socratic pretraining: Question-driven pretraining for controllable summarization," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 12 737–12 755. DOI: 10.18653/v1/2023.acl-long.713. [Online]. Available: https://aclanthology.org/2023.acl-long.713.

[67] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *CoRR*, vol. abs/2004.05150, 2020. arXiv: 2004.05150. [Online]. Available: https://arxiv.org/abs/2004.05150.

[68] L. Murakhovs'ka, C.-S. Wu, P. Laban, T. Niu, W. Liu, and C. Xiong, "MixQG: Neural question generation with mixed answer types," in *Findings of the Association for Computational Linguistics: NAACL 2022*, Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 1486–1497. DOI: 10.

18653/v1/2022.findings-naacl.111. [Online]. Available: https://aclanthology.org/2022.findings-naacl.111.

[69] Y. Chen *et al.*, "UniSumm and SummZoo: Unified model and diverse benchmark for few-shot summarization," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 12 833–12 855. DOI: 10.18653/v1/2023.acl-long.718. [Online]. Available: https://aclanthology.org/2023.acl-long.718.

[70] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online: Association for Computational Linguistics, Aug. 2021, pp. 4582–4597. DOI: 10.18653/v1/2021.acl-long.353. [Online]. Available: https://aclanthology.org/2021.acl-long.353.

[71] N. Oved and R. Levy, "PASS: Perturb-and-select summarizer for product reviews," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online: Association for Computational Linguistics, Aug. 2021, pp. 351–365. DOI: 10.18653/v1/2021.acl-long.30. [Online]. Available: https://aclanthology.org/2021.acl-long.30.

[72] P. Laban, A. Hsi, J. Canny, and M. A. Hearst, "The summary loop: Learning to write abstractive summaries without examples," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 5135–5150. DOI: 10.18653/v1/2020.acl-main.460. [Online]. Available: https://aclanthology.org/2020.acl-main.460.

[73] P. Roit *et al.*, "Factually consistent summarization via reinforcement learning with textual entailment feedback," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 6252–6272. DOI: 10.18653/v1/2023.acl-long.344. [Online]. Available: https://aclanthology.org/2023.acl-long.344.

[74] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," *CoRR*, vol. abs/1612.00563, 2016. arXiv: 1612.00563. [Online]. Available: http://arxiv.org/abs/1612.00563.

[75] B. Tan, L. Qin, E. Xing, and Z. Hu, "Summarizing text on any aspects: A knowledge-informed weakly-supervised approach," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 6301–6309. DOI: 10.18653/v1/2020.emnlp-main.510. [Online]. Available: https://aclanthology.org/2020.emnlp-main.510.

[76] A. W. Yu, D. Dohan, Q. Le, T. Luong, R. Zhao, and K. Chen, "Fast and accurate reading comprehension by combining self-attention and convolution," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=B14TlG-RW.

[77] R. Speer, J. Chin, and C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge," May 2019.

[78] M. T. R. Laskar, E. Hoque, and J. X. Huang, "WSL-DS: Weakly supervised learning with distant supervision for query focused multi-document abstractive summarization," in *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 5647–5654. DOI: 10.18653/v1/2020.coling-main.495. [Online]. Available: https://aclanthology.org/2020.coling-main.495.

[79] M.-Q. Pham, S. Indurthi, S. Chollampatt, and M. Turchi, "Select, prompt, filter: Distilling large language models for summarizing conversations," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 12 257–12 265. DOI: 10.18653/v1/2023.emnlp-main.753. [Online]. Available: https://aclanthology.org/2023.emnlp-main.753.

[80] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," *CoRR*, vol. abs/1908.08345, 2019. arXiv: 1908.08345. [Online]. Available: http://arxiv.org/abs/1908.08345.

[81] E. Chu and P. J. Liu, "Unsupervised neural multi-document abstractive summarization," *CoRR*, vol. abs/1810.05739, 2018. arXiv: 1810.05739. [Online]. Available: http://arxiv.org/abs/1810.05739.

[82] C. Baziotis, I. Androutsopoulos, I. Konstas, and A. Potamianos, "SEQˆ3: Differentiable sequence-to-sequence-to-sequence autoencoder for unsupervised abstractive sentence compression," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 673–681. DOI: 10.18653/v1/N19-1071. [Online]. Available: https://aclanthology.org/N19-1071.

[83] F. Carichon, F. Fettu, and G. Caporossi, "Unsupervised update summarization of news events," *Pattern Recognition*, vol. 144, p. 109 839, 2023, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2023.109839. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003132032300537X.

[84] R. K. Amplayo and M. Lapata, "Unsupervised opinion summarization with noising and denoising," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational

Linguistics, Jul. 2020, pp. 1934–1945. DOI: 10.18653/v1/2020.acl-main.175. [Online]. Available: https://aclanthology.org/2020.acl-main.175.

[85] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *ArXiv*, vol. 1409, Sep. 2014.

[86] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/29921001f2f04bd3baee84a12 Paper.pdf.

[87] A. Bražinskas, M. Lapata, and I. Titov, "Few-shot learning for opinion summarization," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 4119–4135. DOI: 10.18653/v1/2020.emnlp-main.337. [Online]. Available: https://aclanthology.org/2020.emnlp-main.337.

[88] R. Schumann, L. Mou, Y. Lu, O. Vechtomova, and K. Markert, "Discrete optimization for unsupervised sentence summarization with word-level extraction," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 5032–5042. DOI: 10.18653/v1/2020.acl-main.452. [Online]. Available: https://aclanthology.org/2020.acl-main.452.

[89] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks," vol. 2006, Jan. 2006, pp. 369–376. DOI: 10.1145/1143844.1143891.

[90] J. Besag, "Statistical analysis of non-lattice data," *The Statistician*, vol. 24, pp. 179–195, 1975.

[91] R. Kohita, A. Wachi, Y. Zhao, and R. Tachibana, "Q-learning with language model for edit-based unsupervised summarization," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 470–484. DOI: 10.18653/v1/2020.emnlp-main.34. [Online]. Available: https://aclanthology.org/2020.emnlp-main.34.

[92] X. Zhang, F. Wei, and M. Zhou, "HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5059–5069. DOI: 10.18653/v1/P19-1499. [Online]. Available: https://aclanthology.org/P19-1499.

[93] Y. Wang, Q. Mao, J. Liu, W. Jiang, H. Zhu, and J. Li, "Noise-injected consistency training and entropy-constrained pseudo labeling for semi-supervised extractive summarization," in *Proceedings of the 29th International Conference on Computational Linguistics*, Gyeongju, Republic of Korea: International Committee on Computational Linguistics, Oct. 2022, pp. 6447–6456. [Online]. Available: https://aclanthology.org/2022.coling-1.561.

[94] S. Basu Roy Chowdhury, N. Monath, K. Dubey, A. Ahmed, and S. Chaturvedi, "Unsupervised opinion summarization using approximate geodesics," in *Proceedings of the 4th New Frontiers in Summarization Workshop*, Y. Dong, W. Xiao, L. Wang, F. Liu, and G. Carenini, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 105–120. DOI: 10.18653/v1/2023.newsum-1.11. [Online]. Available: https://aclanthology.org/2023.newsum-1.11.

[95] H. Zheng and M. Lapata, "Sentence centrality revisited for unsupervised summarization," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 6236–6247. DOI: 10.18653/v1/P19-1628. [Online]. Available: https://aclanthology.org/P19-1628.

[96] R. Mihalcea and P. Tarau, "TextRank: Bringing order into text," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 404–411. [Online]. Available: https://aclanthology.org/W04-3252.

[97] X. Liang *et al.*, "An efficient coarse-to-fine facet-aware unsupervised summarization framework based on semantic blocks," in *Proceedings of the 29th International Conference on Computational Linguistics*, Gyeongju, Republic of Korea: International Committee on Computational Linguistics, Oct. 2022, pp. 6415–6425. [Online]. Available: https://aclanthology.org/2022.coling-1.558.

[98] S. Basu Roy Chowdhury, C. Zhao, and S. Chaturvedi, "Unsupervised extractive opinion summarization using sparse coding," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 1209–1225. DOI: 10.18653/v1/2022.acl-long.86. [Online]. Available: https://aclanthology.org/2022.acl-long.86.

[99] B. Dumitrescu and P. Irofti, *Dictionary Learning Algorithms and Applications*. 2018.

[100] S. Angelidis, R. K. Amplayo, Y. Suhara, X. Wang, and M. Lapata, "Extractive opinion summarization in quantized transformer spaces," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 277–293, 2021. DOI: 10.1162/tacl_a_00366. [Online]. Available: https://aclanthology.org/2021.tacl-1.17.

[101] Y. Suhara, X. Wang, S. Angelidis, and W.-C. Tan, "OpinionDigest: A simple framework for opinion summarization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 5789–5798. DOI: 10.18653/v1/2020.acl-main.513. [Online]. Available: https://aclanthology.org/2020.acl-main.513.

[102] J. Zhou and A. Rush, "Simple unsupervised summarization by contextual matching," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5101–5106. DOI: 10.18653/v1/P19-1503. [Online]. Available: https://aclanthology.org/P19-1503.

[103] X. Fu, Y. Zhang, T. Wang, X. Liu, C. Sun, and Z. Yang, "RepSum: Unsupervised dialogue summarization based on replacement strategy," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online: Association for Computational Linguistics, Aug. 2021, pp. 6042–6051. DOI: 10.18653/v1/2021.acl-long.471. [Online]. Available: https://aclanthology.org/2021.acl-long.471.

[104] P. West, A. Holtzman, J. Buys, and Y. Choi, "BottleSum: Unsupervised and self-supervised sentence summarization using the information bottleneck principle," 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)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3752–3761. DOI: 10.18653/v1/D19-1389. [Online]. Available: https://aclanthology.org/D19-1389.

[105] T. Hosking, H. Tang, and M. Lapata, "Attributable and scalable opinion summarization," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 8488–8505. DOI: 10.18653/v1/2023.acl-long.473. [Online]. Available: https://aclanthology.org/2023.acl-long.473.

[106] Z. Miao, Y. Li, X. Wang, and W.-C. Tan, "Snippext: Semi-supervised opinion mining with augmented data," in *Proceedings of The Web Conference 2020*, ser. WWW '20, Taipei, Taiwan: Association for Computing Machinery, 2020, 617–628, ISBN: 9781450370233. DOI: 10.1145/3366423.3380144. [Online]. Available: https://doi.org/10.1145/3366423.3380144.

[107] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, 1771–1800, Aug. 2002, ISSN: 0899-7667. DOI: 10.1162/089976602760128018. [Online]. Available: https://doi.org/10.1162/089976602760128018.

[108] N. Tishby, C. Pereira, and W. Bialek, "The information bottleneck method," *Proceedings of the 37th Allerton Conference on Communication, Control and Computation*, vol. 49, Jul. 2001.

Appendix A: Unsupervised Summarization

In this chapter we cover the recent trends in unsupervised summarization by reviewing papers published in the field and propose a new way of categorizing methods by the techniques they employ (see Table A.1 for the list of works). We first separate extractive and abstractive from each other, mainly because abstractive and extractive methods are usually different in their techniques and models. We further break down abstractive models into language model-based methods, methods that use reconstruction networks, and other methods. The language models category contains recent research that uses pretrained and large language models for summary generation. The reconstruction networks consist of methods that try to train a model with reconstruction as one of their objective functions. This category mainly contains abstractive methods prior to the introduction of large pretrained language model. The last category of abstractive models includes other methods that could not be classified into the other two categories. Extractive methods are categorized into three sub-classes, classification, ranking, and search-based. The classification methods perform binary classification on each unit (word, phrase, sentence, or document) to decide whether they are salient enough to be included in the summary. Ranking methods use different techniques to score and rank units and choose the best ones as the summary. Search-based methods improve the summary by editing and evaluating the summary iteratively. Lastly, we introduce another category called hybrid methods, which use extractive and abstractive summarization techniques. Methods in this

section either use both extractive and abstractive techniques for summarization; or introduce a training strategy that can be applied to both extractive and abstractive methods. Figure A.1 outlines the proposed structure.

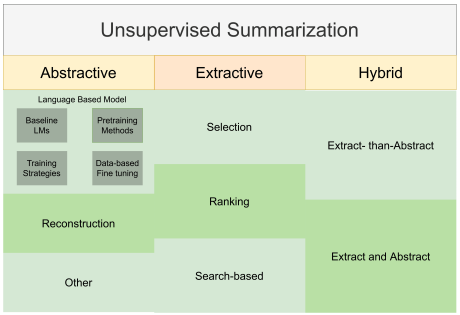

Figure A.1: Hierarchical Structure of Our Taxonomy

A.1 Abstractive Methods

A.1.1 Language Models

Pretrained language models (PLMs) and large language models (LLMs) have recently transformed the landscape of NLP and impacted almost every NLP task including text summarization. Both models are transformer-based [9] models that are pre-trained on large corpora for various NLP tasks. Similar to other natural language generation and natural language understanding tasks, summarization can significantly benefit from models pretrained on a large corpus of data. In fact, these models often generate satisfactory summaries even without any fine-tuning or task-specific training; as a result, they are often used as baseliens in recent research [49, 50]. Their performance can further be improved using few-shot training with as few as ten training examples as shown by [51]. Abstractive summarization aims to generate a summary based on its understanding of the input text. This enables both encoder-decoder architecture and decoder-only architecture to be applicable for the task. While the encoder-decoder architecture was more popular prior to 2023, the introduction and the following popularity of ChatGPT shifted the focus in research towards decoder-

only LLMs 1.1. Moreover, the research utilizing encoder-decoder PLMs often centered around trained and fine-tuned the models with various strategies in order to improve their summarization capabilities; whereas the recent LLM-based research often focus on prompt-engineering instead of fine-tuning. This may be a result of computational power required for fine-tuning LLMs and availability of models and their weights to the public.

This section covers **Baseline Language Models**, **Pre-training Methods** for language models, **Training Strategies** for language models, and **Fine-tuning** PLMs for summary generation.

A.1.2 Baseline Language Models

Baseline language models section focuses on research done utilizing Pretrained Language Models (PLMs) and Large Language Model (LLMs) without applying any training or fine-tuning. These work discuss the performance of LMs in either zero-shot or few-shot setting, without modifying the model's weights.

3.1.1.1 Pretrained Language Models

BART [17], T5 [52], and PEGASUS [53] are three of most popular encoder-decoder PLMs utilized in summarization task. They show strong performance in zero-shot, few-shot, and fully fine-tuned settings [51], and are often used as baselines for performance evaluation [50, 54]. According to [55], in some case summaries generated by these models were favored over human-written ones. BART and T5 are general-purpose pre-trained language models. The former is trained on document rotation, sentence permutation, text-infilling, and token masking and deletion objectives, while the latter is trained on token masking, translation, classification, reading comprehension, and summarization. Contrastively, PEGASUS was explicitly trained for summarization using self-supervised masked language modeling and gap-sentence-generation (GSG) objective [53]. In GSG, the model has to predict the masked sentence conditioned on the other sentences in the input, where the masked sentence is the a salient

sentence resembling the summary.

Regarding the performance of these PLMs, the authors of PEGASUS report that PEGASUS outperforms both BART and T5 on certain datasets. [51] shows that the performance of the three models is dependant on the dataset and reports both T5 and BART outperforming PEGASUS on some datasets in zero-shot setting. As for few-shot setting, PEGASUS showed significant improvements compared to other two models and was the more consistent model across various datasets on average [51].

3.1.1.2 Large Language Models

The recent emergence and success of GPT-based models by OpenAI has made them a popular choice for many NLP tasks, including summarization. ChatGPT is a large language model, with considerably more parameters, trained for next word prediction and fine-tuned with reinforcement learning from human feedback. Due to their recency, LLMs such as ChatGPT, GPT-4, and LLAMA-2 are rather understudied for downstream NLP tasks, e.g. summarization, however recent works aim to examine the capabilities of LLMs. Works such as [56], [57], [58], [59], and [60] focus on prompt engineering and utilizing LLMs in zero-shot and few-shot (also called in-context learning) setting, and/or aim to compare the evaluate and compare performance of various LLMs.

[56] **examine reference summaries** of two popular summarization datasets, CNN/DailyMail [61] and BBC XSum [62], and compare the performance of pre-trained language models, BART, T5, and PEGASUS, to large language models, GPT-3. Authors aim to address the current reference summaries issues (e.g. factual hallucination and information redundancy), by **writing new reference summaries** based on "Lasswell Communication Model" [63]. They further show that zero-shot large language models generate on par, and even better outputs than pre-trained language models, according to automatic evaluation metrics (ROUGE and BERTScore) with the new reference summaries. Authors also utilize a two-step **Chain-of-Thought**

prompting method in GPT-3 that first extracts the important information and generates the summary from extracted information.

[57] and [58] explore the ideal structure and properties of a reference summary. [57] presents an unsupervised approach for summarizing long discussions on online forums using various LLMs. The authors suggest generating an "indicative" summary of long discussions, where only the gist of different aspects in topic are presented as the summary (similar to a book's table of content); as opposed to "informative" summaries that aim to capture as much information as possible. The authors use a three step approach for creating the summary. First, the sentences of all the arguments are clustered. Second, an LLM is used to generate a single line summary of each cluster, called a label. Finally, the labels are assigned to one or more frames, where the frames are different aspects of a topic (e.g. a topic about a political idea can have aspects such as economy or ideology). The experiments show that while open-source LLMs (LLAMA and T0) have acceptable results, they are outperformed by OpenAI's GPT models (ChatGPT and GPT4). [58] analyze GPT-4 generated summaries and the optimal amount of "density" for summaries. The authors argue that an ideal summary should be entity-centric without being too difficult to understand, and the level of entities included in a summary determines its density. The authors propose a technique called "Chain of Density Prompting", that enables iteratively prompting a LLM to generate more dense summaries. Their experiments show that human written summaries have a higher density compared to vanilla GPT-4 summaries, and that the humans prefer summaries with densities closer to human written summaries.

[59] and [60] benchmark the performance of various LLMs in specific summarization tasks. The work of [59] focuses on analyzing the performance of various LLMs in dialogue summarization. Specifically, the authors conduct various experiments for generating both controlled (e.g. length control) and uncontrolled summaries using in-context or few-shot learning. Their experiments show that LLMs perform reasonably well, and models such as LLAMA and Alpaca achieve high factual consistency scores.

[60] focus on cross-lingual summarization, i.e. summarizing input text in another language, by comparing the performance of different LLMs in zero-shot setting. Their experiments showcase that ChatGPT and GPT-4 perform well and produce detailed summaries. Furthermore, they report that GPT-4's performance in zero-shot setting is comparable to an mBART-50 [64] fine-tuned for the task.

A.1.3 Pretraining Methods

[49] and [65] focus on introducing more efficient training objectives for improving the performance of pretrained language models, enabling them to outperform larger pretrained language models that are trained on more data. In [49], the authors introduce three unsupervised pretraining objectives for training sequence-to-sequence based models. They argue that the proposed unsupervised objectives, sentence reordering, next sentence generation, and masked document generation, are closely related to the summarization task, resulting in improved abstractive summaries. Using RoBERTa [10] as the encoder, they show that their model achieves comparable results with only 19 GB of data compared to PLMs trained on more than 160 GB of text. [65] introduces a new encoder-decoder pre-trained language model for summarization. The authors similarly propose three techniques for improving the pre-training stage. First, the model is pre-trained on replaced token detection and corrupted span prediction, and is then on trained document-summary pairs for summary generation. The second technique replaces the self-attention layers in the encoder with the disentangled attention layer. Disentangled attention represents each token with two vectors, content and position, to improve the effectiveness of the model. Third, they use the fusion-in-encoder for handling long sequences.

[50] and [66] introduce pretraining objectives for specific tasks and goals. [50] focus on an objective function for multi-document summarization, utilizing the gap sentence generation objective. They improve upon the gap sentence generation objective by clustering related sentences together, and masking the most informative

sentence in the cluster, using the rest of the sentences in that cluster to predict the masked sentence. After applying their pretraining method on Longformer-Encoder-Decoder [67], they report noticeable improvements in zero-shot setting and marginal improvements after supervised training. [66] introduces an unsupervised pre-training objective to improve controllability of summaries. In this method, the summarization model is asked to answer questions that are automatically generated based on the input document, which allows the method to make use of unlabeled documents. Specifically, the method first selects important sentences from the input document as pseudo-summaries using ROUGE, and generates questions about them using MixQG [68]. The model is trained to generate the questions and answer them given the document. This ensures that the generated summary accurately addresses the user's query by focusing on pertinent content.

[54] and [69] experiment on utilizing pretaining objective for PLMs in areas with not enough training data. [54] experiments on domain adaptation for low-resource summarization tasks such as email summarization. They experiment on further pre-training BART using three different training objectives, source domain pretraining, domain-adaptive pretraining, and task-adaptive pretraining. In source domain pre-training the model is pretrained on labeled data from the source domain (i.e. any domain with substantial labeled data that is not our target). In domain-adaptive pretraining, the model is pretrained on an unlabeled domain-related corpus (i.e. documents in the target domain). Lastly, in task-adaptive pretraining, the model is pre-trained on a smaller unlabeled domain-related corpus that is more task-relevant. Their experiments show that source domain pretraining and task-adaptive pretraining can generally improve performance, whereas the effectiveness of domain-adaptive pretraining depends on the pretraining data and the target domain task data. [69] proposes a new few-shot summarization model pre-trained for different summarization tasks on different datasets. This enables leveraging the shared knowledge available in different datasets. To accomplish this the authors use prefix-tuning [70], where the

main idea is to extract knowledge prepending and tuning additional parameters, called prefixes, before each layer of the PLM. In their approach, the authors first pre-train a summarization model with task-specific prefix vectors using multi-task pre-training. During inference on a new task, the prefix vector is fine-tuned in a few-shot setting from a universal prefix that was trained in the pre-training stage. Their experiments shows that their approach outperforms baselines and achieves comparable results to GPT-3.5.

A.1.4 Training Strategies

The models in this section employ different unsupervised training signals to fine-tune existing language models. The goal of these strategies is to improve the fluency, coverage, or factual consistency of generated summaries.

[71] focus on review summarization, arguing that previous models suffer from two problems, repetitive and factual inconsistent summaries. As a solution, they first fine-tune T5 on a small labeled review summarization model. In order to produce summaries, the model generates multiple candidate summaries for each subset of reviews. A subset is created by dropping k reviews out of the input and concatenating the rest, to increase diversity of generated candidates. Next, using human annotated coherence scores, they train a "coherence summary ranker" model to score the candidate summaries with respect to their coherence and factual consistency. The candidate with the highest score is selected as the summary. The experiments show that the proposed approach generates informative summaries while being diverse and coherent.

[72] and [73] propose new training signals to fine-tune language models using reinforcement learning. [72] introduces a novel approach for fine-tuning a language model (i.e. GPT-2) in order to optimize the fluency and coverage of keywords in the summary while enforcing a length limit as a result of model's decoder-only architecture. The authors combine two scores, fluency score and coverage score, in order to produce

a training signal for the model using the self-critical sequence training (SCST) method [74]. The fluency score is calculated using by GPT-2 using the log-probability of the generated summary. The coverage score is calculated by first masking the keywords in input document, and feeding it alongside the generated summary to a fine-tuned BERT model to predict the masked words. The accuracy of the BERT model is used as the coverage score. The proposed approach outperforms previous unsupervised summarization baselines, and are comparable to supervised approaches at the time in terms of ROUGE score. [73] focus on generating factual consistent summaries using reinforcement learning. They propose using textual entailment of input document and generated summary, as the reward for the reinforcement learning model. The idea behind the approach is that models trained on natural language inference datasets can accurately detect factual inconsistencies. However to ensure the model had summary generation capabilities, i.e. generating coherent and relevant text, it was first pre-trained to produce summaries using maximum-likelihood objective, and only further fine-tuned using reinforcement learning. Their results show improvements over baselines for factual consistency, salience, and conciseness.

A.1.5 Data based Fine-tuning

The methods in this section generate or collect a dataset in an unsupervised or weakly supervised manner, for a specific task or domain where there is not enough data available.

[55] and [75] focus on generating a labeled dataset for summarization using websites and resources available online. [55] introduced an unsupervised method for extracting pseudo-summaries from Wikipedia for fine-tuning PLMs. These pseudo-summaries have similar high-level characteristics to our ideal target summaries, where the high-level characteristics refer to length or the level of abstractiveness. They use the first k sentences in each article as the summary and the rest i sentences as the input. They further augment the dataset by using round translation [76] and is the dataset

to fine-tune BART. The experiments show that the fine-tuned BART outperforms the vanilla BART in zero and few-shot setting across various datasets. [75] aims to facilitate aspect summarization on any given aspect; however, since it is impossible to create a dataset for every aspect, they employ a weakly supervised approach for data collection. In order to collect the training data different aspects of input are extracted from a labeled summarization dataset using an NER model, and all other related aspects to it are also extracted using ConceptNet [77]. The summary of each aspect, i.e. sentences in reference summary related to the aspect (or similar aspects found in ConceptNet) are concatenated together. During summary generation, the aspect, a list of related words (collected from the Wikipedia page of the aspect), are given as context alongside the input document as a single input to the summarization model.

[78] and [79] use language models to generate labeled datasets for specific summarization tasks. [78] tackles query-focused summarization in the multi-document (QF-MDS) setting, where the goal is to generate a summary from multiple documents given a query. Due to the absence of specific training data for the task, the authors adopt a supervised training approach utilizing other QF-MDS datasets as substitutes and generate weak reference summaries from reference summaries. In the next step, the weak reference summaries are used to fine-tune BERTSUM model [80] to generate the query focused abstractive summaries. Finally, the generated summaries are ranked by their relevance to the query and most relevant ones are selected as the summary.[79] proposes an approach for generating weakly-supervised training data using LLMs. The authors argue that fine-tuning LLMs for specific tasks is infeasible due to the resources required, and collecting enough training data for fine-tuning smaller language models such as BART in order to match their performance to LLMs is expensive. Therefore, the authors propose a three step knowledge distillation technique for creating training data using LLMs. The approach generates annotated data from a large corpus of unlabeled data, with the help of a small

validation set. Specifically, the method first extracts text from the large unlabeled corpus that is semantically similar to text in the validation set. Second, it creates a prompt with the labeled data from validation set as few-shot examples, and uses ChatGPT to predict the summary for the sampled text using the prompt as input. In the final step it filters low-quality summaries generated by ChatGPT. Their results show improvements over standard knowledge distillation approaches.

Overall, methods that use PLMs or LLMs achieve better performance on average compared to other methods in this survey. However, they still do suffer from the common problems in abstractive summarization, such as factual correctness [71]. Additionally, training or fine-tuning such models poses limitations on resource and introduces new challenges. For example, [54] have shown that continues pretraining causes catastrophic forgetting, where the pretrained language model loses some of its language understanding ability gained in the initial pretraining step.

A.1.6 Reconstruction

The goal of reconstruction approach is to reconstruct the original input from the modified input. As a result it is a popular training strategy for unsupervised summarization since the original document can be used as the reference summary (output), while the source (input document) can be created by adding noise to the original document. Models trained with reconstruction loss often use an encoder-decoder architecture at their core, where they first try to encode the input document into a representation and then decode the representation to the summary. Ideally, the encoder should encode the salient information from the input, and the decoder should decode the representation into a fluent text that captures the vital information and is shorter than the input. In order to fulfill these requirements, summarization models use different techniques.

[81], [82], and [83] use auto-encoder architecture as part of their approach for unsupervised summarization. [81] uses an auto-encoder trained for reconstruction

for creating representations, [82] uses two auto-encoders one for compression and the other for reconstruction, and [83] trains two auto-encoders for language modelling and constraining length.

[82] proposes a novel architecture for summarization emphasizing on sentence compression using two attention-based encoder-decoder. The first encoder-decoder is the compressor, tasked with summarizing the input, and the second encoder-decoder reconstructs the original input from the summary. In order to train the model, four losses are computed. The first loss is the reconstruction loss between the original input and the output of the second decoder; the second is the language model prior loss to ensure the generated summary is fluent. The third loss, topic loss, encourages coverage by calculating the cosine distance between the average of word embeddings in input and summary. Lastly, the fourth loss enforces the length penalty. [81] proposes a method for multi-document summarization using an LSTM-based encoder-decoder. The model is composed of two parts; the first part is an encoder-decoder that tries to learn the representation for each input using reconstruction loss. The second part learns to generate a summary similar to each of the inputs. The second part is trained using the average similarity between the representation of each input and the representation of the summary generated from the mean of input representations. [83] focus on "update summarization" of news by iteratively updating a summary as new information is available in social media settings using autoencoders. The proposed autoencoder model is trained for two tasks simultaneously, language modeling and information constraint which is implemented by limiting the length of the text. The proposed approach, unlike previous extractive approaches, does not rely on classifying redundant text to find salient information. This is important as social media posts are concise and not repetitive as opposed to traditional news stories, which renders methods that rely on classification of redundant information useless. The proposed approach shows improvements in ROUGE scores over previous baselines.

[84] focus on unsupervised opinion summarization, and creating a synthetic dataset

by introducing noise. They introduce two linguistically motivated noise generation functions to add noise, and train the model to denoise the input. The noise generation functions add noise on word/phrase level (e.g., changing words) and on document level (replacing a review with a similar one). For the model, they use an LSTM-based encoder-decoder with attention [85] and copy [86] mechanisms in the decoder which sets it apart from the previous auto-encoder based models. They experiment on movie and business datasets and show improvements over previous baselines.

Before the language models, reconstruction networks such as [82] used to be the state-of-the-art for abstractive summarization. However, because they are trained to reconstruct input instead of summary, they cannot learn the key characteristics of summaries [87]. Additionally, due to the emphasis of methods on conciseness, they tend to mix generic statements with informative content [71]. Lastly, reconstruction networks that use autoencoders are limited to simple decoders, lacking attention and copy mechanism that has proven to be useful in summary generation [84].

A.1.7 Other

The last classification of abstractive summary includes approaches that often utilize novel techniques for summarization. The works in this section use non-autoregressive architecture, custom networks, and reinforcement learning techniques as part of their approach.

[46], propose a non-autoregressive method for sentence summarization. They use the method introduced by [88] to generate summaries of news articles and use this data to train an encoder-only transformer using Connectionist Temporal Classification (CTC) [89] algorithm for summarization. The non-autoregressive architecture enables length controlling, which can be desirable in summarization tasks. [46] claims their non-autoregressive approach to summarization using encoder-only architecture is several times faster, and better captures the input–output correspondence compared to autoregressive models. However, the performance of the proposed approach

in terms of ROUGE score is not on par with the state-of-the-art.

[87] introduce a few-shot method for review summarization. They argue that previous unsupervised methods trained on review datasets have not been exposed to actual summaries, therefore the summaries generated by them lack essential properties (e.g. writing style, informativeness, fluency, etc). To solve this, they aim to train a model to generate a summary based on reviews. Specifically, they first train an encoder-decoder transformer [9] for the review generation task on a large review dataset, using leave-one-out objective [90]. During training, the model is conditioned on the properties of reviews, such as writing style, informativeness, fluency, and sentiment preservation, however for the inference phase, when the goal is to generate summaries, not reviews, these properties are predicted by a small plug-in network trained on a handful of reviews with summaries. As a result, we can have a small model trained with a few data samples, guiding our encoder-decoder model that generates the summaries. Their experiments showed improvements over previous SOTA at the time, in both human and automatic evaluation.

[91] proposed an edit-based approach using Q-learning. Edit-based approaches are often extractive methods that start from a pseudo summary (e.g. a random selection words, or the original text) and perform continues edits to improve the summary. The edits differ depending on the work, but they often include adding words and removing them. After each edit the quality of the summary is evaluated using a scoring function to check if the edit has improved the summary or not. In [91], the method consists of an agent that selects an action (keep, delete, or replace), and a language model applies the edits to generate summary. The agent predicts an action, delete, keep, or replace, for each word; the first action deletes the word, the second keeps it as is, and the third action replaces the word with a [MASK] token. The new sentence is given to BERT in order to predict masked tokens and generate the summary. In the training phase, after the summary is generated, it is further reconstructed back to the original in order to calculate the step reward score, similar to reconstruction methods. The step

reward score combined with summary assessment, which evaluates informativeness, shortness, and fluency, is the reward that is used for calculating the loss function. Their experiments report on-par results compared to previous baselines.

A.2 Extractive Methods

Extractive methods create a summary by selecting salient units (words, phrases, or sentences) from the input document. Extractive methods often view summarization tasks either as a sequence selection task, where units are classified as to be included in the summary or not, or use a ranking technique to rank all the units by their salience and select the top k as the summary. A less researched approach is search-based summarization, where the goal is to maximize an objective function that evaluates the summary by editing it. These edits are usually composed of add, delete, and replace, and the search space is the units in the input document.

A.2.1 Classification

[92] focus on proposing an unsupervised pretraining approach for training an extractive summarization model. They argue that since large amounts of labeled data for extractive summarization is not available, using an unsupervised pretraining approach enables the model to learn from unlabeled data. Similar to how BERT learns the representation of sentences by predicting words, the authors propose an approach to learn the representation of documents by predicting sentences. Specifically, they use a hierarchical encoder for the sentence selection. The hierarchical architecture consists of two identical encoders, one for sentence-level encoding with words as inputs and the other for document-level encoding with sentence encodings as inputs and one decoder. For the training objective, masked language modeling is used where every word in 80% of sentences is masked, and the model is tasked with predicting the masked tokens. For summary generation, the output of the second encoder is used to predict a true or false label for each sentence in the document. Their experiments

show some improvements in ROUGE compared to baselines.

[93] proposes a semi-supervised approach using consistency training. Consistency training aims to make the model resilient to slight changes by introducing noise and tasking the model to generate consistent summaries. To implement this, authors inject noise by replacing words with semantically similar words using BERT and train the model to generate similar summaries from original and noisy inputs. However, training using consitency training is not enough by itself, as there is not enough labeled data for the task. To this end, the authors also propose an entropy-constrained pseudo-labeling strategy that is used for getting high-confidence labels from unlabeled data. The pseudo label is assigned by comparing the entropy of the predicted result of unlabeled data and the predicted result of labeled data, where the pseudo label is preserved if the entropy of unlabeled data is smaller than the entropy of labeled data. Their experiments show slight improvements over baselines.

[94] proposes a two-step opinion summarization method based on representation learning and a geodesic distance-based selection function. The proposed approach first generates topical representations of input sentences using dictionary learning. Next it uses its selection function to select sentences as the summary, based on the geodesic distance between their representations. The goal of the approach is to extract sentences from opinions that are shared between users as it shows their importance. The authors claim that the learned topical representations using dictionary learning better capture the semantics of a text, compared to pre-trained models. Moreover, they use a geodesic distance-based function for computing the importance of a review, as it considers the "underlying manifold" of representations. Their experiments showcase that their method outperforms previous unsupervised opinion summarization methods.

In addition to the common problems of extractive summarization, selection methods tend to keep the relative order of sentences in the input, which is a limitation [49]. Also, generating datasets for these models can be difficult since they require

sentence-level labels [92].

A.2.2 Ranking

[95] improve upon a popular graph-based baseline, TextRank [96]. TextRank first creates a graph from input, where sentences are the nodes, and the edges are the similarity scores between them. Next, the centrality of each node (i.e. sentence) is calculated, the nodes are ranked by their centrality and the top ones are chosen as the summary. The authors improve upon this by making two changes. First, they used BERT for generating sentence representations, improving the accuracy of sentence similarities, and second, they made the edges in the graph directed. This allows taking the relative position of sentences with respect to each other into account, which enables prioritizing earlier sentences in a document that are more general. Their experiments show signifcant improvement over original TextRank but perform slightly worse than previous SOTA.

[97] focus on summarizing long documents by introducing a two-step ranking method using semantic blocks. Semantic blocks are consecutive sentences in a document that describe the same facet. To achieve this, proposed method first finds all the semantic blocks in the input and filters insignificant facets using a centrality estimator. In the next stage, relevant sentences to facets in each block are selected as candidates, and the final summary is selected from candidates using a sentence-level centrality-based estimator. The authors use the previously mentioned approach [95] as the centrality-based estimator. The proposed approach has a similar performance in terms of ROUGE compared to previous SOTA, however the authors report significant improvements in inference speed.

[98] uses dictionary learning [99] and builds upon the work of [100] which introduced Quantized Transformer (QT) for review summarization. Dictionary learning or sparse coding aims to find a sparse representation of the input data over latent semantic units. The authors use dictionary learning to improve upon the previous

work by representing sentences as a distribution over latent units (as opposed to a single latent representation), and aspect-focused summaries. In their approach, the sentences in reviews are the inputs, and the representations are learned by training an encoder-decoder transformer on reconstruction. In summary generation, the mean representation of all the review sentences is calculated, then the relevance score between each sentence and the mean representation is computed, and finally, the top k sentences with the highest score are selected as the summary. Their experiments show improvements over the previous extractive SOTA including QT.

While ranking methods enable extraction of the most salient parts of the input, resulting in a high coverage score, the output summary they generate lacks coherence and cohesion since all the selected sentences are concatenated together.

A.2.3 Search-based

[88] propose a hill-climbing search-based method for sentence summarization that iteratively improves the summary. They initialize by selecting k random words from the input with order intact, with k being the desired length. At each time step, a new sentence summary is sampled by randomly removing a word summary and selecting another from the original sentence while preserving the relative order of words. The new sentence summary is selected if it achieves a higher score computed by an objective function that evaluates fluency (using language model perplexity) and similarity score between the summary and original sentence. As mentioned by [46], the search-based methods are slow at inference as the methods need hundreds of search steps for each input sentence. These methods are also often restricted to keep the same word order as the input which affects their coherence.

Table A.1: Categorizing works in unsupervised summarization by their approach.

	Works	Method	Pros and Cons
Abstractive	[17]	PLM	
	[52]	PLM	
	[53]	PLM	
	[56]	LLM	
	[58]	LLM	
	[59]	LLM	
	[60]	LLM	
	[57]	LLM	
	[49]	LM:PM	Pros:
	[50]	LM:PM	High performance,
	[54]	LM:PM	Ability to understand
	[69]	LM:PM	and generate new
	[66]	LM:PM	words/sentences
	[65]	LM:PM	Cons:
	[71]	LM:TS	High training
	[72]	LM:TS	and inference cost,
	[73]	LM:TS	Hallucination,
	[55]	LM:DFT	Text degeneration
	[75]	LM:DFT	Topic drift,
	[78]	LM:DFT	Factual correctness
	[79]	LM:DFT	
	[84]	Reconstruction	
	[82]	Reconstruction	
	[81]	Reconstruction	
	[83]	Reconstruction	

Continued on next page

Table A.1 – continued from previous page

	Works	Method	Pros and Cons
	[87]	Other	
	[46]	Other	
	[91]	Other	
Extractive	[92]	Selection	Pros:
	[93]	Selection	Faster training
	[94]	Selection	and inference,
	[95]	Ranking	Cons:
	[97]	Ranking	Worse performance
	[98]	Ranking	Fluency and coherence
	[88]	Search-based	
Hybrid	[101]	Ext-than-Abs	Pros:
	[102]	Ext&Abs	Scalable,
	[103]	Ext&Abs	Modular
	[104]	Ext&Abs	Cons:
	[105]	Ext&Abs	Complex architecture

A.3 Hybrid Methods

As the name suggests, hybrid models use both extractive and abstractive models in their summary generation process. Some of these methods use an extract-than-abstract method, where an extractive technique is used to select only parts of the input, then an abstractive model is used to generate the summary from the extracted input. The other line of work either focuses on training strategies (rather than extractive or abstractive models), where the training strategy can be applied to both extractive and abstractive models. Or use both extractive and abstractive techniques for summarization.

A.3.1 Extract-than-Abstract

[101] employ an extract-than-abstract approach for review summarization. Their approach to review summarization consists of three steps: extracting opinion phrases from reviews, selecting desired opinions (based on popularity or specific aspect), and a generator for constructing summaries from selected opinions. During training, they extract opinions from a review using Snippext [106] and train a transformer on generating the original review given opinions. In inference, in order to generate a summary from multiple reviews, first, the opinions are extracted and clustered, similar opinions are merged and the most popular one in each cluster is selected as input to the previously trained transformer to generate a summary. Their approach shows improvements over baselines at the time.

A.3.2 Extractive and Abstractive

[102] propose an approach using two language models for generating a summary that can be used in both abstractive and extractive setting. The model uses the product-of-experts model [107]using two decoder-only language models to predict the next word in the summary by computing the product of their probabilities. The first language model is used for contextual matching (selecting relevant information), while the other is fine-tuned to ensure fluency. This approach could used in both extractive and abstractive setting, since the vocabulary of language models could either be restricted to input document (extractive) or to the vocabulary of language model itself (abstractive). Their approach had competitive results compared to its baselines without any joint training of the models.

[104] uses the information bottleneck [108] principle as a signal for unsupervised summarization to train an extractive and an abstractive model. The information bottleneck aims to produces a summary (x) that is optimized for predicting another relevant information (Y). The authors claim that this approach is more suitable for summarization in contrast to reconstruction loss, since the goal of this method is to

retrieve relevant information instead of training to recreate all the information. For this task, X is the summary generated by the model, and Y is the next sentence, as a result the model tries to find the summary that best predicts the next sentence. The authors use this approach to train extractive model and further use the data generated by the model to train an abstractive model in a self-supervised manner. Their extractive model performs slightly better than unsupervised SOTA at the time in terms of ROUGE, however their abstractive method does not show improvements over previous methods.

[103] and [105] propose an unsupervised method for summarization that can produce both extractive or abstractive summaries. [103] proposes a self-supervised training strategy for dialogue summarization that can be applied to both abstractive and extractive approaches. The approach uses a dialogue generator model to predict the next utterance of dialogue under two scenarios. In one, they condition the dialogue generator on previous dialogues, and in the other, they condition the dialogue generator on the summary of the previous dialogues. They use the difference between summaries to calculate the loss function. They use the loss to train both abstractive and extractive models. The idea behind this approach is that a good summary should offer a replacement for the original dialogue; as a result, the next utterance of dialogue generate from both previous dialogues, and the summary should be similar. Both approaches perform better than the baselines at the time. [105] proposes a scalable and controllable approach for opinion summarization based on hierarchical encoding that can be used in both abstractive and extractive settings. The approach first encodes the inputs into a hierarchical space. Next, depending on the setting, it generates summaries either by extracting opinions belonging to the most popular encodings (extractive), or decodes the most popular encodings (abstractive). Their approach provides scalability as it can aggregate encodings of multiple documents, while also allowing for aspect-based summaries due to the hierarchical nature of their encodings. Their results show general improvements across datasets compared to

previous SOTA in terms of ROUGE for both extractive and abstractive settings.

The Extract-than-Abstract approach combines the strengths of both methods and covers their weaknesses; the extractive module finds salient pieces of information, and the abstractive module generates a summary from a much smaller input. As a result, the extractive module does not have to deal with the fluency of generated text, and the abstractive module is not conditioned on long texts, which is an issue for attention and RNN-based models. Extract-than-Abstract models perform exceptionally well when in long or multi-document settings. Extractive and abstractive strategies, on the other hand, are more flexible as they can be applied to both techniques, enabling fair comparison between the two methods. However, since these models use external extractive and abstractive models, their performance relies heavily on them. For example, RepSum [103] relies on a dialogue generation model for training, restricting the summarization performance on the performance of the auxiliary task.

Appendix B: Human Evaluation Criteria

You are given a pair of argument and key point sentences, and your have to decide whether the pair match or not

A matching pair should satisfy two conditions: 1. The argument should cover the same aspect as the key point. 2. The argument should be a clear and understandable argument regarding an aspect by itself given the topic.

You are supposed to give a score to each argument key point pair

Score 1 if the argument and key point are matching

Score -1 if the argument and key point are not matching

Score 0 if you are not sure

- You are given 80 sentences in total, 20 sentences for 4 topics. The topics are:
  - The USA is a good country to live in
  - Social media platforms should be regulated by the government
  - Abortion
  - Gay rights
- The arguments and key points can be "pro" the topic, or "against" it

Examples

- Below are some examples of matching and non-matching pairs

- Matching
  - Argument: Prevents a large number of diseases
  - Key point: Vaccines prevent diseases
  - Matching because "Prevents a large number of diseases" is understable given the topic of "Vaccination should be mandatory"
- Non Matching
  - Argument: Vaccination should not be mandatory
  - Key Point: Vaccines have side-effects
  - Non matching because the key point mentions the aspect of side effect and the argument does not
- Non Matching
  - Argument: That is not possible.
  - Key point: Vaccinating everyone is not possible
  - Non matching because the argument is not an understandable sentence by itself, i.e. it is too vague.